Deduktive Generierung von adaptierten Plänen mit Kontrollstrukturen

Dissertation
zur Erlangung des Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)
der Technischen Fakultät
der Universität des Saarlandes
von

Dietmar Dengler

nach einem Thema von
Prof. Dr. Wolfgang Wahlster

Saarbrücken
1998

Tag des Kolloquiums: 10.12.98

Dekan der Technischen Fakultät: Prof. Dr. Wolfgang Paul

1. Gutachter: Prof. Dr. Wolfgang Wahlster
2. Gutachter: Prof. Dr. Jörg Siekmann

Danksagung

Mein besonderer Dank gilt meinem Doktorvater, Herrn Prof. Dr. Wolfgang Wahlster, der mir es ermöglichte und mich darin unterstützte, im Rahmen der Projekte PHI und RAP am Deutschen Forschungszentrum für Künstliche Intelligenz (DFKI) in Saarbrücken die vorliegende Arbeit zu verfassen.

Herrn Prof. Dr. Jörg Siekmann danke ich für seine Bereitschaft, die Arbeit als Zweitgutachter zu begutachten.

Dank gebührt meinen Kolleginnen und Kollegen der Projekte PHI und RAP, die mir für anregende Diskussionen zur Verfügung standen und damit zum Abschluß der Arbeit beitrugen: Dr. Mathias Bauer, Dr. Susanne Biundo, Dr. Harald Feibel, Dr. Matthias Hecking, Dr. Jana Köhler und Gabriele Paul.

Kurzzusammenfassung

Die vorliegende Arbeit untersucht die Formalisierung und Anwendung eines generischen, deduktiven Planungsansatzes im Kontext intelligenter Assistenzsysteme. Der Ansatz ermöglicht, unterschiedliche, konsumentengerechte Planerstellungs- und verarbeitungsmethoden zu spezifizieren und zu operationalisieren, die die vielfältigen, notwendigen Planschlußfolgerungsmechanismen für Aktionsplanungsaufgaben in Assistenzsystemen entweder definieren oder zumindest unterstützen. Die Basis des Planungsverfahrens bildet eine temporale Planungslogik, für die ein kompositionales Verfeinerungs-Verfahren eingeführt wird, das es erlaubt, konstruktive Beweise von Planspezifikationen nach planungsspezifischen Gesichtspunkten zu komponieren. Die im Anwendungskontext notwendige Adaptivität von Planerstellung und resultierenden Plänen wird aus Deduktionssicht durch die Verwendung konfigurierbarer Beweisstrategien im Kontext des taktischen Theorembeweisens erreicht. Die Beweisstrategien sind ein direktes Abbild von Planungsstrategien, die an verschiedenen Adaptionspunkten durch die Verwendung entsprechender Verfeinerungsmethoden und allgemeinen oder domänenabhängigen Entscheidungsfunktionen konkretisiert werden können. Für typische Adaptionspunkte, die den Plangenerierungsprozeß detailliert in sechs korrelierte Phasen untergliedern, werden ihre Beziehungen zum Verfeinerungsverfahren analysiert und konkrete Adaptionsmerkmale und -kombinationen spezifiziert. Eine Konkretisierung des formalen Modelles zur Kontrollstrategiekonfiguration wird exemplarisch durchgeführt.

Short Abstract

This thesis describes the formal model and the application of a generic deductive planning approach investigated in the context of intelligent assistant systems. The approach allows the specification and operationalization of different consumer-oriented plan generating and plan processing methods which either define the manifold plan reasoning mechanisms for action planning tasks in assistant systems or at least support them. Constructive proofs of plan specifications can be composed on the basis of a temporal planning logic and its associated compositional refinement process. Regarding the application context, the necessary adaptation of planning and the resulting plans can be reached from a deductive viewpoint by the use of configurable proof strategies in the sense of tactical theorem proving. The proof strategies result as a direct mapping of planning strategies, which can be refined on different adaptation points by using special methods and general as well as domain specific decision functions. The relations of the plan refinement process to a typical setup of adaptation points, which splits the planning process into six correlated phases, is analyzed and concrete adaptation features and combinations are specified. A sample instantiation of the formal model for configuring control strategies is given.

Zusammenfassung

Die vorliegende Arbeit untersucht die logische Formalisierung und Anwendung eines generischen, logischen Planungsansatzes im Kontext intelligenter Assistenzsysteme. Der Ansatz ermöglicht, unterschiedliche, konsumentengerechte Planerstellungs- und verarbeitungsmethoden zu spezifizieren und zu operationalisieren, die die vielfältigen, notwendigen Planschlußfolgerungsmechanismen für Aktionsplanungsaufgaben in Assistenzsystemen entweder definieren oder zumindest unterstützen.
Aufgrund der vielfältigen Anforderungen an die Qualität von Plänen und die Vorgehensweisen ihrer Erzeugung und Verarbeitung wird ein formales Modell konfigurierbarer, dynamischer Planungsstrategien entworfen. Das Konzept des taktischen Beweisens wird dazu nach planungsspezifischen Gesichtspunkten angepaßt und auf die Konfigurierung von Taktiken/Strategien erweitert. Die Operationalisierung adaptierbarer Strategien erfolgt durch ihre Implementierung in Form ausführbarer, algebraischer Spezifikationen, für die ein dynamisches Ausführungsmodell in der logischen Programmiersprache PROLOG angegeben wird; dieses Modell erlaubt unter gewissen Einschränkungen die selbstadaptive Veränderung einer Planungsstrategie während ihrer Ausführung. Die Schnittstelle zur aktuellen Konfiguration einer Strategie ist durch eine Merkmalsalgebra definiert, für die Übersetzungsmethoden zu Taktikspezifikationen definiert werden.
Als Konkretisierung des formalen Modelles wird ein deduktiver Planungsansatz in einer modalen, temporalen Planungslogik spezifiziert und implementiert. Ein allgemeines kompositionales Verfeinerungs-Verfahren erlaubt die freie Kombinierbarkeit von progressivem und regressivem (jeweils inkrementellem) Vorgehen, sowohl aus der Perspektive von Zuständen und partiellen Plänen als auch aus einer aufgabenorientierten Betrachtungsweise heraus. Pläne mit temporalen und konditionalen Kontrollstrukturen entlang eines großen Abstraktionsspektrums können auf diese Weise erzeugt werden. Grundlage ist ein auf Aktionsmodellen basierender verallgemeinerter Planbegriff, der die aus Adaptivitätssicht nötige, stufenlose Flexibilität in der Planrepräsentation ermöglicht. Das Plankonzept macht sich zunutze, daß Pläne in der gewählten logischen Sprache als Formeln repräsentiert sind und dadurch als Informationsstrukturen sowohl intensionale als auch extensionale Beschreibungen von Aktionsintervallen beinhalten können. Aktionen sind dabei nicht mehr notwendige Basisbestandteile von Plänen, sondern stellen nur noch eine unter vielen Möglichkeiten dar, Aktionsverhalten zu charakterisieren; als allgemeines Schema zur Charakterisierung von Plänen wird das zeitliche Verhalten von Eigenschaften und Ereignissen, das auf bestimmte Weise das Erreichen spezifizierter Ziele beschreibt, benutzt.
Zur Erreichung der geforderten Adaptivität wird der logische Ansatz spezialisiert durch die Verwendung konfigurierbarer Beweisstrategien, die aus der Sicht des Planens eine direkte Abbildung von Planungsstrategien sind. Ausgehend von dem Metasprachenkonzept zur Beweisprogrammierung wird eine Metasprache zur Berücksichtigung planungsspezifischer Aspekte definiert. Sie sieht Konzepte zur Konfigurierung von Strategien/Taktiken vor und bietet Konstrukte zur Integration von Entscheidungsfunktionen an. Diese Funktionen repräsentieren in allgemeiner Weise planungsrelevante Präferenzen oder Heuristiken, beispielsweise zur Aktionsauswahl und Teilzielordnung. Unterschiedliche Planungsstrategien zur Erzeugung einer Vielzahl von Planabstraktionen und Kontrollstrategien werden spezifiziert und jeweils hinsichtlich progressiver und regressiver Vorgehensweise untersucht.

Dies geschieht, um die freie Kombinierbarkeit verschiedener Verfeinerungsstrategien zu gewährleisten.
Die technische und praktische Umsetzung des Metasprachenkonzeptes erfolgt mit den Methoden metalogischer Programmierung, da hier sowohl die Spezifikation eines Beweiskalküls als auch die geforderte Dynamik und Adaptivität nahezu optimal unterstützt werden.
Gemäß dem verfolgten flexiblen Verfeinerungsverfahren zur Planerstellung werden Beweisbausteine in Form von Taktiken für die verschiedenen planungsspezifischen Phasen mit Berücksichtigung der Schnittstelle "Planspezifikation" spezifiziert. Ein konfigurierbarer Beweisplan mit einer definierbaren Zahl von Adaptionspunkten wird als Taktikskelett spezifiziert. Als Spezifikationssprache zur Variantenspezifikation des generischen Planers wird eine deklarative Merkmalsalgebra verwendet. Die geeignete Übersetzung von Termen dieser Algebra in Taktiken und deren Integration in ein Taktikskelett definiert dann den Prozeß der Beweisstrategiekonfiguration.
Für typische Adaptionspunkte, die den Plangenerierungsprozeß detailliert in sechs korrelierte Phasen untergliedern, werden ihre Beziehungen zum Verfeinerungsverfahren analysiert und konkrete Adaptionsmerkmale und -kombinationen basierend auf den eingeführten Planungsstrategien spezifiziert. Eine Konkretisierung des formalen Modelles zur Kontrollstrategiekonfiguration wird exemplarisch durchgeführt.
In Beziehung zu bisher existierenden Planungsansätzen ergeben sich folgende Vorteile:

- Die gesamte Verarbeitungskette der Planerstellung, von der Definition und Repräsentation der Anwendungsdomäne und der gewünschten Lösungen, der Spezifikation von Kontrollstrategien, ihrer Anpassung auf eine konkrete Domäne, bis hin zur Ausführung der Generierung und Lösungsextraktion, wird als ein wohldefinierter, zusammenhängender Prozeß mit präziser Semantik beschrieben.

- Es wird ein verallgemeinerter Planbegriff zugrundegelegt, der neben Aktionen auch intensionale Beschreibungen vorsieht, um intendiertes Aktionsverhalten zu charakterisieren.

- Das formale Modell der adaptierbaren Kontrollstrategien ist unabhängig von dem konkreten logischen Formalismus zur Planrepräsentation und -erstellung und erlaubt eine flexible Festsetzung von Adaptionspunkten.

- In der konkretisierten Planungslogik können Pläne durch eine Vielzahl unterschiedlicher Kontrollstrukturen und Abstraktionsgrade beschrieben werden.

- Es besteht eine hohe Variationsmöglichkeit bei der Beschreibung von Planungsproblemen und alle erstellten Lösungen sind beweisbar korrekt.

- Die semantische Nähe von Planrepräsentation und Plangenerierung erlaubt eine unmittelbar verifizierbare Anpassung und Veränderung beider Prozesse.

- Die mit der Plangenerierung nahe verwandten Prozesse der Planinterpretation, Planverifikation und Planvalidierung erhalten in dem logischen Rahmenwerk eine natürliche Abbildung und sind mit nahezu den selben Strategien und Inferenzregeln wie die Plangenerierung durchführbar.

Abstract

This thesis describes the formal model and the application of a generic deductive planning approach investigated in the context of intelligent assistant systems. The approach allows the specification and operationalization of different consumer-oriented plan generating and plan processing methods which either define the manifold plan reasoning mechanisms for action planning tasks in assistant systems or at least support them.
Since a lot of different requirements on the quality of plans and their associated processes of generation and reasoning can be formulated, a formal model of configurable dynamic planning strategies is developed. As means of support the concept of tactical theorem proving is adapted according to planning specific purposes and it is extended in order to allow the configuration of tactics/strategies. The operationalization of adaptable strategies is reached by their implementation as executable algebraic specifications. A dynamic execution model for the specifications is defined for the logic programming language PROLOG. This model allows under certain restrictions the self-adaptation of a planning strategy during its execution. An algebra of features with its related compilation into tactic specifications defines the interface to the current configuration of a strategy.
An instantiation of the formal model is given by the specification and implementation of a deductive planning approach which is based on a modal temporal planning logic. Thereby, a general compositional refinement process allows a restriction-free combination of progressive and regressive planning methods. This is possible from a state-based, plan-based, as well as task-based perspective. The basis for the generation of plans along a wide spectrum of abstractions is a general plan notion based on action models which allows to represent plans with the necessary homogeneous flexibility. The plan concept benefits from the fact that in the planning logic used plans are represented as formulas. In this way they can be specialized information structures consisting of intensional as well as extensional descriptions of action intervals. Actions are no longer a necessary part of plans but only specify one of many possibilities in order to characterize action behavior. The temporal behavior of properties and events which describes in a certain way the satisfaction of specified goals is used as the general scheme for the characterization of plans.
The logical approach is specialized by the use of configurable proof strategies which are direct mappings of planning strategies. On basis of the meta language concept of proof programming a new meta language for considering planning specific aspects is defined. It contains constructs in order to support the configuration of strategies and to integrate decision functions. These functions represent in a general way planning specific preferences and heuristics, e.g. to realize action selection or goal ordering decisions. Using all this, different planning strategies are specified to be able to generate plans choosing from a lot of abstractions and control methods. The strategies consider a progressive as well as a regressive refinement process in order to support the non-restrictive combination of different refinement strategies.
The technical realization of the meta language concept is performed using the methods of meta-logical programming, since that perfectly supports the specification of a proof calculus including the required dynamics and adaptation.
According to the flexible plan refinement process proof elements in form of tactics with an interface of kind “plan specification” are specified for the different phases of planning.

As part of that process a configurable proof plan augmented with a definable number of adaptation points is specified as a tactic skeleton. The specification language for the specification of variants of the generic planner is given by a declarative algebra of features. The process of proof strategy configuration is defined by the translation of terms of this algebra into tactics and their integration into a tactic skeleton.
Considering typical adaptation points of the planning process which split it into six correlated phases the relations between adaptation points and the refinement process are analyzed and instances of adaptation features and combinations are specified on basis of the planning strategies which has been modeled. Examples for the concrete realization of control strategy configurations are given.
The planning approach introduced in this thesis has certain benefits w.r.t. other approaches:

- The complete processing chain of planning, starting from the definition and representation of the application domain and desired solutions, the specification of control strategies, their adaptation to a concrete domain, the planning and also the solution extraction mechanism, are all well-defined interrelated processes with precise semantics.
- A general notion of plans is used which allows besides actions also the use of intensional descriptions in order to characterize action behavior.
- The formal model of adaptable control strategies is independent from the concrete logical formalism used for plan representation and generation. It allows the flexible setting of adaptation points.
- Using the planning logic proposed plans can be described by many different control structures and abstractions.
- There is a great variety on structures of planning problems which can be specified and all planning results are provably correct w.r.t. their specification.
- Since plan representation is semantically close to plan generation, the adaptation or change of the two processes can immediately be verified.
- The logical framework provides a natural embedding of the reasoning processes of plan interpretation, plan verification, and plan validation, which are closely related to plan generation. They can be performed with essentially the same strategies and inference rules used by plan generation.

Inhaltsverzeichnis

1 Einleitung **1**
1.1 Motivation . . . 1
1.1.1 Pläne in intelligenten Assistenzsystemen . . . 2
1.1.2 Plankonsumenten und ihre Anforderungen . . . 4
1.1.2.1 Qualitätsmerkmale von Plänen . . . 6
1.1.3 Formallogische Modellierung . . . 7
1.2 Zielsetzung . . . 8
1.3 Gliederung der Arbeit . . . 11

2 Stand der Forschung **13**
2.1 Planen und Logik . . . 13
2.2 Wohlfundiertes Planen . . . 17
2.2.1 Taktisches Beweisen . . . 28
2.2.1.1 Metalogisches Programmieren . . . 29
2.2.2 Algorithmisches Planen . . . 30
2.3 Formale Spezifikationen und logisches Programmieren . . . 33
2.4 Benutzermodellierung . . . 34
2.5 Planqualität . . . 36

3 Eine formale Betrachtung von Planervarianten **41**
3.1 Strukturierung des Konsumenten- und Domänenwissens . . . 41
3.1.1 Plankonsumenten und ihre Modellierung . . . 42
3.1.1.1 Die Rollen eines Planers in einem Assistenzsystem . . . 42
3.1.1.2 Der Benutzer als Plankonsument . . . 48
Annahmen über die Benutzermodellierung . . . 48
3.1.2 Basisaktionen . . . 50
3.1.3 Pläne . . . 51
3.1.4 Entscheidungsfunktionen . . . 52
3.1.4.1 Aktionsspezifische Präferenzen . . . 53
3.1.4.2 Abstraktionsspezifische Präferenzen . . . 55
3.1.4.3 Planspezifische Präferenzen . . . 56
3.1.5 Anwenderspezifisches Domänenwissen . . . 59
3.1.6 Wissen über Mensch-Maschine-Interaktion . . . 59
3.1.7 Kompilation von Aktionswissen . . . 60
3.2 Die Spezifikation von Planungsproblemen . . . 61
3.3 Die Spezifikation von Kontrollstrategien . . . 63

3.3.1 Adaptionspunkte 64
3.3.2 Eine Spezifikationssprache für Kontrollstrategien 67
3.4 Konfigurierbare Strategien 76
3.4.1 Eine erweiterte Spezifikationssprache 76
3.4.1.1 Die Adaptionspunktrealisierung 85
3.4.2 Kontrollstrategien als Metaprogramme 92
3.4.3 Die Planerverfeinerung 95

4 Ein deduktiver Planungsansatz 98
4.1 LLP – Eine Logische Planungssprache 99
4.1.1 Die Syntax 99
4.1.2 Die Semantik 100
4.1.3 Der Kalkül 103
4.1.3.1 Sequenzenkalküle – Das Prinzip 103
4.2 Flexible Planrepräsentation und -generierung 105
4.2.1 Charakterisierung von Aktionen 105
4.2.2 Zulässige Planspezifikationen 110
4.2.3 Pläne in der Logik LLP 111
4.2.4 Planen als *Modellverfeinerung* 116
4.2.5 Aktionsaxiomenschemata 117
4.2.5.1 Die Grundlagen von Regression und Progression 126
Die Berechnung von $\mathcal{FT}$ 126
Die Berechnung von $\mathcal{FT}^{-1}$ 127
Zwei Strategien zum Einsatz von Rahmenaxiomen 128
4.2.5.2 Transformation in Sequenzenregeln 129
4.2.6 Was spricht für einen Sequenzenkalkül? 133
4.2.7 Generierung Abstrakter Pläne 135
4.2.7.1 Sequentielle Pläne 135
Generierung unter Einsatz von Progression 140
Eine effizientere Beweisstrategie 149
Generierung unter Einsatz von Regression 152
4.2.7.2 Nichtdeterministische Pläne 159
Adaptierte Sequenzenregelerzeugung 165
4.2.7.3 Konditionale Pläne 166
Disjunktive Vorbedingungen 168
4.2.7.4 Pläne mit Sicherheitsbedingungen 169
Notwendige Sicherheitsbedingungen 174
4.2.7.5 Nichtlineare Pläne 177
Generierung einfacher nichtlinearer Pläne 180
Generierung beliebiger nichtlinearer Pläne 185
Konflikterkennung und -lösung 189
4.2.7.6 Planen mit konditionalen Effekten 193
4.2.7.7 Planen durch Aufgabenreduktion 195
Indirekte Plandetermination 197
Wiederverwendung von Problemlösungen 199
4.2.8 Planinterpretation 199

4.2.9 Planverifikation und -validierung 204
4.2.10 Zusammenfassung 205

5 Ein konfigurierbarer deduktiver Planer im IAS-Kontext 208
5.1 Kontrollstrategien zur taktischen Beweissteuerung 208
5.2 Realisierung von Entscheidungsfunktionen 212
5.3 Realisierung von Adaptionspunkten 214
5.3.1 Die Merkmalszuordnung 215
5.3.2 Intendierte Merkmalskombinationen 225
5.3.3 Die Umsetzung in Taktiken 228
5.4 Beispiele von Planervarianten 236
5.5 Beispiele im implementierten System 247

6 Abschließende Bemerkungen 267
6.1 Zusammenfassung der Arbeit 267
6.2 Offene Fragen und weiterführende Forschungen 269

A Grammatik verfügbarer Pläne 271

B Verwendete Sequenzenregeln 273

Literatur 277

Abbildungsverzeichnis

1.1 Integriertes Mensch-Maschine-System 1
1.2 Adaptivitätssicht 4
1.3 Plankonsumenten in einem IAS 5
1.4 Klassen zieläquivalenter Pläne 10

2.1 Partielle Taxonomie von Qualitätsmetriken 36

3.1 Grobe Skizze des Planungsansatzes 42
3.2 Beispielhafte Rangfolge von Abstraktionen 55
3.3 Prinzip der Kontrollstrategieadaption 64
3.4 Ein allgemeines Planerschema 66
3.5 Prinzipielle Taktikverarbeitung 75
3.6 Einführung von Adaptionspunkten 77
3.7 Konfigurierbare Taktiken 91
3.8 Zusammenhang der einzelnen Spezifikationen 92
3.9 Drei Sprachebenen einer Kontrollstrategie 93

4.1 Objektstruktur vs. lokale Variable 107
4.2 Hierarchische Objektstruktur 107
4.3 Assoziationslisten als Objekteigenschaften 108
4.4 Beispielabfolge von Verfeinerungsschritten 118
4.5 Einführung einer Basisaktion 120
4.6 Rechtfertigung einer Aktionsregel 121
4.7 Progressive Strategie 128
4.8 Regressive Strategie 129
4.9 Korrespondenz Beweis-, Planstruktur 134
4.10 Prinzipielle Beweiserarchitektur 135
4.11 Skizze einer Progressionsstrategie: Schritt 1 141
4.12 Skizze einer Progressionsstrategie: Schritt 2 142
4.13 Skizze einer Progressionsstrategie: Progressionsschritt 142
4.14 Durchführung einer Progressionsstrategie: Teil 1 144
4.15 Durchführung einer Progressionsstrategie: Teil 2a 145
4.16 Durchführung einer Progressionsstrategie: Teil 2b 147
4.17 Durchführung einer Progressionsstrategie: Teil 2c 148
4.18 Entstehung notwendiger Zwischenziele 149
4.19 *Entschärfung* bereits eingeführter Zwischenziele 151
4.20 Skizze einer Regressionsstrategie: Regressionsschritt 152

4.21 Skizze einer Regressionsstrategie: Erreichen des Anfangszustandes 153
4.22 Durchführung einer Regressionsstrategie: Teil 1 154
4.23 Durchführung einer Regressionsstrategie: Teil 2 155
4.24 Skizze einer Regressionsstrategie: Redundante Aktionen 158
4.25 Vergleich von Progression und Regression 159
4.26 Verfeinerung durch Sicherheitsbedingungseinführung 171
4.27 Kontinuierliche Sicherheitsbedingungseinführung 172
4.28 Einführung einer Ordnungsbeziehung in nebenläufige Verfeinerung 190
4.29 Beweisschritte einer Planinterpretation 203
4.30 Skizzierung einfacher Planverifikationsschritte 206

5.1 Merkmale des Adaptionspunktes AP_{split} 216
5.2 Merkmale des Adaptionspunktes AP_{foc} 217
5.3 Merkmale des Adaptionspunktes AP_{imp} 218
5.4 Merkmale des Adaptionspunktes $AP_{probtrans}$ 220
5.5 Merkmale des Adaptionspunktes AP_{act} 222
5.6 Merkmale des Adaptionspunktes $AP_{plantrans}$ 223
5.7 Adaptionspunktspezifikation Kalenderdomäne 240
5.8 Adaptionspunktspezifikation Roboterdomäne 243
5.9 Skizze eines Roboterbeispielszenarios 245
5.10 Architektur des generischen RAP-Systems 248
5.11 Strukturierter Typ `blocks_world` 249
5.12 Strukturierter Typ `block` 250
5.13 Abstrakte Aktionscharakterisierung 251
5.14 Detaillierte Aktionsspezifikation 252
5.15 Abstrakte Planspezifikationsdefinition 254
5.16 Detaillierte Modellierung einer Planspezifikation 255
5.17 Oberfläche der Planungsumgebung 257
5.18 Visualisierung eines nichtlinearen Planes 259
5.19 Beispiel aus der *Mail*-Domäne 260
5.20 Ausschnitt einer graphischen Robotersimulation 261
5.21 Roboter nach Ausführung von load_device 262
5.22 Modulnetz zur sicheren Planausführung 263

Tabellenverzeichnis

4.1 Eigenschaften von Progression vs. Regression 130

5.1 Zulässige Merkmalskombinationen für AP_{split} 224
5.2 Zulässige Merkmalskombinationen für AP_{foc} 226
5.3 Zulässige Merkmalskombinationen für AP_{imp} 227
5.4 Zulässige Merkmalskombinationen für $AP_{probtrans}$ 229
5.5 Zulässige Merkmalskombinationen für AP_{act} 230
5.6 Zulässige Merkmalskombinationen für $AP_{plantrans}$ 231
5.7 Übersetzung logischer Operatoren 256

Kapitel 1

Einleitung

1.1 Motivation

Der immer kürzer werdende Lebenszyklus von Anwendungssystemen und die sich ständig ändernden Anforderungen und steigenden Ansprüche an ihre Funktionalität führen dazu, daß insbesondere unerfahrene und unregelmäßige Anwender beim Umgang mit diesen Systemen überfordert sind oder das Potential der Systeme nicht effektiv nutzen. *Intelligente Assistenzsysteme* (IAS) können in solchen Fällen Anwender bei der Bedienung des Systems unterstützen und ihr Verständnis, wie das System arbeitet, fördern (vgl. [Boy 91]). Sie spielen die Rolle eines Vermittlers bei der Kommunikation zwischen Mensch und Maschine (vgl. Abbildung 1.1). Es gibt in dem Zusammenspiel Aufgaben, die besser von Men-

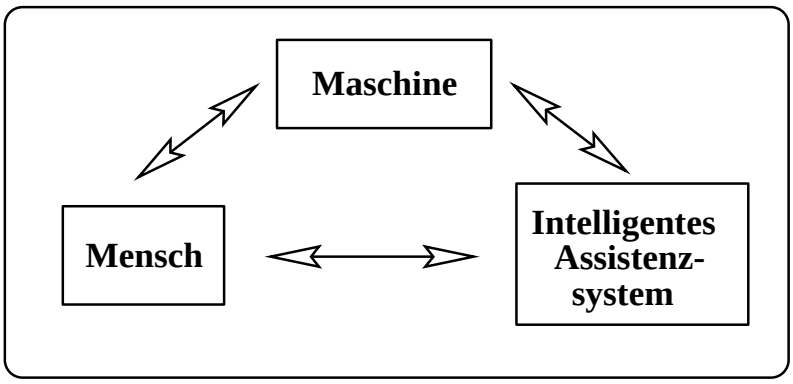


Abbildung 1.1: Integriertes Mensch-Maschine-System

schen ausgeführt werden können und solche, für die eine Maschine besser geeignet ist. Im allgemeinen wird nicht das Ziel verfolgt, eine vollständige Automatisierung zu erreichen. Eine effektive Kooperation kommt zustande, wenn ein Assistent die Pläne des Benutzers erkennen und Fehler oder Inkonsistenzen lokalisieren kann, Pläne für ihn erstellt oder inkrementell erweitert, Details ausarbeitet, die nicht die Aufmerksamkeit des Benutzers erfordern und ihn bei der Optimierung seines Verhaltens unterstützt.[1] Ausprägungen von

[1]Im TRAINS-Projekt [Allen et al. 94, Ferguson et al. 96] wird z.B. ein intelligenter Planungsassistent entwickelt, der in natürlichsprachlicher Interaktion solche Assistenzleistungen erbringt.

Assistenzsystemen entstehen u.a. durch die Bereitstellung von *Hilfe* [Tattersall 92], *Tutoring* [Elsom-Cook 93], *Entscheidungsunterstützung* [Blanning & King 93] und *adaptiven Schnittstellen* [Dieterich et al. 93].
Die Unterstützung von Benutzern interaktiver Softwaresysteme durch intelligente Assistenz bildet den Rahmen für die vorliegende Arbeit.

1.1.1 Pläne in intelligenten Assistenzsystemen

In interaktiven Systemen teilen sich Mensch und Maschine die Ausführung intelligenter Funktionen. Die Interaktion geschieht dabei durch *Kommunikation*, zum einen zwischen dem Benutzer und einzelnen Systemkomponenten, andererseits aber auch zwischen den Systemkomponenten untereinander [Boy 91, Hefley & Murray 93]. *Pläne* und *Planspezifikationen* sind die Kommunikationsobjekte, die in dieser Arbeit im Assistenzsystemkontext schwerpunktmäßig untersucht werden.
Pläne beschreiben allgemein eine strukturierte Menge von Aktionen im Sinne von konkreten oder abstrakten Handlungsanweisungen. Planspezifikationen beschreiben den logischen Zusammenhang zwischen Plänen, ihren zu erreichenden Zielen und den Bedingungen, unter denen sie anwendbar bzw. ausführbar sind.
Dem Benutzer eines interaktiven Anwendungssystems steht in zunehmendem Maße üblicherweise eine graphikbasierte oder natürlichsprachliche Ein-/Ausgabeschnittstelle zur Verfügung. Im folgenden wird davon ausgegangen, daß eine solche Schnittstelle existiert; spezifische Problemstellungen, die sich im Zusammenhang mit einer solchen Schnittstelle ergeben, werden hier nicht weiter betrachtet. Es wird davon ausgegangen, daß entsprechende Transformationen zwischen Plänen/Planspezifikationen und einer benutzergerechten Repräsentationsform existieren, z.B. eine Übersetzung von natürlichsprachlichen Anfragen in formallogische Planspezifikationen.[2]
Auslöser einer Assistenzleistung ist der Benutzer des Anwendungssystems selbst: **explizit**, indem er eine Hilfe- oder Unterstützungsleistung anfordert, oder **implizit**, indem sein Verhalten oder sein Wissensstand zu einer aktiven Unterstützungsleistung des Systems führen.
Der Benutzer kann zielorientierte Fragen und Aufforderungen formulieren, die in irgendeiner Form Aktivitäten bzgl. des *Schließens über Plänen*[3] auslösen. Beispiele sind im folgenden den Anwendungen "*electronic mail*" bzw. "*calendar management*" entnommen:

- Der Wunsch des Benutzers nach der Herstellung eines konkreten Planes zu einem konkreten Problem:
  "Wie kann ich einen Termin für den 30.08.97 eintragen?"

- Der Wunsch des Benutzers nach der Herstellung eines abstrakten Planes zu einem abstrakten Problem:

[2]Neben Aktionsplanungsaspekten spielen in Assistenzsystemen in zunehmendem Maße multimodale Fähigkeiten eine wesentliche Rolle. Dabei treten dann auch Planungsaufgaben wie etwa die Planung natürlichsprachlicher Dialoge oder die Animationsplanung bei der Bereitstellung animierter Hilfestellung zutage (vgl. [Andre 95]).

[3]Damit sind verschiedenste Schlußfolgerungsmethoden angesprochen, deren Gegenstand der Verarbeitung Pläne darstellen.

"Wie kann ich eine existierende Nachricht permanent speichern und danach aus meiner Mailbox löschen?"

- Der Wunsch des Benutzers nach der automatischen Erledigung eines von ihm geschilderten Problems:
  "Ich möchte, daß die Nachricht 5 gespeichert und aus der Mailbox gelöscht wird?"

- Der Wunsch des Benutzers nach Überprüfung eines von ihm erzeugten Planes zur Erreichung bestimmter Ziele:
  "Kann ich mit den Aktionen `type 3` *gefolgt von* `delete 3` *die Nachricht 3 nach dem Lesen löschen?"*

- Der Wunsch des Benutzers nach einer Erklärung für das Fehlschlagen eines Planes:
  "Warum kann ich nach `move 3 myfolder` *nicht* `print 3` *ausführen?"*

- Der Wunsch des Benutzers nach Verbesserung eines von ihm erzeugten Planes zur Erreichung bestimmter Ziele:
  "Kann ich das Lesen und Löschen von Nachrichten mit Absender 'Joe' effizienter ausführen?"

Eine aktive Reaktion des IAS kann z.B. durch folgende Ereignisse hervorgerufen werden:

- Die Beobachtung des Benutzerverhaltens ergab, daß der Benutzer suboptimal gehandelt hat; ihm wird ein Verbesserungsvorschlag unterbreitet.

- Die Beobachtung des Benutzers ergab, daß er mit einer hohen Wahrscheinlichkeit ein bestimmtes Ziel verfolgt; ihm wird vorgeschlagen, die restlichen, notwendigen Schritte automatisch für ihn auszuführen.

- Der Vergleich des aktuellen Wissensstandes des Benutzers mit dem zum Verständnis eines Planvorschlages notwendigen Wissen veranlaßt eine tutorielle Wissensstanderweiterung für den Benutzer.

- Die Überwachung der vom Benutzer vollzogenen Ausführung eines Planes ergibt, daß ein opportunistisch eingeschobener Plan die weitere Ausführung des ursprünglichen Planes behindert oder gar unmöglich machen würde. Ihm wird vorgeschlagen,wie er handeln müßte, um seinen ursprünglichen Plan weiterverfolgen zu können, bzw. es wird ihm erklärt, daß man keine Möglichkeit sieht, daß er für den Fall des Fortschreitens mit seiner aktuellen Aktivität, seinen ursprünglichen Plan zu Ende führen kann.

Adaptives Systemverhalten

Ein wesentliches Ziel von einem IAS ist, die Interaktion zwischen Benutzer und Softwaresystem erfolgreich zu unterstützen. Wenn von der Systemseite Mechanismen bereitstehen, die automatisch alternatives Verhalten selektieren, spricht man von *adaptiven* Systemen ([Benyon 93]).
Die Benutzer interaktiver Softwaresysteme unterscheiden sich hinsichtlich unterschiedlicher Aufgaben, Fähigkeiten, Wissensstände und Präferenzen. Mit der Vorgabe, die Verwendbarkeit eines Systems zu verbessern, sollte eine Unterstützungsleistung aus diesem

Grund angepaßt an den individuellen Benutzer angeboten werden. Das adaptive System braucht als Grundlage eine Beschreibung der Attribute, auf die es sich einstellen kann. Für die Kommunikationsverbindung Benutzer – Software ist dies das Benutzermodell ([Benyon & Murray 93]). Daneben besitzt ein adaptives System ein Domänenmodell, also eine Repräsentation der Anwendungsdomäne, und ein Interaktionsmodell, das alle Aspekte der Beziehung zwischen Benutzer- und Domänenmodell enthält. Verfolgt wird eine anwendungsabhängige Realisierung der Adaptivitätsleistung ([Grunst et al. 91]).[4] Die Anpassung besteht in unserer Betrachtung in situations- und benutzerspezifischer Unterstützung, aber nicht in einer Veränderung der Anwendung, etwa in ihrem Erscheinungsbild oder ihrem Interaktionsverhalten (vgl. Abbildung 1.2).

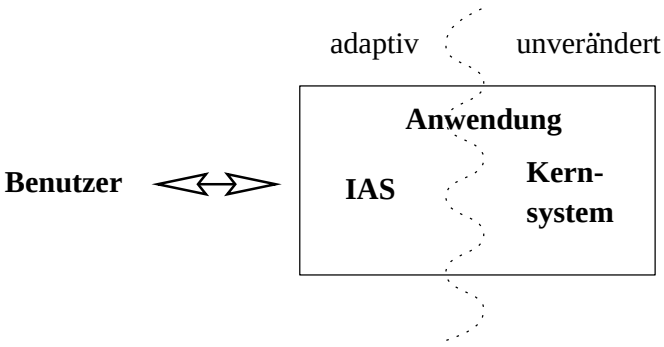


Abbildung 1.2: Adaptivitätssicht

Die mittlerweile zum Standard gewordenen graphischen Benutzungsoberflächen vieler Applikationen sind ein vielschichtiges Anwendungsfeld für adaptive Assistenzleistung. Intelligente, benutzerorientierte Hilfe kann hier sinnvoll nur durch die Integration verschiedener Präsentationsmedien und dynamischer, aufgabenbezogener Hilfestrategien geleistet werden (vgl. [Thies 94]).
Die Adaptivität und insbesondere die Benutzermodellierung sind in einem intelligenten Assistenzsystem nicht Selbstzweck, sondern sie helfen, die Benutzbarkeit eines Softwaresystems in bezug auf Effizienz, Verringerung der Fehlerrate, Erweiterung des Verständnisses der Systemleistungen und Steigerung der Benutzerzufriedenheit zu verbessern.

1.1.2 Plankonsumenten und ihre Anforderungen

Ein intelligentes Assistenzsystem besteht aus mehreren eigenständigen Komponenten, die gemäß einem Interaktionsmodell miteinander verbunden sind. Aus der Sicht des Planungssystems, als der fokusssierten Komponente unserer Betrachtung, können für den Anwendungsbereich "Interaktive Software" folgende Plankonsumenten lokalisiert werden:

1. Der Benutzer des Anwendungssystems.

2. Eine Schlußfolgerungskomponente als Teil des IAS,
   z.B.: ein Planerkenner, ein Planmonitor, eine *Ratgebende*- bzw. *Tutor*-Komponente.

[4] Anwendungsunabhängige Realisierungen würden etwa Leistungen enthalten, die sich rein auf Interaktions- und Präsentationskomponenten beziehen.

3. Eine Aktionsausführungskomponente des Anwendungssystems.

Die Abbildung 1.3 hebt in einer beispielhaften Konstellation den Datenfluß hervor, der einzelne Plankonsumenten mit dem Planungssystem verbindet. Der *Planer*, der aus Plan-

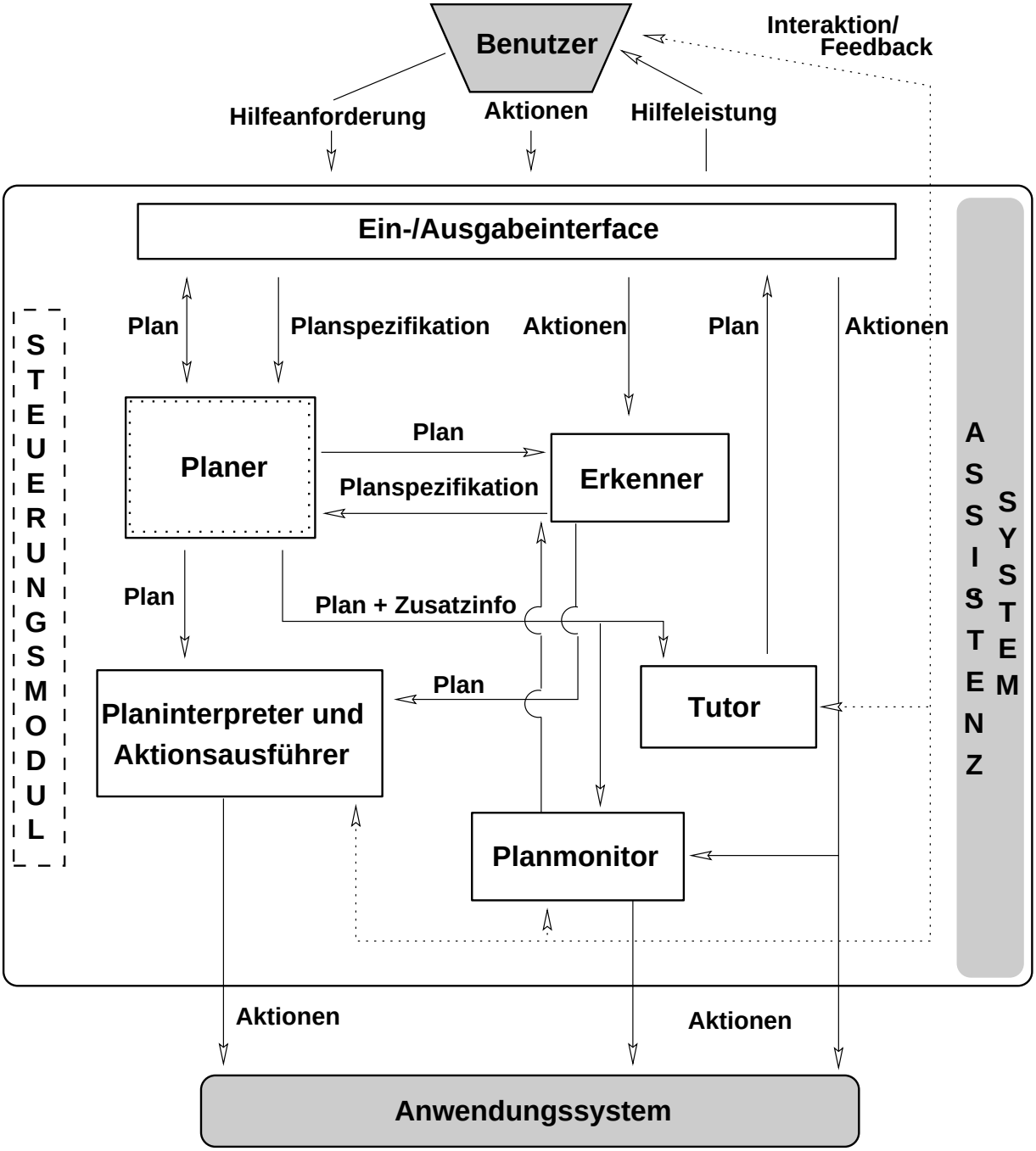


Abbildung 1.3: Plankonsumenten in einem IAS

spezifikationen Pläne erzeugen kann, ist in unserer Betrachtungsweise die wesentliche Komponente des IAS. Mit dem *Benutzer* tritt der Planer direkt auf verschiedene Arten in Beziehung:

- er erzeugt Pläne, die dem Wissensstand des Benutzers genau oder nur partiell entsprechen;
- er arbeitet kooperativ mit ihm zusammen und bezieht ihn falls notwendig in Entscheidungsprozesse mit ein;
- er ist in der Lage, auch planenverwandte Aufgaben wie Planverifikation und -validierung zu lösen.

Der *Planinterpreter* konkretisiert abstrakte Pläne, die von dem *Aktionsausführer* dann im Anwendungssystem automatisch ausgeführt werden können. Der Ausführungszeitpunkt muß dabei nicht unmittelbar dem Zeitpunkt des Planens folgen, was u.a. eine Aufteilung in Generierung und Interpretation rechtfertigt; der Planinterpreter hat aktuell noch einen gewissen Spielraum, konkrete Aktionen zu bestimmen. Der Planerkenner oder einfach *Erkenner* beobachtet und interpretiert das Benutzerverhalten bzgl. einer Menge von Planhypothesen, um Aussagen darüber zu treffen, welches Ziel der Benutzer durch sein Verhalten am ehesten verfolgt. Für die Verarbeitung als Planhypothese kann es nützlich sein, wenn inkrementelle bzw. Pläne mit bestimmtem Abstraktionsgrad erstellt werden. Die *Tutor*-Komponente kann immer dann aktiv werden, wenn benutzerspezifische Pläne nicht allein auf der Grundlage des Benutzerwissens erstellt werden können. Neue Aktionen, Konzepte oder Zusammenhänge werden dem Benutzer entsprechend aufbereitet präsentiert. Die *Planmonitor*-Komponente hat die Aufgabe, die Ausführung eines Planes durch den Benutzer zu überwachen, wobei die Erreichung des Zieles gegenüber der Verwendung bestimmter Aktionen im Vordergrund steht. Erkannte Abweichungen können zur automatischen Generierung von Reparaturplänen führen.

1.1.2.1 Qualitätsmerkmale von Plänen

Pläne können je nach ihrer Form und der aktuellen Anwendungsdomäne entweder als *konkret* oder als *abstrakt* klassifiziert werden.

- Konkrete Pläne simulieren unmittelbar das Aktionsverhalten eines Agenten[5] in der Interaktion mit dem Anwendungssystem. Sie enthalten im einfachsten Fall als Kontrollstruktur die Sequenzialisierung und als Aktionen syntaktisch zulässige Anweisungen an das Anwendungssystem. Dies ergibt sich dadurch, daß der minimale Sprachumfang interaktiver Softwaresysteme aus unstrukturierten Anweisungen besteht, mit der Einschränkung, daß auch nur eine Anweisung zu einem Zeitpunkt verarbeitet werden kann. Die Sequenzialisierung stellt die relative zeitliche Abfolge von Einzelanweisungen dar.
- Abstrakte Pläne hingegen sind eine Repräsentationsform aus der sich unterschiedliche konkrete Pläne herleiten lassen. Sie beschreiben also nur mittelbar das Aktionsverhalten eines Agenten, jedoch mit dem Vorteil, daß eine gesamte Klasse von konkreten Plänen kompakt beschrieben werden kann. Die Beziehung zwischen abstrakten und konkreten Plänen ist durch Interpretationsrelationen ρ für jede Abstraktionsart bestimmt.

[5]Dies kann der Benutzer eines Anwendungssystems sein, oder aber auch eine Systemkomponente.

$$\boxed{\text{Abstrakter Plan}} \Leftrightarrow \rho_n \circ \ldots \circ \rho_1 \Leftrightarrow \boxed{\text{Menge von konkreten Plänen}}$$

Kontrollstrukturen werden verwendet, um die einzelnen Interpretationsrelationen intensional in den Plänen zu repräsentieren.

Die unterschiedlichen Anforderungen von Plankonsumenten in dem betrachteten Szenario an die Planqualität verlangen von dem eingesetzten Planungsmechanismus die Fähigkeit zur Generierung **adaptierter** Pläne. Die einzelnen Dimensionen der Adaptierbarkeit sind in dem folgenden Katalog zusammengestellt:

1. Es gibt Einschränkungen und/oder Präferenzen bzgl. der Aktionen, die in einem Plan auftreten dürfen.
2. Es gibt Einschränkungen und/oder Präferenzen bzgl. der Kontrollstrukturen, die in einem Plan auftreten dürfen.
3. Ein Plan erreicht einen bestimmten Abstraktionsgrad.
4. Ein Plan enthält, angepaßt an den Wissensstand des Konsumenten, ihn unterstützende Aktionen (etwa informationsspendende, sensorische Aktionen).
5. Ein Plan ist unter Verwendung einer bestimmten Problemlösungsstrategie entstanden. Der Einsatz einer bestimmten Strategie kann sich in strukturellen Eigenschaften des generierten Planes widerspiegeln.
6. Ein Plan berücksichtigt in seiner Form und Struktur früheres Handlungsverhalten eines Benutzers.
7. Ein Plan soll mit dem Einsatz möglichst weniger Resourcen zum Ziel führen. Mögliche Resourcen sind etwa: die Planlänge, die Anzahl unterschiedlicher Basisaktionen in einem Plan, Wissensstanderweiterung bzw. -stagnation eines Konsumenten.

Um alle Typen von Konsumenten mit ihren unterschiedlichen Anforderungen effektiv zu bedienen, muß der Planer sowohl flexibel in seiner Steuerung als auch in seiner Ergebnisproduktion sein. Dies bedeutet, daß er in der Lage sein muß, für die jeweilige Nutzung *adaptierte Pläne mit Kontrollstrukturen* zu erzeugen, die das jeweilige intendierte Aktionsverhalten mit der geforderten Abstraktion beschreiben; sie integrieren aus Plankonsumentensicht idealerweise sowohl intensionale als auch extensionale Beschreibungen von Aktionsvorkommen und -zusammenhängen.

1.1.3 Formallogische Modellierung

Für eine vielschichtige Betrachtungsweise von planbezogenen Vorgängen und der Integration heterogener Plankonsumenten in Assistenzsystemen gibt es bisher kein umfassendes Modell. Formallogisches Vorgehen sowohl auf der Repräsentations- als auch auf der Durchführungsebene erlaubt es, komplexe Zusammenhänge exakt zu spezifizieren und eindeutige überprüfbare Beziehungen im Ein-/Ausgabeverhalten komplexer Prozesse aufzubauen und zu beschreiben. Die **deduktive Generierung von adaptierten Plänen mit Kontrollstrukturen** im Kontext eines IAS stellt eine Herausforderung für die Anwendung formalen Vorgehens hinsichtlich seiner Praxistauglichkeit dar. Die Automatisierung des formalen Ansatzes durch *Deduktion* liefert einen Berechnungsmechanismus, durch den die Plangenerierung prinzipiell realisiert werden kann.

1.2 Zielsetzung

Wie bereits zuvor erwähnt, definiert das Anbieten intelligenter Assistenzleistung zur Unterstützung von Benutzern interaktiver Softwaresysteme den globalen Rahmen der vorliegenden Arbeit. Fokussiert wird dabei die Lokalisierung und differenzierte Lösung von Aktionsplanungsaufgaben unter Berücksichtigung unterschiedlicher Anforderungen jeweiliger Plankonsumenten. *Planen* wird als ein Prozeß des *Überlegens* verstanden, dessen Ziel es ist, dem Plankonsumenten zu helfen, sich intelligent in seinem Umfeld zu verhalten. Die geforderte qualitative Variabilität in den zu generierenden Plänen erfordert hohe Flexibilität in der Planrepräsentation und im Planungsansatz. Aus dieser Betrachtung heraus und insbesondere unter Berücksichtigung der beschriebenen Varianten der Planverwendung charakterisiert die Lösung der folgenden Fragestellungen die Ziele der vorliegenden Arbeit.

1. *Ist ein Planbegriff, der Aktionen als Basiseinheiten wählt, flexibel genug, um die Bedürfnisse heterogener Plankonsumenten auf geeignete Weise zu befriedigen?*
   Bei der Lösung dieser Frage wird untersucht, wie man eine stufenlose Flexibilität in der Plandarstellung zur Charakterisierung von Aktionsverhalten, die intensionale und extensionale Aspekte gleichermaßen berücksichtigen soll, erreichen kann. Zu beantworten ist auch, ob die übliche Annahme *notwendiger* Basisbestandteile von Plänen (meist Aktionen) zugunsten einer weniger restriktiven Sichtweise ersetzt werden kann.

2. *Auf welche Art und Weise kann ein adaptierbarer Planungsprozess den Aufgaben im Kontext eines IAS gerecht werden?*
   Wie in dem Abschnitt zuvor beschrieben, ist der Planer als eine eigenständige Komponente eingebunden in ein übergeordnetes Assistenzsystem. Dieses System kann selbst wieder als eigenständige Schlußfolgerungskomponente aufgefaßt werden, deren Verhalten über ein entsprechendes Steuerungsmodul festgelegt ist. Der Planer ist auf der Ebene dieser Strategie nur eine funktionale Komponente in Form einer *Black-Box*. Da der Planer in der Lage sein muß, differenzierte Ergebnisse zu produzieren, muß er folglich *adaptierbar* sein, hinsichtlich der produzierten Ergebnisse aber auch hinsichtlich der verfolgten Lösungsstrategie. Seine Adaptionsanweisungen sind dann elementare Einheiten des Steuerungsmoduls des IAS. Adaptierend einwirken auf den Planer kann nun das Steuerungsmodul (bei einer zentralistischen Struktur) oder jeder einzelne Plankonsument direkt (bei einer objektorientierten Struktur). Im letzteren Fall gehört dazu auch der Planer selbst, d.h. er kann sich *selbstadaptiv* verhalten. Es muß nun ein geeignetes Modell gefunden werden, das die notwendigen Adaptionsprozesse beschreiben kann.

3. *Gelingt es, die vielfältigen IAS-Anforderungen an die Planverwendung und die Planentstehung in einem deduktiven, beweisorientierten Kontext konzeptionell zu realisieren?*
   Bekannte Planungsansätze können unter formalen Gesichtspunkten u.a. hinsichtlich zweier Kategorien unterschieden werden: einem *algorithmischen, datenstrukturorientierten* Vorgehen und einem *formallogischen, beweisorientierten* Vorgehen.

- Der datenstrukturorientierte Ansatz ist kurz folgendermaßen charakterisierbar: Aktionen, Planungsprobleme und Systemzustände sind durch mathematische Strukturen, wie Tupel, Mengen, Ordnungen beschrieben. Für die Problemlösung steht ein spezieller Algorithmus zur Verfügung, der die Strukturen gezielt verändert. Spezielle Funktionen legen charakteristische Eigenschaften der Strukturen fest, z.B., wann eine Problemlösung erreicht wurde. Um die Überprüfung formaler Eigenschaften, wie etwa Korrektheit, zu ermöglichen, muß der spezifischen Verwendung der Datenstrukturen über eine geeignete Abbildung eine Semantik gegeben werden, und die Korrektheit des Algorithmus individuell bewiesen werden.

  Flexibilisierung in der Planrepräsentation erfordert eine stärkere Datendifferenzierung und eine Anpassung bzw. Erweiterung des gesamten Algorithmus und der Semantik.

- Beim beweisorientierten Vorgehen sind Planungsdomäne und Planungsprobleme in einer formalen Logik mit wohldefinierter Semantik axiomatisiert. Die Problemlösung geschieht durch Beweisen einer entsprechenden Formel unter Verwendung eines für die Logik bestehenden Kalküls. Der Plan wird entweder aus dem Beweis extrahiert oder wird als Seiteneffekt während des Beweises mitkonstruiert. Solange man den Rahmen der verfügbaren logischen Sprache noch nicht ausgeschöpft hat, hat eine Flexibilisierung der Planrepräsentation hier im allgemeinen keine weiteren Konsequenzen.

Wohlfundierte Planungsansätze, die dem algorithmischen Vorgehen folgen, sind entweder domänenspezifisch oder die Form der von ihnen produzierbaren Ergebnisse ist fix; außerdem verwenden sie in der Regel ein festes Verarbeitungsprinzip. Durch die relativ *große Distanz* zwischen Algorithmus und Semantik erfordert eine (möglicherweise dynamische) Flexibilisierung der Ergebnisse und Strategien hier im allgemeinen einen zunehmenden Aufwand in der Absicherung der Korrektheit des Vorgehens.

Der zweite Ansatz ist genaugenommen ein Spezialfall des ersten Ansatzes in der Hinsicht, daß die Teilschritte des Planungsalgorithmus hier durch logische Inferenzschritte beschrieben sind und demnach die Semantik des Planungsprozesses unmittelbar gegeben ist. Durch die Verwendung einer geeigneten Planungslogik ist gleichzeitig auch die semantische Fundierung einer flexiblen Planrepräsentation inklusive ihrer Kopplung mit dem Planungsprozeß unmittelbar gegeben. Dies wird im Laufe der Arbeit genauer untersucht werden.

Ausgangspunkt für das Planen nach dem beweisorientierten Ansatz ist eine formale Planspezifikation. Sie beschreibt wie Vorbedingungen und Ziele mit einem Plan in Beziehung stehen. Der Beweis dieser Formel bzgl. der Theorie der axiomatisierten Domäne führt zu einem Plan.

Die möglichen Lösungen für die Plangenerierungsaufgabe bilden eine Planklasse. Im allgemeinen kann eine solche Planklasse unendlich groß sein und alle ihre Repräsentanten erfüllen die Spezifikation, sind also Lösungen für das spezifizierte Planungsproblem (vgl. Abbildung 1.4). Die Pläne sind äquivalent in dem Sinne, daß sie die spezifizierten Ziele bei gleichen Voraussetzungen erreichen (im allgemeinen

auf unterschiedliche Art und Weise; möglicherweise werden mehr Ziele erreicht als notwendig).

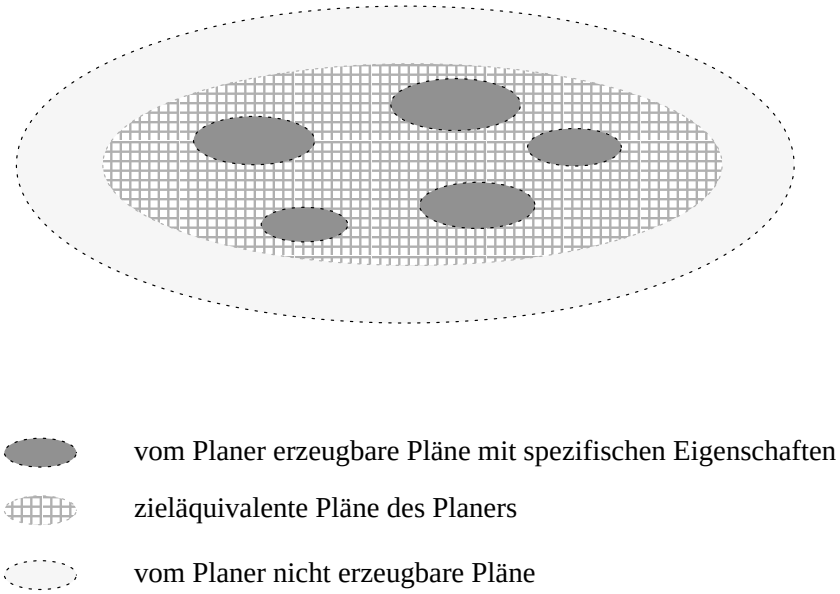


Abbildung 1.4: Klassen zieläquivalenter Pläne

In Abhängigkeit von bestimmten Randbedingungen und bestimmten zukünftigen Plankonsumenten sind jeweils unterschiedliche Repräsentanten aus der Planklasse gefragt. Daraus leitet sich die Anforderung an einen deduktiven Planungsansatz ab, daß der Beweis der gleichen Spezifikationsformel zu unterschiedlichen Plänen führen können muß.

Als Einschränkung wird angenommen, daß die Berücksichtigung unterschiedlicher Wissensstände einzelner Plankonsumenten über die Anwendungsdomäne durch expertenbasierte Modellierung ohne Berücksichtigung fehlerhafter Konzepte geschieht ([Nwana 91, Elsom-Cook 93]).

4. *Wie kann ein deduktiver Planungsansatz maschinell realisiert werden, um effizient adaptierte Pläne zu erzeugen?*
   Es muß untersucht werden, wie allgemeine Planungsstrategien in Beweisstrategien übersetzt werden können, wobei die Forderung nach Adaptivität zugleich erfüllt werden muß. Die Vorteile eines spezialisierten algorithmischen Ansatzes hinsichtlich der Effektivität und Domänenspezialisierung müssen erreicht werden können.

   Grob betrachtet kann man aus Benutzersicht zwei unterschiedliche Beweisansätze zum deduktiven Planen unterscheiden:

   (a) *Planen mit Hilfe "starrer" Beweiser*
   Die Beweiser haben eine wenig flexible Suchstrategie und sind meist Allzweckbeweiser. Aufgrund ihrer relativen Effizienz werden für vollautomatische Beweisführungen am häufigsten Klauselform-verwendende Beweisverfahren basierend auf Resolutions- oder Konnektionskalkülen eingesetzt ([Bibel 92]). Die Vorgehensweise ist sehr maschinennah und die entstehenden Beweise sind kaum

lesbar. Da man als Benutzer wenig Einfluß auf die Vorgehensweise des Beweisers hat, erhält man beim Beweis der gleichen Planspezifikationsformel stets den gleichen Plan.[6] Die Struktur des Beweises entspricht in der Regel nicht der Struktur des resultierenden Planes bzw. der Struktur der zugrundeliegenden Planspezifikation; dies macht auch eine Ansteuerung des Beweisers hinsichtlich der Lieferung einer differenzierten Lösung schwierig.

(b) *Planen mit Hilfe "programmierbarer" Beweiser*
Die Beweiser enthalten eine Metasprache, in der der Benutzer seine eigenen Beweisstrategien spezifizieren kann, er *programmiert* seinen eigenen Beweiser durch die Formulierung von Metaprogrammen. Dies ist eine attraktive Idee, um Wissen in einen Beweiser einzubauen ([Plaisted 90]).
Ob die Beweisprogrammierung eine geeignete Möglichkeit darstellt, bestimmte Repräsentanten einer Planklasse als Lösungen zum spezifizierten Planungsproblem zu erreichen, wird untersucht werden. Es muß auch ein Beweiskalkül gefunden werden, der die adaptierbare Plan- und Beweisentstehung unterstützt.

5. *Können neben den Anforderungen eines planbasierten Assistenzsystems auch die eines klassischen Planungsszenarios wie der Robotersimulation erfüllt werden?*
Der verfolgte Planungsansatz sollte in seiner Anwendbarkeit nicht auf die Domäne interaktiver Softwaresysteme beschränkt sein, sondern so domänenunabhängig und parametrisierbar sein, daß er vielseitig anwendbar ist, z.B. in der Roboterwelt. Hier spielen Sicherheitsbedingungen und integrierte Planungs- und Ausführungsprozesse eine große Rolle.

1.3 Gliederung der Arbeit

Der Schwerpunkt der Arbeit liegt auf der Spezifikation und Umsetzung eines adaptierbaren deduktiven Planungsansatzes, der in der Lage ist, eine Vielzahl von Aktionsplanungsaufgaben im Kontext von Assistenzsystemen zu erfassen und zu lösen.
In Kapitel 2 werden verwandte Gebiete diskutiert. Es werden unterschiedliche formale Planungsansätze präsentiert und ihre Grundlagen beschrieben. Ebenso werden relevante Arbeiten im Umfeld des verfolgten deduktiven Vorgehens eingeführt. Speziell im Umfeld von Assistenzsystemen wichtige Aspekte der Benutzermodellierung und Lösungsqualität sind weitere Gegenstände der Betrachtung.
Aus der Adaptivitätsanforderung an den Planer entspringt die Notwendigkeit ein formales Modell des Planers bzw. des Adaptionsprozesses zu haben. Kapitel 3 führt ein solches Modell ein. Nach der Analyse des relevanten Plankonsumenten- und Domänenwissens werden eine allgemeine Methode zur Kontrollstrategiespezifikation und ein Berechnungsmodell in Form einer Übersetzung in logische Programme beschrieben. Dieses Modell wird anschließend erweitert auf die dynamische Konfigurierung von Kontrollstrategien.
In Kapitel 4 wird detailliert ein auf einer modalen temporalen Planungslogik basierender Planungsansatz in seiner ganzen Vielfalt vorgestellt, der alle in Kapitel 1 beschriebenen Anforderungen an Pläne und Planungsprozesse erfüllt. Es wird hier auch gezeigt, daß die

[6] Der Nachweis der Existenz eines Beweises steht hier im Vordergrund.

spezifizierten Methoden sich eignen, Prozessen wie der Planinterpretation und -verifikation zu einer natürlichen Interpretation im selben Rahmenwerk zu verhelfen.
Kapitel 5 zeigt schließlich, wie die Ergebnisse aus Kapitel 3 und 4 zusammengeführt werden können, um einen konkreten, adaptierbaren deduktiven Planer zu erhalten.
Eine abschließende Zusammenfassung und einen Ausblick auf weiterführende Forschungsarbeiten gibt Kapitel 6.

Kapitel 2

Stand der Forschung

Die vorliegende Arbeit bildet Beziehungen zu einer Vielzahl von Forschungsgebieten, dem *wohlfundierten Planen*, dem *automatischen, insbesondere taktischen Beweisen*, dem *formalen Spezifizieren*, der *Benutzermodellierung* und der *Suchkontrolle*. Es werden hier nun die wesentlichen Verbindungen aufgezeigt, wobei die Diskussion der einzelnen Gebiete nicht vollständig sein kann, jedoch die wesentlichen Ideen werden abgedeckt.

2.1 Planen und Logik

Das in der *Künstlichen Intelligenz* (*KI*) behandelte Planungsproblem, insbesondere das Aktionsplanungsproblem[1], kann folgendermaßen charakterisiert werden. Es gibt eine spezifische *Welt* oder ein System als Anwendungsbereich – auch Anwendungs*domäne* genannt – für das *Planen*. Es wird angenommen, daß diese Welt zu einem bestimmten Zeitpunkt durch einen bestimmten *Zustand* modelliert werden kann. Durch die Ausführung von *Aktionen* in der Welt kann man *Zustandsübergänge* hervorrufen. Aktionen werden oft mit Hilfe von *Operatoren* beschrieben. Das allgemeine Planungsproblem lautet nun:

> Gegeben seien ein *Start*zustand und ein *Ziel*zustand in der Welt. Finde nun eine Folge von Aktionen, die die Welt von dem Startzustand in den Zielzustand überführen.

In der traditionellen Roboter-Planung etwa (vgl. [Fikes & Nilsson 71, Fikes & Nilsson 93]) betrachtet man eine Domäne mit einem Roboter, der vordefinierte Aktionsroutinen sequentiell ausführen kann, um Objekte in einer physikalischen Umgebung zu manipulieren. Dies spielt sich z.B. in der Blockswelt ab. Um ein Ziel zu erreichen, muß der Roboter ein Planungsproblem lösen; er muß eine Sequenz von Aktionen finden, die den aktuellen Zustand der Welt in einen Zustand überführt, der das gegebene Ziel erfüllt. Im allgemeinen ist dies eine Aufgabe, ein Programm zu konstruieren, nämlich eines, das, wenn es in einem Zustand ausgeführt wird, eine Sequenz von Aktionen produziert.
Einführungen in die KI-Planungsproblematik findet man beispielsweise in [Hertzberg 89], [Georgeff 90], [Hendler et al. 90] und [Russell & Norvig 95]. Eine gute Einordnung der prägenden Ansätze des klassischen Planens unter der vereinigenden Sichtweise des *"Refinement planning"* (Planen durch Verfeinerung) wird in [Kambhampati 96] gegeben. Bibel

[1] In Abgrenzung zu etwa geometrischen Pfadplanungsproblemen.

gibt in [Bibel 97] u.a. auch einen Überblick über verschiedene Ansätze zum deduktiven Planen. Darüberhinaus findet man Untersuchungen zu speziellen Themen der Planungsproblematik in den Veröffentlichungen großer KI-Konferenzen, etwa die der *International Joint Conference on Artificial Intelligence* [IJC 97] oder der *European Joint Conference on Artificial Intelligence* [Wahlster 96] jeweils in eigenen Sektionen, und seit einigen Jahren auch in Veröffentlichungen spezieller Planungskonferenzen, die der *Artificial Intelligence Planning Systems* [Drabble 96] oder die der *European Conference on Planning* [Steel 97].

Planungsaspekte, die in dieser Arbeit angesprochen werden, sind gewachsen aus einer formallogischen Betrachtung des Planens. Aus diesem Grund wird im Laufe diese Kapitels ein kurzer Überblick über existierende logische Formalisierungen des Planens gegeben.

Forderungen, die im Zusammenhang mit einem *Planer* – der Mechanismus oder Algorithmus, der das Planen durchführt – als offensichtlich notwendig erscheinen können, sind die nach *Korrektheit* und *Vollständigkeit* eines Planers.
Ein *korrekter* Planer liefert niemals einen inkorrekten Plan zurück; dies gilt natürlich nur bzgl. der Festlegung eines bestimmten Korrektheitsbegriffes. Er findet jedoch nicht immer eine Lösung, selbst wenn eine existiert. Ein *vollständiger* Planer hingegen liefert immer eine korrekte Lösung, wenn eine existiert.
Die beiden genannten Eigenschaften eines Planers kann man jedoch nur zeigen, wenn der Planer auf einer formalen Grundlage aufgebaut ist. Eine ausgezeichnete Grundlage bilden logische Formalisierungen des Planens. In der Literatur werden viele logische Formalismen zum Planen vorgeschlagen, aber spezifizierte oder implementierte Planer sind daraus nur wenige hervorgegangen.
Ein Ansatz mit langer Tradition, der auch noch heute großen Einfluß auf das Forschungsgebiet Planen hat, ist der *Situationskalkül*-Formalismus (vgl. [McCarthy & Hayes 69]). Der Situationskalkül ist eine Methode zum Ausdruck von Fakten über Aktionen und Zustandsveränderungen in mathematischer Logik (Prädikatenlogik 1. Ordnung); man spricht in diesem Zusammenhang auch von einer *Situationslogik*. Es gibt dabei Ausdrücke für Situationen und Aktionen (Terme in der Logik) und eine Funktion *"result"*, die diese zueinander in Beziehung setzt. *Zeit* ist in dem Originalansatz nur implizit repräsentiert, in dem Sinne, daß Zustände geordnet sind. In dem folgenden Beispiel aus einer Blockswelt-domäne,

$$s = result(Move(Block1, Top(Block2)), result(Move(Block2, Table), S_0))$$

wird die Startsituation S_0 durch die Ausführung von zwei Aktionen $Move$ in die Situation s übergeführt. Die Situation s wird also durch eine Aktionskomposition ausgehend von einer Startsituation konstruiert. Die Zustandsübergangsfunktion $result(A, Z)$ benennt die Situation, die durch die Ausführung von Aktion A in Situation Z entsteht. Jede Relation bzw. Funktion, die ihren Wert bei der Ausführung einer Aktion verändern kann, erhält ein zusätzliches Situationsargument. Dadurch wird es möglich, Schlußfolgerungen über die Werte von Relationen bzw. Funktionen in unterschiedlichen Zeitpunkten zu ziehen. Es ist nun möglich, Axiome zu formulieren, die die durch eine Aktion bewirkten Veränderungen beschreiben. Die Axiome haben die folgende schematische Form:

$$\forall X, s\ [pre(X, s) \wedge cond(X, s) \rightarrow \mathit{eff}(X, result(a(X), s))].^2$$

[2] X sei ein Vektor von Variablen.

$pre(X, s)$ beschreibt die Vorbedingungen, die in Zustand s gelten. Die Formel $eff(X, result(a(X), s))$ beschreibt eine Bedingung, die nach Ausführung von Aktion a in dem Zustand $result(a(X), s)$ gilt; $cond(X, s)$ beschreibt Randbedingungen, unter denen die Zustandsänderung vorgenommen wird, z.B. Ungleichungen zwischen bestimmten Variablen.
Für jede Aktion a kann man auch Axiome spezifizieren, die die Eigenschaften der Situation s zu den Eigenschaften von $result(a, s)$ in Beziehung setzen; man beschreibt damit die Eigenschaften, die von der Aktion nicht verändert werden. Die allgemeine Form dieser Axiome ist:

$$\forall X, s\ [pre(X, s) \wedge cond(X, s) \wedge persist(X, s) \rightarrow persist(X, result(a(X), s))].$$

Das Prädikat *persist* beschreibt dabei die Bedingung, die von der Aktion a nicht betroffen ist. Ein Beispiel aus der Blockswelt wäre etwa:

$$on(x, y, t) \wedge x \neq c \rightarrow on(x, y, result(Move(c, d), t))$$

d.h. die Eigenschaft, daß x in der Situation t auf y steht wird von der Ausführung der Aktion $Move(c, d)$ – der Bewegung von c nach d – in der Situation t nicht berührt, wenn x ungleich c ist.
Die *konstruktive* Eigenschaft des Situationskalküls macht ihn als Planungsformalismus verwendbar. Um einen Plan für ein bestimmtes Ziel zu konstruieren, beweist man, daß eine Situation existiert, in der das Ziel gilt. Während des Beweises wird die Situation durch Aktionskomposition konstruiert und die Aktionssequenz, also der Plan, kann aus dem Beweis extrahiert werden.
Gelfond et al. beschreiben in [Gelfond et al. 91] die Möglichkeiten und Beschränkungen des Situationskalkül-Ansatzes. Aktuelle Erweiterungen des Situationskalküls beziehen Konstrukte für *Ereignisse* [Miller & Shanahan 94] und Zeit [Pinto 94] explizit mit ein, um Wissen über die reale Welt geeignet darstellen zu können.

Ein weiterer Situationslogikansatz, dem insbesondere durch seine Verwendung in aktuellen Planungssystemen [McDermott 91, Penberthy & Weld 92] Bedeutung zukommt, ist ADL (*Action Description Language*) [Pednault 86, Pednault 88]. ADL kann als Umformulierung des Situationskalküls in sogenannte *Aktionsschemata*, die dem klassischen STRIPS-Formalismus [Fikes & Nilsson 71] verwandt sind, angesehen werden. Die Schemata charakterisieren eine Aktion in vier unterschiedlichen Kategorien: Vorbedingungen, *Add-* bzw. *Delete*-Mengen von Relationen und schließlich eine Menge von Funktionsveränderungen. ADL enthält nicht die volle Ausdrucksmächtigkeit des Situationskalküls[3], kann damit allerdings auch mit einem besseren Berechnungsverhalten aufwarten.

Eine Vielzahl von Logiken, die für das Planen Verwendung finden, sind *explizite* Zeitlogiken, entweder auf Zeitpunkten oder Zeitintervallen basierend, d.h. Symbole zum Ausdruck der Zeit kommen als Terme in diesen Logiken vor.
Bekannte Ansätze sind:

- Die punktbasierte Logik von McDermott [McDermott 82] basierend auf mehrsortiger Prädikatenlogik 1. Ordnung: seine zugrundeliegende Zeitrelation stellt sicher,

[3]Die Formeln zur Beschreibung der Effekte von Aktionen sind syntaktisch eingeschränkt.

daß die vollständige Menge von Zuständen eine baumartige Struktur aufweißt, d.h. er modelliert eine in die Zukunft verzweigende Zeit, die die Unbestimmtheit der Zukunft beschreibt.

- Die intervallbasierte Logik von Allen basierend auf mehrsortiger Prädikatenlogik 1. Ordnung: auf einer zugrundeliegenden Intervallalgebra [Allen 83] wird eine Zeitlogik definiert [Allen 84]. Später wird ein kompakteres Modell vorgestellt, das nur noch *eine* primitive Zeitrelation betrachtet [Allen & Hayes 89]. Zeitliches Schließen und Planen mit dieser Logik ist in [Allen 91, Allen et al. 91] beschrieben. In [Allen et al. 91] ist auch ein nichtdeterministischer Planungsalgorithmus angegeben, der jedoch für die Praxis eher irrelevant ist.
  In aktuellen Arbeiten [Allen & Ferguson 94, Ferguson 95] bildet Allen's Logik die Grundlage für die allgemeine Repräsentation von Ereignissen und Aktionen, um eine Vielzahl von Schlußfolgerungsaufgaben zu unterstützen. Anwendung findet der Formalismus in TRAINS [Allen et al. 94, Ferguson et al. 96], einem Projekt zum Aufbau eines "sprachlich erfahrenen Planungsassistenten".

- In [Shoham 87a] werden die reifizierten[4] Ansätze von McDermott und Allen kritisiert und eine alternative zeitpunktbasierte Logik mit klarer Semantik wird angegeben, die mindestens so ausdrucksstark ist wie die Logiken der kritisierten Ansätze.

- Bacchus et al. [Bacchus et al. 89] geben eine nichtreifizierende Zeitlogik mit expliziten Zeitargumenten an, die Shoham's Ansatz subsummiert. Es werden dabei keine ontologischen Verpflichtungen, ob die zeitlichen Objekte nun Intervalle oder Punkte sind, gemacht.

- Eine explizite Zeitlogik, die speziell für die Anwendung im nichtlinearen Planen[5] formuliert wurde, wird in [Zhang & Barringer 94b, Zhang 94] beschrieben. Sie ist eine punktbasierte Intervallogik mit der Besonderheit, daß Zeit semantisch durch eine partiell geordnete Struktur modelliert wird. Im Gegensatz dazu benutzt Allen etwa ein lineares Zeitmodell und McDermott eine in die Zukunft verzweigende Baumstruktur als Zeitmodell.

In Erweiterung der rein prädikatenlogischen Ansätze zur Planung sind auch einige modallogische Ansätze von Bedeutung.

- Die frühen *Dynamische Logik*[6]-Ansätze von Rosenschein [Rosenschein 81] und Kautz [Kautz 82].

- Stephan und Biundo [Stephan & Biundo 93] stellen ebenfalls einen Ansatz der Dynamischen Logik zum Planen vor. Basisaktionen werden dabei aus zwei elementaren Aktionen zusammengesetzt, ähnlich einem Vorgehen, das in [Pednault 87][7] beschrieben ist.

[4] Propositionen werden verdinglicht (reifiziert), d.h. eine Proposition *Farbe(Haus,rot)* wird zu einem Term der logischen Sprache gemacht.

[5] Die Menge der Aktionen, die sich zu einem Plan zusammensetzten, sind hier nur partiell geordnet.

[6] Siehe [Harel 79].

[7] Jedoch nicht für dynamische Logik.

- In [Shu 94] wird ein multimodaler Ansatz zur Planung in dynamischen Systemen gewählt. Verwendet wird eine propositionale dynamische Logik, wobei es Modaloperatoren für Zeit, Aktionen, Glaube, Ziele und Pläne zusammengefaßt in einer Sprache gibt.
- Ein modallogischer Ansatz, der Konzepte von Programmlogiken [Kröger 87] und temporalen Intervallogiken [Halpern et al. 83, Rosner & Pnueli 86] verbindet, ist in [Biundo et al. 92a, Biundo et al. 92b] beschrieben (vgl. auch [Dengler 96a] zu Planungsmechanismen in diesem Ansatz). Pläne sind hier gleichberechtigte Formeln der zugrundeliegenden Logik, im Gegensatz etwa zu Ansätzen der dynamischen Logik, wo Pläne (Programme) ausgezeichnete Objekte der Logik sind.

Das *Frame-Problem*[8] (engl.: *frame problem*) [McCarthy & Hayes 69] ist eines der klassischen Probleme beim Schließen über Aktionen. Das Problem besteht darin, zu spezifizieren, welcher Teil der Welt sich nicht ändert, wenn eine Aktion ausgeführt wird, d.h. die Aktionsinvarianten der Planungsdomäne zu beschreiben. Im Beispiel der Blockswelt bedeutet dies z.B. zu spezifizieren, daß eine Aktion $Move(Block, Position)$ zur Bewegung eines Blockes auf eine neue Position etwa weder die Farbe noch die Größe des Blockes verändert.
Es gibt zahlreiche Veröffentlichungen und Ansätze zu diesem Thema (vgl. z.B. [Shoham 87b, Hayes 90, Schubert 90], [Brown 87] enthält eine Sammlung von Beiträgen), so daß auf eine Übersicht der kompletten Problematik und der verschiedenen Lösungsansätze hier verzichtet wird; eine aktuelle detaillierte Übersicht zu diesem Thema ist in [Shanahan 97] erschienen. Instanzen des Problems werden im folgenden nur angesprochen, wo sie gerade von Bedeutung sind.
Im Zusammenhang mit dem Frame-Problem stehen auch das *Qualifikationsproblem* (engl.: *qualification problem*) [Ginsberg & Smith 87] – d.h. alle nur erdenklichen Vorbedingungen zu spezifizieren, die gelten müssen, damit eine Aktion wirklich ausführbar ist – und das *Ramifikationsproblem* (engl.: *ramification problem*) [Ginsberg & Smith 88] – d.h. all das herauszufinden, was durch eine Aktion tatsächlich verändert wurde; neben den unmittelbaren Effekten einer Aktion gibt es im allgemeinen auch noch davon ableitbare Effekte.

2.2 Wohlfundiertes Planen

In der Forschungsgeschichte im Bereich des Planens kann man eine Zweiteilung der verfolgten Ansätze vornehmen. Zum einen gibt es aus der *praktischen Anwendung* motivierte Formalismen (vgl. z.B. STRIPS [Fikes & Nilsson 71], NOAH [Sacerdoti 75],
SIPE [Wilkins 88], FORBIN [Dean et al. 90], O-PLAN [Currie & Tate 91]), die eine effektive Verwendbarkeit von Planern in unterschiedlichen Anwendungsfeldern wie etwa Robotik und Logistik unterstützen.
Wohlfundierte Ansätze zum Planen hingegen zeichnen sich vorallem dadurch aus, daß man bestimmte ihrer Eigenschaften *beweisen* kann, so z.B. die Korrektheit ihrer Ergebnisse bzgl. der Problemspezifikation. Neben eher *algorithmisch datenstrukturorientierten* Ansätzen wie SNLP [McAllester & Rosenblitt 91] und den populären Systemen

[8] Im Deutschen auch oft als *Rahmenproblem* bezeichnet.

UCPOP [Penberthy & Weld 92] und PRODIGY [Fink & Veloso 95] gibt es eine Reihe von formallogischen Ansätzen, die *deduktives Beweisen* als Planungsmethode verwenden; man spricht dann von *deduktivem Planen*. Das Planungsproblem sowie die Aktionen sind dabei durch logische Formeln repräsentiert. Die Lösung eines Planungsproblems geschieht durch die Führung eines konstruktiven Beweises mithilfe eines automatischen oder interaktiven *Theorembeweisers*. Dabei wird bewiesen, daß die Formel, die die Planungsziele beschreibt, eine logische Konsequenz der formulierten initialen Planungsvoraussetzungen und Aktionen ist. Der Plan entsteht dabei als *Seiteneffekt* des Beweisvorganges.
Biundo [Biundo 94] gibt einen Überblick über aktuelle Trends im deduktiven Planen. Im folgenden erscheinen detailliertere Beschreibungen bzw. Ergänzungen der dort genannten Ansätze.

Situationskalkülansätze

Eine fundamentale Arbeit ist der Planer QA3 von Green [Green 69], der als erster eine Implementation des Situationskalküls [McCarthy & Hayes 69] vornahm. Er verwendet einen Resolutionsbeweiser, der Paramodulation für den Umgang mit Gleichungen verwendet.[9] Die Ausdrucksmächtigkeit des Situationskalküls, die notwendige Verwendung einer Vielzahl von Frame-Axiomen und schließlich die Anwendung eines allgemeinen Beweisverfahrens widersprachen einer ernsthaften Verwendung seines Ansatzes.
Kowalski [Kowalski 79] führt eine metalogische Variante des Situationskalküls ein, die sich durch die Verwendung eines HOLDS-Prädikates von der bei Green verwendeten unterscheidet. Die Formel $\mathsf{HOLDS}(ontable(b), s_0)$ ist z.B. dann wahr, wenn die Eigenschaft $ontable(b)$ in Situation s_0 wahr ist. Somit ist es erlaubt, Eigenschaften – also Prädikate – als logische Objekte zu verwenden. Ein Vorteil der Methode ist die Reduzierung der Zahl notwendiger Frame-Axiome, indem für jeden Operator nur ein solches Axiom notwendig wird. Im Gegensatz zu Green, wo für jeden Operator soviele Frame-Axiome notwendig sind, wie Eigenschaften vorhanden sind. Ein Frame-Axiom lautet dann etwa:

$$\mathsf{HOLDS}(f, s) \wedge f \neq clear(y) \rightarrow \mathsf{HOLDS}(f, result(Move(x, y, s)))$$

Es sagt aus, daß abgesehen von der Eigenschaft, daß Position y frei ist, alle Eigenschaften erhalten bleiben, wenn Objekt x in Zustand s mit Hilfe der $Move$-Aktion auf Position y bewegt wird.

Die Plantheorie von Manna und Waldinger

Manna und Waldinger machen einen der ersten Versuche, automatisch rekursive Pläne zu erzeugen [Manna & Waldinger 86, Manna & Waldinger 87]. Sie formalisieren das Problem "einen Block frei zu machen" in ihrer *Plantheorie*, einer Logik erster Ordnung basierend auf dem Situationskalkül, und beschreiben, wie mit ihrem deduktiven Tableau-Beweiser ein rekursives Programm synthetisiert werden kann, das das spezifizierte Problem löst.
Sie sehen eine starke Analogie zwischen Plänen und Programmen:

> *"Plans are closely analogous to imperative programs in that actions may be regarded as computer instructions, tests as conditional branches and the world*

[9] Ausgezeichnete Übersichtsvorträge über Resolutionsbeweiser und das Schließen über Gleichungen finden sich in [Blaesius & Buerckert 92, Gabbay et al. 93].

> *as a huge data structure. This analogy suggests that techniques for the synthesis of imperative programs may carry over into the planning domain. Conversely, we may anticipate that insights we develop by looking at a relatively simple planning domain, such as the blocks world, would then carry over to program synthesis in a more complex domain, involving array assignments, destructive list operations, and other alterations of data structures."*
>
> [Manna & Waldinger 86], Seite 622

In der Plantheorie gibt es zwei Klassen von Termen: *statische* Terme, die zustandsunabhängig die gleiche Entität repräsentieren und sogenannte *fluent*-Terme, die Entitäten repräsentieren, deren Bedeutung sich von Zustand zu Zustand verändern kann. Durch statische Terme bezeichnet man etwa Zustände, Objekte oder Wahrheitswerte. Fluents bezeichnen hingegen bestimmte Eigenschaften oder Merkmale von Objekten, z.B. der Term $auf(d)$ steht für das Objekt, das auf dem Objekt d steht. Die Abbildung von Fluents auf statische Objekte wird durch *Verbindungsoperatoren (linkage operators), :, :: und ;,* formalisiert. Der statische Ausdruck $s : auf(d)$ bezeichnet das konkrete Objekt, das auf d in dem Zustand s steht.[10]
Pläne werden nun repräsentiert als Programme in einer LISP-ähnlichen Sprache[11]. Sie werden betrachtet als Fluents, die mit unterschiedlichen Zuständen verbunden werden können. Der Plan, um einen Block frei zu machen, sieht in der Plantheorie wie folgt aus:

$$makeclear(a) \Leftarrow \begin{cases} if & clear(a) \\ then & empty\_plan \\ else & makeclear(auf(a)); put(auf(a), tisch) \end{cases}$$

Die Generierung von Plänen bzw. Programmen geschieht nun durch Beweisen eines bestimmten Theorems; für das Problem des "Blockfreimachens" ist dies:

$$\forall s_0 \forall a \exists z_1 [Clear(s_0; z_1, a)]$$

Es bedeutet, daß für alle initialen Zustände s_0 und für alle Objekte a ein Plan z_1 existiert, so daß das Objekt a frei ist, nachdem der gewünschte Plan z_1 im initialen Zustand ausgeführt wurde. Ein Programm, das den gewünschten Plan produziert, wird aus dem Beweis des Theorems extrahiert. Dieses Programm wird oft auch direkt als Plan bezeichnet, obwohl es eine Funktion berechnet, die nur dann einen Plan produziert, wenn sie auf ein Argument angewendet wird. Manna und Waldinger schlagen vor, daß ein allgemeiner Theorembeweiser mit einer strategischen Komponente für planungsspezifische Beweisaufgaben ausgerüstet werden sollte, um mit anderen nicht beweiserorientierten Problemlösern konkurrieren zu können. Eine solche Komponente fehlt jedoch in ihrem implementierten Beweiser.

Rosenscheins und Kautz' "Dynamische Logik"-Ansätze

Rosenschein und Kautz [Rosenschein 81, Kautz 82] waren die Ersten, die Dynamische Logik [Harel 79] für das Planen verwendeten. Rosenschein verwendet propositionale dynamische Logik ohne Schleifen. Dynamische Logik wurde hauptsächlich für das Schließen

[10] Durch "::" erhält man Wahrheitswerte und durch ";" Zustände.
[11] Die Programmiersprache LISP ist grundlegend in [Winston & Horn 88] beschrieben.

über Programmen verwendet, aber Rosenschein sieht auch die enge Verbindung zwischen Programmen und Plänen und dadurch scheint ihm die Verwendung dynamischer Logik zur Beschreibung von Planungsproblemen adäquat. Eine typische Modalformel in der dynamischen Logik ist z.B. $[\alpha]p$, das gelesen wird als "nach der Ausführung von α gilt p". Der Plan α ist dabei die *Nullaktion* (Λ), eine atomare Aktion (a), die Komposition zweier Pläne $(\beta;\gamma)$, oder ein konditionaler Plan $(t \rightarrow \beta, \gamma)$. Die dynamische Logik macht eine explizite Unterscheidung in ihrer Signatur zwischen Objekten, die Programme beschreiben, und Objekten, die Eigenschaften von Zuständen ausdrücken.
Ein Planungsproblem wird definiert durch die Angabe von:

1. *Domänenrandbedingungen*
   Neben nichtmodalen Formeln sind Axiomatisierungen von atomaren Aktionen a durch die Angabe von Vorbedingungen und Effekten in der Form $pre \rightarrow [a]goals$ enthalten (pre und $goal$ sind nichtmodale Formeln).

2. *Planrandbedingungen*
   Sie sind von der Form $p \rightarrow [u]q$ für ein einziges ausgezeichnetes Aktionssymbol u, zu dem keine Domänenrandbedingungen spezifiziert sind; p und q bezeichnen modalfreie Formeln und können als Ausgangsbedingungen bzw. zu erreichende Ziele des zu findenden Planes angesehen werden.

Eine Lösung für das Planungsproblem besteht darin, eine Substitution für die Aktion u in Form eines Planes α zu finden, so daß von den Domänenrandbedingungen ableitbar ist, daß α alle Planrandbedingungen erfüllt.
Die Suche nach einem solchen Plan wird entweder durch Progression, Berechnung der stärksten Nachbedingungen, oder durch Regression, Berechnung der schwächsten Vorbedingungen, gesteuert. Es werden nur solche Pläne als Lösungen akzeptiert, die die Spezifikationen beweisbar erfüllen.
Kautz erweitert diesen Ansatz auf dynamische Logik erster Ordnung ohne Funktionssymbole; er erhält dadurch u.a. parametrisierbare Aktionen. Er verwendet einen beweisbar korrekten Progressions- bzw. Regressionsalgorithmus zur Suche nach der Lösung eines Planungsproblems.

Der "Dynamische Logik"-Ansatz von Stephan und Biundo

Pläne als Programme zu betrachten, zeichnet auch den Ansatz von Stephan und Biundo (vgl. [Stephan & Biundo 93]) aus. Sie schlagen ein logisches Rahmenwerk zur Spezifikation konsistenter Axiomatisierungen von Domänen in dynamischer Logik vor. Auf Zustände wird hier mittels Modaloperatoren verwiesen. Zustände werden durch Mengen von Grundliteralen charakterisiert und Aktionen bzw. Pläne sind Programme, die diese Mengen durch Hinzufügen bzw. Löschen von Literalen verändern. Es gibt dementsprechend zwei wesentliche Basisaktionen *add* und *delete*, die jeweils ein Literal zu einem Zustand hinzufügen bzw. aus ihm entfernen, und dies sind auch die einzigen Anweisungen, um eine Zustandsänderung hervorzurufen. Die Aktion zum Herunterstellen eines Blockes ist z.B.

beschrieben durch eine einfache Abstraktion $\gamma_{unstack}(x, y)$[12]:

$$\begin{array}{lll} \textbf{rec } unstack(x,y). & \textbf{if} & on(x,y) \wedge clear(x) \\ & \textbf{then} & \textbf{add} - table(x) \\ & & \textbf{add} - clear(y) \\ & & \textbf{delete} - on(x,y) \\ & \textbf{else} & \textbf{abort fi} \end{array}$$

Der Ansatz erfordert, daß für alle in der Domäne relevanten benutzerdefinierten Relationen jeweils *add-* bzw. *delete*-Operationen definiert sind. Die wesentlichen Aktionen als Elemente der Plansprache sind:

$$\begin{array}{rcl} \pi & ::= & \textbf{skip}, \mid \textbf{abort} \mid \textbf{delete-r}(\tau_1, \ldots, \tau_n) \mid \textbf{add-r}(\tau_1, \ldots, \tau_n) \mid \\ & & (\pi_1; \pi_2) \mid \textbf{if } \varphi \textbf{ then } \pi_1 \textbf{ else } \pi_2 \textbf{ fi} \mid (\pi_1 \textbf{ or } \pi_2) \mid \\ & & \textbf{choose } x \textbf{ begin } \pi \textbf{ end} \mid \gamma(\tau_1, \ldots, \tau_n) \end{array}$$

$\gamma(\tau_1, \ldots, \tau_n)$ ist dabei eine Aktionsabstraktion, etwa eine *unstack*-Aktion wie oben beschrieben.
Um über Veränderungen von Zuständen, die durch **add-** bzw. **delete**-Operationen hervorgerufen werden, schließen zu können, benötigt man Formeln der Art $[\pi]\varphi$ mit der Bedeutung "wenn das Programm π terminiert, dann gilt hinterher φ"; der duale Operator $\langle\pi\rangle$ bedeutet "π terminiert mit der Bedingung φ". Durch die Angabe von domänenspezifischen Randbedingungen kann der Zustandsraum zusätzlich eingeschränkt werden. Ein Beispiel einer solchen Randbedingung ist etwa $\forall x\ (clear(x) \equiv \neg\exists y\ on(y, x))$. Stephan und Biundo behandeln solche Randbedingungen in ihrem Ansatz als Invarianten bzgl. der definierten Aktionen. Man kann etwa beweisen, daß gilt:

$$(\forall x\ (clear(x) \equiv \neg\exists y\ on(y, x))) \rightarrow [\gamma_{unstack}(x, y)](\forall x\ (clear(x) \equiv \neg\exists y\ on(y, x)))$$

Wenn man zeigt, daß dies für alle anderen Basisaktionen auch gilt, sichert man sich eine konsistente Beschreibung der Planungsdomäne. Rahmenaxiome werden in dem Ansatz nach Bedarf generiert und zwar nach einem einheitlichen Schema. Die elementaren **add-** bzw. **delete**-Operationen in den Basisaktionen bestimmen dabei die Form der jeweiligen Rahmenaxiome.
Planen mit diesem Ansatz bedeutet, Formeln bestimmter Art zu beweisen. Wenn Γ bzw. Δ Mengen von Formeln sind, die den Anfangszustand bzw. den Zielzustand eines konkreten Planungsproblems beschreiben, so ist eine Planspezifikation gegeben durch die Formel $\bigwedge\Gamma \rightarrow \langle ?a\rangle \bigwedge\Delta$.[13] Hier steht $?a$ für eine Metavariable, die in einem zielgerichteten Beweis durch einen Plan instantiiert wird. Solch eine Vorgehensweise kann in einem vorhandenen taktischen Theorembeweiser [Heisel et al. 90] basierend auf dynamischer Logik implementiert werden.

Der "Temporallogik"-Ansatz von Stephan und Biundo

Aufbauend auf den in ihrem "Dynamische Logik"-Ansatz (vgl. [Stephan & Biundo 93]) eingeführten Prinzipien der elementaren *add-* und *delete*-Aktionen und dem Konzept,

[12] Der Formalismus erlaubt auch die Definition einfach rekursiver Aktionen; **rec** $a(x_1, \ldots, x_n).\pi$ ist eine rekursive Aktion mit formalen Parametern $x_1, \ldots, x_n$ und π ist der Rumpf der Aktion.

[13] $\bigwedge\Gamma$ repräsentiert die Konjunktion der in Γ zusammengefaßten Elemente.

Pläne als Programme anzusehen, betten sie dieses Vorgehen in eine modale Temporallogik ein (vgl. [Stephan & Biundo 96]). Sie überwinden damit die Einschränkung, daß Pläne und Spezifikationen syntaktisch streng getrennt sind und ermöglichen damit die Formulierung aussagekräftigerer Spezifikationen, die nun auch Aussagen über Zwischenzustände und nicht nur Anfangs- und Endzustände ermöglichen. Die verwendete Modallogik basiert semantisch auf *Intervallen von Zuständen*. Ein Zustandsübergang wird dabei durch genau eine *add*- bzw. *delete*-Aktion hervorgerufen. Da die Basisaktionen einer Planungsdomäne als prozedurale Abstraktionen beschrieben werden, bedeutet dies, daß eine Basisaktion im allgemeinen mehr als einen Zustandsübergang umfaßt.
Ausgangspunkt für die Planerstellung ist eine initiale Spezifikation vom Aufbau

$$FIN \wedge EF \wedge SAFE \wedge INV$$

Die vier Konjunkte beschreiben Eigenschaften, die ein Plan erfüllen muß, damit er als Lösung akzeptiert wird und zwar eine Terminierungsbedingung, eine Effektbeschreibung in Form von Vor- und Nachbedingungen, Sicherheitsbedingungen, die in allen Zwischenzuständen gelten müssen und Invariantenbedingungen, d.h. Fakten, die unverändert bleiben müssen.
Die initiale Spezifikation wird nun durch Anwendung korrektheitserhaltender Transformationen solange verfeinert, bis ein ausführbarer Plan entsteht. Verwendung finden dabei abstrakte Prozeduren, die abhängig von der jeweiligen Ausgangssituation zu unterschiedlichen Plänen konkretisiert werden können.

Bibels "Lineare Konnektionsmethode"

Bibel [Bibel 86] schlägt vor, seine Konnektionsmethode [Bibel 83, Bibel 87] auch zur Lösung von Planungsproblemen zu verwenden. Die Idee ist, nur *lineare* Beweise zu verwenden, d.h. Beweise, in denen jede Instanz eines Literals höchstens einmal an einer Konnektion beteiligt ist. Die Konnektionsmethode ist ein Beweisformalismus, der versucht, soviel Redundanz wie möglich aus dem Deduktionsprozeß fernzuhalten. Es wird dabei auf die Erzeugung von neuen Formeln verzichtet, stattdessen wird etwa die Widersprüchlichkeit der untersuchten Formel direkt aus ihrer inneren Struktur abgeleitet. Ein aktueller Vergleich verschiedenster Methoden und Kalküle zur Deduktion findet sich z.B. in [Gabbay et al. 93]; es wird deshalb hier auch nicht weiter auf die genaue Wirkungsweise der Methode eingegangen.
In der linearen Konnektionsmethode werden Literale als Resourcen, die konsumiert bzw. produziert werden, angesehen. Sobald ein Literal an einer Konnektion teilhat, ist es konsumiert und nicht mehr verfügbar. Der Ansatz erlaubt es somit, ohne Rahmenaxiome auszukommen, da Literale (Eigenschaften) so lange Zustandsübergänge überleben, bis sie konsumiert werden. Eine Semantik für die Methode wurde z.B. in [Bibel et al. 89] angegeben.
Eine Erweiterung der Konnektionsmethode ist in [Bibel 97] beschrieben. Er führt hier die sogenannte *"transition logic"* ein, die darauf beruht, die ressourcensensitive Konnenktionsmethode in klassische Logik einzubetten, so daß klassisches und ressourcenspezifisches Schließen in einer Logik vereint sind.

Der Ansatz von Schneeberger und Hölldobler

Schneeberger und Hölldobler [Hölldobler & Schneeberger 90] modellieren in ihrem ELP^2-Ansatz ebenfalls eine Art von Resourcenabhängigkeit. Verwendet wird eine Hornklauselogik mit Gleichheit als Repräsentationsmechanismus und SLDE-Resolution[14] (siehe etwa [Hölldobler 89, Große et al. 92]) in ihrem Kalkül zur Generierung von Plänen.
Situationen oder Zustände werden mit Hilfe der binären Funktion $\circ$ repräsentiert, die als assoziativ und kommutativ definiert angenommen wird; sie repräsentiert dadurch eine Situation als eine Multimenge. Der Planungsprozeß wird durch ein dreistelliges Prädikat $plan(s_1, p, s_2)$ spezifiziert. Die Intention ist, daß die Situation s_1 von der Situation s_2 aus erreicht werden kann, indem Plan p ausgeführt wird. Situationen und Pläne sind damit Terme in der Logik. Aktionen sind durch Regeln der Form

$$plan(preconditions \circ V, [action|P], W) : -plan(postconditions \circ V, P, W)$$

definiert. Zur Terminierung einer Ableitung und zur Spezifikation einer leeren Aktion ist ein Fakt

$$plan(V \circ W, [], W)$$

notwendig, der beschreibt, daß es nichts zu tun gibt, wenn die Zielsituation bereits Teil der Ausgangssituation ist (die Variable V bindet in diesem Fall den Rest der Ausgangssituation).
Am Beispiel einer *move*-Aktion, die einen Block von einer Position auf eine andere bewegt, erhält man:

$$plan(clear(X) \circ clear(Y) \circ on(X, Z) \circ V, [move(X, Y)|P], W) : - \\ plan(clear(X) \circ clear(Z) \circ on(X, Y) \circ V, P, W).$$

Den Planungsprozeß stößt man nun durch eine Anfrage an das spezifizierte logische Programm an, z.B. durch

$$: -plan(ontable(a) \circ ontable(b) \circ clear(a) \circ clear(b) \circ handempty, P, on(a, b)).$$

und erhält dann als Antwortsubstitution für die Variable P etwa den Plan $[move(a, b)|[]]$, also die Aktion $move(a, b)$. Dies bedeutet nun, daß es einen Plan gibt, der die initiale Situation überführt in eine Situation, in der Block a auf Block b steht.
Durch die explizite Verwaltung von Multimengen als Situationsbeschreibungen (durch die Funktion $\circ$) wird das Rahmenproblem ohne Angabe von Rahmenaxiomen gelöst. Durch die Anwendung einer Aktion werden ihre Vorbedingungen *verbraucht*[15].

Ein dritter resourcenorientierter Ansatz basiert auf *Linearer Logik* [Girard 87, Girard 93] und wurde durch Masseron präsentiert [Masseron et al. 90]. Als kurze Charakterisierung sei hier nur angebracht, daß die Handhabung von Multimengen wie im ELP^2-Ansatz hier in den deduktiven Kalkül, einen speziellen Sequenzenkalkül, eingebaut ist.
Ein interessantes Resultat von Schneeberger [Schneeberger 92] ist nun, daß die Ansätze zur Generierung von Plänen durch die lineare Konnektionsmethode, den ELP^2-Ansatz und den "Lineare Logik"-Ansatz für konjunktive Planungsprobleme[16] äquivalent sind.

[14] Der traditionelle Unifikationsalgorithmus ist dabei durch eine AC1-Unifikationsprozedur ersetzt.

[15] Die *AC1*-Unifikation berechnet den exakten Rest der Ausgangssituation.

[16] Der Begriff wird hier gebraucht, um Restriktionen bei der Domänen- bzw. Problembeschreibung zu charakterisieren. Initiale Situationen und Ziele werden als Multimengen von Grundatomen beschrieben. Die Vorbedingungen bzw. Effekte von Operatoren sind Multimengen von Literalen. Axiome über Randbedingungen sind nicht erlaubt.

Weitere Ansätze aus dem logischen Programmieren

Zhang [Zhang 94, Zhang & Barringer 94a] beschreibt einen neuen nichtlinearen deduktiven Planungsansatz, "*Constraint Logic Planning*". Er kombiniert dabei das Auflösen von Randbedingungen für nichtlineares Planen mit einem Ansatz zum temporallogischen Programmieren. Er benutzt eine reifizierte Intervallogik und repräsentiert intervallrelevante Information mittels zweier Prädikate *Holds* und *Exec*. Die Formel $Holds(P, t_i, t_j)$ drückt aus, daß die Eigenschaft P über dem maximalen Zeitintervall (t_i, t_j) gilt, d.h. sie wird zum Zeitpunkt t_i zugesichert und zum Zeitpunkt t_j wieder gelöscht; $Exec(A, t_i, t_j)$ beschreibt, daß die Aktion A über dem maximalen Zeitintervall (t_i, t_j) ausgeführt wird. Die Basiskomponenten von nichtlinearen Plänen, d.h. Aktionen und zeitliche Beziehungen zwischen ihnen, werden als Randbedingungen (engl. *constraints*) repräsentiert[17]. Ein Planungsproblem wird gelöst, indem schrittweise durch den Einsatz von *Planungsregeln*, die Spezifikationen von Aktionen in der Intervallogik darstellen, ein nichtlinearer Plan erstellt wird. Es werden dabei nur zeitliche Constraints zwischen Aktionen eingeführt, wenn sie notwendig sind.
Eine Planungsregel für eine Aktion A ist schematisch von der Form:

$$Hold(p, t, SKF(t, v_1, \ldots, v_n)) : -B_1, \ldots, B_m, Exec(A, t, t), C$$

Die B_i sind von der Form $Hold(q_i, t_i, t_j)$, q_i gilt maximal in dem Intervall (t_i, t_j). C ist eine Formel, die zeitliche Constraints beschreibt. Die Eigenschaft p ist eine Nachbedingung der Aktion A, parametrisiert mit den Variablen $v_1, \ldots, v_n$. SKF ist eine Skolemfunktion, die im Verlaufe des Planens konkretisiert werden kann.
Zu berücksichtigen sind weiterhin *Domain Constraints*, d.h. eine Menge von Formeln der Art

$$\forall t_1, t_2, t_3, t_4 (Hold(p, t_1, t_2) \wedge Hold(q, t_3, t_4) \rightarrow t_2 \leq t_3 \vee t_4 \leq t_1)$$

die ausdrücken, daß zwei bestimmte Eigenschaften sich gegenseitig ausschließen.
Zhang definiert eine Menge von Restriktionen über Aktionen und Domain Constraints, um zeigen zu können, daß unter den Aktionsrestriktionen ein nichtlinearer Plan konfliktfrei ist genau dann, wenn sein zeitliches Modell die korrespondierende Menge von Domain Constraints erfüllt. Aus einer Menge von Domain Constraints ist es auch möglich, eine Menge von entsprechenden *Aktionsbedingungen* zu erhalten; sie beschreiben etwa, daß sich zwei Aktionen unter bestimmten Bedingungen nicht überlappen dürfen. Es wird bewiesen, daß unter bestimmten Bedingungen ein nichtlinearer Plan konfliktfrei ist genau dann, wenn er eine korrespondierende Menge von Aktionsbedingungen erfüllt. Dies vereinfacht die generell notwendige Konflikterkennung im nichtlinearen Planen. Ein nichtlinearer Plan mit zwei nichtgeordneten Aktionen wird z.B. als Formel der Art

$$Exec(Puton(A, T, B), t_1, t_1) \wedge T_0 < t_1 \wedge$$
$$Exec(Puton(B, T, C), t_2, t_2) \wedge T_0 < t_2$$

beschrieben. T_0 bezeichnet hier den initialen Zeitpunkt vor Beginn des Plans.

[17] Eine ausführliche Übersicht über "Constraint Logic Programming"-Methoden gibt [Jaffar & Maher 94].

Planen im ALPS-Projekt

Das ALPS-Projekt [Calistri-Yeh & Segre 94a, Calistri-Yeh & Segre 94b] beschäftigt sich mit der Entwicklung einer adaptiven Planungsarchitektur zum Einsatz in der Transportplanungsdomäne. Die zwei Hauptmotivationen des Projektes sind zum einen, daß Realweltplanungssysteme notwendigerweise adaptiv sein sollten und zum anderen die Untersuchung, in wie weit Planen durch logische Deduktion für sehr große Transportplanungsprobleme geeignet ist.
Die Implementierung des Planers gründet sich auf eine deduktive Inferenzmaschine, die in dem Sinne adaptiv ist, daß ihre Performanz sich mit zunehmender Erfahrung verändert. Der verwendete Inferenzmechanismus ist Resolution für definite Klausellogik mit "*Iterative Deepening*" [Korf 85] als Suchstrategie. Es werden verschiedene *Speedup*-Methoden genutzt, um die Performanz des verwendeten Deduktionssystems zu erhöhen. Es hat sich gezeigt, daß erst die Übersetzung der Inferenzmaschine in ein äquivalentes Programm, das unnötigen Zusatzaufwand, der inhärent bei der Interpretation logischer Regeln entsteht, vermeidet, eine dramatische Geschwindigkeitssteigerung bewirkt.

Temporallogik zur Suchkontrolle

Bacchus und Kabanza [Bacchus & Kabanza 95, Bacchus & Kabanza 96] konzentrieren sich in ihrer Arbeit auf die Verwendung einer metrischen Intervalltemporallogik zum einen zur Spezifikation komplexer zeitlich strukturierter Planungsziele und zum anderen zur Spezifikation von Planungskontrollwissen. Beide Spezifikationsarten drücken in Form temporallogischer Formeln ein von einem gesuchten Plan zu erbringendes zeitliches Verhalten über aufeinanderfolgenden Zuständen aus. Ihr Planungsmechanismus ist eine vorwärtsverkettende Inferenzmaschine, die lineare Sequenzen von Aktionen mit Hilfe von Progression über vollständigen Zustandsbeschreibungen bzgl. vorgegebener Aktionsspezifikationen generiert. Im Verlaufe der Generierung der linearen Zustandssequenzen[18] werden diese inkrementell gegenüber dem spezifizierten zeitlichen Zielverhalten getestet. Wann immer dabei gezeigt werden kann, daß die entstehende Sequenz das Erreichen des Zielverhaltens unmöglich werden läßt, so werden die Sequenz und alle ihre Erweiterungen von der weiteren Suche nach einer Lösung abgetrennt. Gleichzeitig kann die Suche gestoppt werden, sobald eine Sequenz gefunden ist, die das gewünschte Zielverhalten aufweist. Die Basis der inkrementellen Technik wird durch eine *Formelprogression* gegeben. Dabei wird der initiale Zustand durch das zu erreichende Zielverhalten *beschriftet*, also die Bedingung, nach der initial gesucht werden soll. Nachfolger des initialen Zustandes, die durch die gegebenen Aktionsdefinitionen deterministisch bestimmt sind, erhalten durch Progression der initialen Beschriftung eine neue Beschriftung, die wiederum festlegt, nach welcher Bedingung nun noch gesucht werden muß, um das spezifizierte Zielverhalten zu erreichen. Die Progression einer temporallogischen Beschriftung verhält sich dabei analog zur durch die Semantik der Temporallogik definierten Zustandsübergangsfunktion, d.h. die Interpretation einer temporallogischen Formel ist direkt operationalisiert in dem Steuerungsmechanismus eines Planungsverfahrens. Eine Lösung ist gefunden, wenn sich als neue Beschriftung eines Zustandes *true* (wahr) ergibt.
Die Konsequenz aus dem beschriebenen Vorgehen ist, daß eine effektive Suchraumkontrolle nur durch die zusätzliche Spezifikation von geeignetem *domänenspezifischem Zustandsverhalten während einer Problemlösung*, ausgedrückt als temporallogische Formel,

[18] Und damit der Aktionssequenzen.

erreicht werden kann. Fehlt diese Information, ist das Verfahren mit einer *blinden* Suche gleichzusetzen.

Taktisches Planen ist ein Gebiet, das aktuell an Bedeutung gewinnt [Biundo 94]. Verstanden wird unter diesem Begriff, die Verwendung formaler Planungsmethoden, die unter dem Einsatz von taktischen Theorembeweisern realisiert werden (vgl. hierzu auch den Abschnitt 2.2.1). Der Vorteil liegt dabei zum einen in der hohen Flexibilität dieser Systeme und zum anderen wird die Verwendung von formalen Schlußfolgerungsmethoden für das Planen zu einem vertretbaren Aufwand realisierbar.[19]

Der Ansatz von Cranefield

Cranefield [Cranefield 92] beschreibt ein Planungssystem basierend auf einer Logik von Planspezifikationen. Der Benutzer hat dabei gewisse Kontrolle über das Verhältnis von Ausdrucksmächtigkeit und Komplexität des Planungsmechanismus, indem er zum einen eigene Plankompositionsoperatoren mit korrespondierenden Inferenzregeln definieren kann und zum anderen die Komplexität testbarer Bedingungen durch entsprechend zu formulierende Anfragen in einer logischen Programmiersprache steuern kann.
Die Basis ist ein einfach strukturiertes Weltmodell in Form von Mengen von Tripeln der Art $(entity, attribute, value)$; somit sind Zustände beschrieben als Konjunktionen $attribute_1(entity_1) = value_1 \wedge \ldots \wedge attribute_n(entity_n) = value_n$. Planspezifikationen haben bei Cranefield folgende Form: $\Gamma \triangleright \langle \phi, P, \psi \rangle$, mit

- P ein Planterm oder eine atomare Aktion.
- ϕ bzw. ψ Mengen von Gleichungen $attribute(entity) = value$, ϕ sind die unmittelbaren Vorbedingungen für P und ψ die Effekte in Form von Wertveränderungen der Gleichungen aus ϕ.
- Γ ist eine Menge von Literalen, die benutzt werden, um komplexe Vorbedingungen zu testen und um Werte für Variablen in ψ zu konstruieren; die Literale werden als Anfragen einer logischen Programmiersprache während des Planens evaluiert.

Die Interpretation einer Planspezifikation besagt, daß immer dann, wenn die Literale in Γ erfüllt sind, die Ausführung von P auf alle Zustände, die ϕ erfüllen, den gleichen Effekt hat, d.h. ψ beschreibt die minimalen Veränderungen von Zuständen. Eine Aktion zur Bewegung eines Blockes von einer auf eine andere Position ist z.B. beschrieben als:

$$\left\langle \left\{ \begin{array}{l} loc(A) = T_1 \\ on(T_1) = A \\ on(T_2) = nil \end{array} \right\}, move(A, T_1, T_2), \left\{ \begin{array}{l} loc(A) = T_2 \\ on(T_1) = nil \\ on(T_2) = A \end{array} \right\} \right\rangle$$

Planen geschieht nun durch Beweisen von Planspezifikationen (der Planterm P ist dann im allgemeinen zunächst eine Variable) durch die Anwendung spezieller Inferenzregeln im Stile des interaktiven Beweisens mit ISABELLE [Paulson 89, Paulson 90]. Als Axiome dienen Aktionsbeschreibungen der Art, wie gerade oben am Beispiel von *move* gezeigt. Die Inferenzregeln geben etwa eine Zieldekomposition mit einer entsprechend einhergehenden Plandekomposition an. Die Entscheidungen des Planers sind dann jeweils:

[19] Im Gegensatz etwa zu dem Einsatz allgemeiner Beweiser.

- Auswahl eines zu dekomponierenden Teilzieles,
- Auswahl einer anwendbaren Inferenzregel,
- Auswahl eines der möglichen Unifikatoren von aktuellem Teilziel und Konklusion der Regel.

"Pläne sind Taktiken", Guinchiglia und Traverso et al.

Guinchiglia und Traverso et al. [Traverso et al. 92, Guinchiglia et al. 92] beschreiben eine domänenunabhängige Entwicklungsumgebung für Planungssysteme, `MRG`. Die Idee ist, daß komplexe Realweltanwendungen einen erweiterten Begriff "Plan" erfordern. Ein Plan ist nicht nur ein Aktionsplan im klassischen Sinne, sondern enthält auch Aktivitäten wie Plangenerierung, Planausführung, Reaktionen auf externe Ereignisse und Fehlerbehandlungen. Als Kontrollstrukturen sind neben der Sequenzialität auch Konditionale und Rekursion erlaubt. Die `MRG`-Sprache ist eine Programmiersprache, in der die Taktiken beschrieben werden können. Sowohl Aktionspläne als auch Planungsaktivitäten und Kontrollmechanismen eines Planungsagenten werden damit in einer einzigen Sprache repräsentiert. Verschiedene Planungsmechanismen können so mit Hilfe von Taktiken repräsentiert werden, d.h. Planungssysteme können durch die Ausführung von entsprechenden Taktiken implementiert werden. Ein klassischer Planer wird in MRG etwa beschrieben als:

```
(deftac classical-planner (goal)
 (let (plan (plan-for goal))
  (orelse (exec plan)
   (then (modify-planner-status)
         (classic-planner goal)))))
```

`plan-for` und `exec` sind dabei eine Plangenerierungs- bzw. Planausführungstaktik.
`modify-planner-status` aktualisiert die interne Weltrepräsentation des Planers in Abhängigkeit eines möglicherweise auftretenden Fehlschlages der Planausführung.
Neuere Entwicklungen gehen dahin, *Programmtaktiken*, Taktiken in einer Programmiersprache wie sie etwa in `MRG` benutzt werden, mit sogenannten *Logiktaktiken* in Beziehung zu setzen [Giunchiglia & Traverso 93, Giunchiglia & Traverso 94]. Taktiken sind dabei Terme in einer Metatheorie erster Ordnung [Giunchiglia & Traverso 92]. Die Metatheorie beschreibt die Berechnung, die Deduktionen in der Objekttheorie ausführen. Es ist damit möglich, Eigenschaften von Logiktaktiken zu beweisen und insbesondere auch vorhandene Taktiken zu modifizieren bzw. neue zu synthetisieren. Ein zweiter Punkt ist, daß es eine Abbildung zwischen Logiktaktiken und Programmtaktiken gibt, d.h. ein bidirektionale Übersetzung ist möglich.

Verwandte Gebiete Neben deduktiven Ansätzen zur Plangenerierung gibt es auch Ansätze zur deduktiven Planverifikation [Subramanian 93]. Dabei werden Pläne spezifiziert als plangenerierende LISP-Programme und verifiziert durch den Beweis der Korrektheit der Programme. Als Beschreibungssprache wird die Boyer-Moore-Logik und deren Theorembeweiser NQTHM verwendet (vgl. [Boyer & Moore 79]).

Die beschriebenen Situationslogikansätze nehmen wie auch die 'Dynamische Logik'-Ansätze eine objektsprachliche Trennung zwischen Plänen und Domänenbeschreibungen bzw. -axiomatisierungen vor. Eine konsequent deduktive Umsetzung der Verfahren mit Stan-

dardbeweisern ohne dedizierte Ansteuerung führte nicht zu zufriedenstellenden Ergebnissen. Wenn der Logikansatz primär auf der Repräsentationsebene eingesetzt wird und der Standarddeduktionsmechanismus durch einen spezifischen Algorithmus ersetzt wird, ist eine effizientere Vorgehensweise möglich, jedoch mit Einschränkungen in der Ausdrucksmächtigkeit und Flexibilität sowohl bei der Planspezifikation als auch bei den Plänen. Bibel's Ansatz verwendet bei einer streng deduktiven Vorgehensweise zwar einen leistungsstarken Beweiser, jedoch gibt es bei beschränkter Plandarstellungsmöglichkeit zusätzlich die Einschränkung, daß eine planungsspezifische Ansteuerung des Beweisers schwierig ist.
Die Ansätze in [Biundo et al. 92a] und [Stephan & Biundo 96] vollziehen eine objektsprachliche Gleichbehandlung von Planspezifikationen und Plänen, beides sind Formeln in einer gemeinsamen Logik. Diese Vorgehensweise bereitet den Weg hin zu einer flexiblen und ausdrucksstarken Planspezifikation und -darstellung. Der zweite Ansatz erlaubt gegenüber dem ersten eine komplexere Spezifikation von Aktionsbeschreibungen, durch die damit verbundene Aufhebung der zeitlichen Atomarität einer Aktion gestaltet sich das zeitliche Schließen im Zusammenhang mit komplexen Planstrukturen als aufwendiger als dies in dem ersten Ansatz der Fall ist. Beide Ansätze sehen die Verwendung "programmierbarer" Beweiser vor und ermöglichen damit eine planungsspezifische Ansteuerung.

Aufgrund des in dieser Arbeit verwendeten Ansatzes des taktischen Theorembeweisens zur Durchführung des deduktiven Planens auf Basis von [Biundo et al. 92a] folgt nun ein kurzer Überblick dieser Technik.

2.2.1 Taktisches Beweisen

Im Gebiet der *Automatischen Deduktion* gilt es inzwischen als ein anerkannter Ansatz, eine Metasprache zu benutzen, um Beweisstrategien als Taktiken zu beschreiben (vgl. z.B. [Basin & Constable 92]). Die Metasprache ist gewöhnlich eine (funktionale) Programmiersprache und Taktiken sind Programme, die spezifizieren, wie Inferenzregeln der Objektlogik zu verwenden sind, um einen Beweis zu erstellen. Taktiken verfeinern dabei ein Ziel (im Sequenzenstil repräsentiert) in angenommenerweise einfachere Teilziele. Die einfachsten Taktiken sind dann gerade die primitiven Verfeinerungsregeln. Durch spezielle Taktikkonstrukte können diese zu komplexeren Taktiken kombiniert werden. Taktiken stellen sich damit als Schlußregeln von flexibler Granularität dar, deren Abstraktionsgrad dem Problem angepaßt werden kann. Taktisches Theorembeweisen steht letztendlich für eine Art kooperative Deduktion, wobei die Integration automatischer Deduktion und menschlicher Beweiserfahrung möglich wird. Ursprünglich wurden Taktiken in LCF [Gordon et al. 79] eingeführt und in der Sprache ML [Wikstroem 87, Paulson 91, Ullman 94] geschrieben. Taktikbasierte Theorembeweiser wie NUPRL [Constable 86], HOL [Gordon & Melham 93] oder ISABELLE [Paulson 90] sind Nachfolger von LCF. Obwohl ihre Objektsprachen sich voneinander unterscheiden, haben sie als gemeinsames Merkmal die Metasprache ML.[20]
Die Einführung von Taktiken als neue Inferenzregeln beseitigt das Problem der Suchraumbewältigung bei der Beweissuche oft nicht alleine. Bundy (siehe [Bundy 96] für einen aktuellen Überblick) schlägt vor, eine Metalogik zur Beweisplanung zu benutzen, um über

[20] OYSTER [Bundy et al. 90], die Edinburgh-Version von NUPRL, benutzt PROLOG (vgl. z.B. [Clocksin & Mellish 87]) als Metasprache.

Beweise zu schließen und sie zu planen. Beweispläne sind dabei Kombinationen von *Methoden*, die wiederum Spezifikationen von Taktiken darstellen und zwar in dem Sinne, daß wenn eine Zielformel mit dem Eingabemuster der Taktik *matcht* und bestimmte Vorbedingungen erfüllt sind, dann ist die Taktik anwendbar; wenn sie dann erfolgreich endet, gelten bestimmte Effekte, d.h. Bedingungen der resultierenden Zielformeln. Realisiert ist solch ein Konzept im CLAM-System [Bundy et al. 90]. Für eine konkrete Beweisaufgabe wird dabei zunächst ein Beweisplan auf der Methodenebene erstellt. Eingesetzt wird diese Technik z.B. bei der Programmsynthese [Kraan et al. 92, Kraan et al. 93] oder auch bei Konfigurierungsarbeiten [Lowe 91]. Spezielle Kontrollstrategien sind notwendig, um einen Beweisplan zur Lösung einer neuen Aufgabe zu konstruieren. Der Ansatz von Melis [Melis 95] realisiert dazu z.B. eine analogiegetriebene Beweisplankonstruktion; ein bereits erfolgreich ausgeführter (vorgegebener) Beweisplan wird verwendet, um den Beweisplan für ein neues Problem zu erstellen. Der vorgegebene Plan wird dabei gegebenenfalls auch umformuliert. Die analogiegetriebene Kontrollstrategie wird auch im Beweisplaner des ΩMEGA-Systems (vgl. [Benzmüller et al. 97]) verwendet. ΩMEGA ist ein Beweisassistenzsystem, das den Beweisprozess als eine nebenläufige Abfolge von Beweisplanung, Planausführung und -verifikation ansieht. Hierbei spielt eine benutzergerechte Beweispräsentation eine wichtige Rolle [Huang & Fiedler 96].
Eine weitere Verfeinerung der Beweisplantechnik ist in [Manning et al. 93] beschrieben. Die Veränderung manifestiert sich darin, daß zum einen in jedem Zustand der Suche nach geeigneten anwendbaren Methoden jeweils nur die beste Methode gemäß einem heuristischen Maß verwendet wird; zum anderen ist durch die Verwendung einer *Beste-zuerst*-Suche das "normale" Backtracking durch ein opportunistisches Backtracking ersetzt. Man erreicht somit ein dynamisches Umordnen der anwendbaren Strategien.
Ein Ansatz zur Programmsynthese in einem anderen metalogischen System, dem *Edinburgh Logical Framework (LF)* [Harper et al. 87] wird in [Anderson 94a] beschrieben, Basin [Basin 94] beschreibt einen Ansatz zur Generierung logischer Programme im ISABELLE-System. Ein Programmverifikationsansatz unter Verwendung von taktischen Theorembeweisern ist z.B. im System KIV [Heisel et al. 90] realisiert. Dieses wurde z.B. auch für die Implementierung eines speziellen deduktiven Planungssystems verwendet (vgl. [Berens 97]).

2.2.1.1 Metalogisches Programmieren

Metalogisches Programmieren ist eine wesentliche Grundlage für die Implementierung von taktischen Beweisern. Aus diesem Grund werden hier ein paar Bemerkungen aus der Sicht der logischen Programmierung angeführt.
Eine wichtige Technik der Methode des logischen Programmierens ist die Metaprogrammierung [Kowalski 79, Kowalski 91]. Dabei werden Programme oder Theorien als Datenobjekte behandelt bzw. manipuliert. Diese Technik wird im allgemeinen verwendet, um die Beschränkungen einfacher Ausführungsstrategien von logischen Programmiersprachen wie PROLOG ([Clocksin & Mellish 87]) zu überwinden und anspruchsvollere Strategien zu implementieren.
Metalogisches Programmieren kann auch erfolgreich dazu verwendet werden, um ausdrucksstärkeres Schließen auf der Objektebene zu erreichen. PROLOG selbst hat als formale Grundlage nur Hornklausellogik, aber Modallogiken z.B. lassen sich darin sehr leicht

repräsentieren und das Schließen in solchen Logiken mithilfe eines entsprechenden Kalküls läßt sich durch ein geeignetes metalogisches Programm in PROLOG durchführen. Der zusätzliche Berechnungsaufwand, der durch den Ablauf eines Metainterpreters auftritt, kann durch Techniken wie etwa partielle Evaluation [Owen 89, Sahlin 93, Jones et al. 93] abgeschwächt werden.

Ein taktischer Beweiser für eine Logik $\mathcal{L}$ kann als Metaprogramm einer logischen Programmiersprache $\mathcal{O}$ angesehen werden:

> Formeln aus $\mathcal{L}$ sind Datenobjekte in $\mathcal{O}$; die Basisinferenzregeln eines logischen Kalküls für $\mathcal{L}$ werden durch $\mathcal{O}$-Programme dargestellt; die Steuerung der Anwendung von Inferenzregeln geschieht durch ein Metaprogramm in $\mathcal{O}$, das die $\mathcal{O}$-Abarbeitungsstrategie im Sinne der Emulation einer neuen Strategie auch verändern kann; ein solches Metaprogramm kann man dann als eine Taktik ansehen.

2.2.2 Algorithmisches Planen

Als Vertreter der algorithmisch, datenstrukturorientierten Planungsansätze sei der Ansatz von Kambhampati (vgl. z.B. [Kambhampati 94, Kambhampati & Srivastava 96]) erwähnt. Er beschreibt ein Rahmenwerk, das viele klassischen Planungsansätze vereinigt und nur als spezifische Ausprägungen seiner Planrepräsentation und allgemeinen *verfeinerungsbasierten* Planungsstrategie erscheinen läßt. Somit kann sein Ansatz als Repräsentant aller datenstrukturorientierten Planungsansätze dienen.

Ein Planungsproblem wird hierin als Tripel $\langle I, G, A\rangle$ beschrieben, mit I die Beschreibung des Anfangszustandes, G die Beschreibung des Zielzustandes und A eine Menge von Aktionen. Eine Aktionssequenz S wird als Lösung eines Planungsproblemes bezeichnet, wenn S, ausgeführt im Anfangszustand, letztlich zu einem Zustand führt, indem alle Ziele des Planungsproblems gelten. Bei der Lösung des Planungsproblems navigiert ein Planer über *Mengen potentieller Lösungen*. Diese Mengen sind als partielle Pläne repräsentiert. Ein partieller Plan P ist die Kurzbeschreibung für die Menge von Aktionssequenzen, die konsistent sind mit den Randbedingungen, die der Plan spezifiziert. Diese Menge wird als *Kandidatenmenge* $\ll P \gg$ bezeichnet. Durch die während des Planens vorgenommenen Verfeinerungen entstehen partielle Pläne, deren Kandidatenmengen jeweils Teilmengen, von denen früherer partieller Pläne sind; dies schreitet solange voran, bis eine Lösung aus einer der resultierenden Kandidatenmengen entnommen werden kann.

Ein partieller Plan ist nun als Quintupel $\langle T, O, B, ST, L\rangle$ beschrieben, wobei gilt:

- T ist eine Menge von Planschritten mit zwei ausgezeichneten Schritten t_0 und t_∞.
- ST ist eine Symboltabelle, die Planschritte auf Aktionen abbildet; t_0 wird auf die Aktion `start` (ihre Effekte korrespondieren mit I) und t_∞ auf die Aktion `finish` (ihre Vorbedingungen korrespondieren mit G) abgebildet.
- O ist eine partielle Ordnung über T.
- B ist eine Menge von Bedingungen über Variablen, die in den Aktionsbeschreibungen vorkommen; sie entsprechen Gleichungen oder Ungleichungen von Variablen mit

Werten oder anderen Variablen. Eine Grundlinearisierung eines partiellen Planes P ist eine bzgl. O konsistente Permutation seiner Planschritte, wobei alle Variablen durch Werte, konsistent mit B, ersetzt sind.

- L ist nun noch eine Menge von Zusatzbedingungen IPC, PTC und CC, die zulässige Ordnungen und Bindungen weiter einschränken:
  - Eine IPC-Bedingung ist ein Tripel $\langle t, p, t' \rangle$ und fordert, daß alle Kandidaten des partiellen Planes die Eigenschaft erfüllen, daß die Bedingung p zwischen den Aktionen, die den Schritten t bzw. t' entsprechen, erhalten bleibt.
  - Eine PTC-Bedingung ist ein Tupel $\langle p, t \rangle$ und fordert, daß für alle Lösungen des Planes gilt, daß in dem Zustand, in dem die mit Schritt t korrespondierende Aktion ausgeführt wird, die Eigenschaft p gilt.
  - Eine CC-Bedingung schließlich ist eine Relation zwischen zwei Schritten t und t', die fordert, daß in keinem der Kandidaten eines partiellen Planes eine Aktion zwischen den Aktionen, die t bzw. t' entsprechen, auftreten darf.

Zum Planen stehen nun drei prinzipielle Verfeinerungsarten zur Verfügung:

1. *Vorwärtszustandsverfeinerung*
   Es wird eine Aktion ausgewählt oder neu bestimmt, die im aktuellen Ausgangszustand ausgeführt werden kann. Ihre Ausführung bestimmt den neuen Ausgangszustand mit den dann geltenden Bedingungen. Zwischen altem und neuem Ausgangszustand wird eine CC-Bedingung eingeführt.

   Eine *dichte* initiale Aktionssequenz entsteht.

2. *Rückwärtszustandsverfeinerung*
   Analog wird hier eine Aktion ausgewählt oder neu bestimmt, die im aktuellen Zielzustand rückwärts ausgeführt wird.

   Eine dichte terminale Aktionssequenz entsteht.

3. *Planverfeinerung*
   Hier wird die Vorbedingung eines Planschrittes s ausgewählt und es werden eine ausreichende Zahl von Randbedingungen bzgl. Planschritten, Ordnungs- und Bindungsrelationen hinzugefügt, die garantieren, daß die Vorbedingung in Schritt s gilt.

   Es entsteht ein Netzwerk von Aktionen ohne strenge Aufbaustruktur.

Bei Verwendung der ersten beiden Verfeinerungen spricht man traditionell von *zustandsbasiertem* Planen, d.h. die Knoten des sich ergebenden Suchbaumes repräsentieren Weltzustände und Kanten zwischen den Knoten beschreiben Übergänge zwischen Weltzuständen, ausgelöst durch die Anwendung einer Aktion.
Bei Verwendung der Planverfeinerung spricht man von *planbasiertem* Planen, da hier die Knoten des aufgespannten Suchbaumes partiellen Plänen entsprechen und die Kanten Verpflichtungen darstellen, einen partiellen Plan auf verschiedene Weisen weiter einzuschränken. Ein partieller Plan charakterisiert dabei nicht eindeutig einen bestimmten Weltzustand.

Die Planrepräsentation und die Vorgänge bei der Verfeinerung von partiellen Plänen müssen nun zu einem Aktionsrepräsentations und -schlußfolgerungsformalismus in Beziehung gesetzt werden, um eine durchgängige Semantik für den gesamten Planungsprozeß zu erreichen. Erst dann ist es möglich, eine Basis für die Formulierung von aussagekräftigen Korrektheitsaussagen und damit zusammenhängend ein Modell für die Ausführung eines Planes zu erlangen. Ein algorithmischer Ansatz, der diesen Schritt konsequent vollzogen hat, jedoch nicht alle oben beschriebenen Planrepräsentationsoptionen und Verfeinerungsmethoden verwendet,[21] ist durch UCPOP [Penberthy & Weld 92] gegeben. Verwendet wird hier die Aktionsrepräsentationssprache ADL (vgl. [Pednault 88]). Es wird eine bijektive Abbildung der Plandatenstruktur in eine Menge dichter Aktionssequenzen definiert, wobei für jede dieser Sequenzen eine Ausführungsfunktion im logischen Modell von ADL definiert ist. Die Korrektheit des Planungsalgorithmus und der erzeugten Pläne muß also durch die Definition geeigneter Abbildungen und durch den Beweis entsprechender Invarianten des Algorithmus gesichert werden. Schon eine leichte Änderung oder Erweiterung des Planungsproblem- oder Planbegriffes führt zwangsweise zu einer veränderten Beweiskonstruktion.

Neuere Ansätze des algorithmischen Planens wie Graphplan [Blum & Furst 95] oder Descartes [Joslin & Pollack 96] beschäftigen sich mit disjunktiven Repräsentationen partieller Pläne. Graphplan z.B. nutzt dabei eine Repräsentation korrespondierend mit der Disjunktion von Vorwärtszustandsverfeinerungen und führt damit keine Verzweigungen in den Suchbaum ein. Die Repräsentation resultiert in einer ersten Phase des Verfahrens als sogenannter *Plangraph*. Dieser besteht aus einer alternierenden Abfolge von Fakten- und Aktionsschichten. Eine Faktenschicht enthält Grundfakten der Domäne indiziert mit dem Zeitpunkt der Schicht. Eine Aktionsschicht enthält vollinstanziierte Operatoren. Kanten des Graphen gehen zum einen von Fakten zu Operatoren, die sie dann als Vorbedingungen benötigen und zun anderen von jeder Aktion zu ihren Effekten in der jeweils folgenden Schicht. Es gibt zusätzlich "maintain"-Operatoren, die einen Fakt von einer Zeitstufe zur nächsten übertragen. Die Expansion des Grahen geht beginnend mit den Vorbedingungen des Planungsproblems so lange, bis eine Faktenschicht erreicht wird, in der alle spezifizierten Ziele enthalten sind.
In der zweiten Phase des Verfahrens geschieht nun die Lösungsextraktion als systematische Suche auf dem Plangraphen. Genutzt wird dabei eine Constraint-Propagierung basierend auf sich zu einem bestimmten Zeitpunkt gegenseitig ausschließenden Aktionen, d.h., keine Aktionsschicht darf am Ende Operatoren enthalten, die im Konflikt miteinander stehen. Constraint-Propagierung wird ebenfalls von Descartes verwendet, jedoch entstehen Disjunktionen von Planverfeinerungen.
Erste empirische Ergebnisse bescheinigen den Ansätzen Effizienzvorteile gegenüber klassischen Ansätzen.
Ein aktueller Ansatz, der ebenfalls eine sehr spezielle Problemkodierung mit einer Suche kombiniert, um einen sehr laufzeiteffizienten Planer zu erhalten, ist im SATPLAN-System [Kautz & Selman 96, Kautz et al. 96] kodiert. Das Planungsproblem wird hier als aussagenlogisches Erfüllbarkeitsproblem in konjunktiver Normalform formuliert. Ein Plan korrespondiert mit einem Modell, d.h. einer Wahrheitswertzuweisung, das eine Menge von

[21] *PTC*- und *CC*-Bedingungen sind nicht formuliert und es wird nur eine Planverfeinerung vorgenommen.

logischen Constraints, die den Anfangs- und Endzustand und die Domänenaxiome repräsentieren, erfüllt. *Zeit* wird als festgelegte, diskrete Anzahl von Schritten modelliert (die maximale Länge eines Planes ist somit vorab fixiert).
Die Effizienz des Verfahrens resultiert nun daraus, daß zum Testen des Erfüllbarkeitsproblems ein sehr schneller stochastischer Algorithmus mit lokaler Suche verwendet wird.

2.3 Formale Spezifikationen und logisches Programmieren

Formale Spezifikationen spielen in dieser Arbeit an verschiedenen Stellen eine Rolle: Planungsprobleme werden in einer formalen Logik spezifiziert, die Planungsstrategie ist aufgrund des verfolgten deduktiven Ansatzes formal als spezielles Beweisverfahren spezifiziert und schließlich wird auch eine Adaption der Planungsstrategie an unterschiedliche Anforderungen formal spezifiziert.
Es folgen nun ein paar Bemerkungen über den Einsatz formaler Methoden im Hinblick auf eine Begriffsklärung und auf die enge Verwandtschaft von formalem Planen und Softwareentwurf. Wie bereits in Abschnitt 2.1 erwähnt, gibt es eine Verbindung zwischen Plänen und Programmen, insbesondere, wenn man an die Roboterplanung denkt. Damit ist auch eine Verbindung zwischen Planen und Programmsynthese ([Lowry & McCartney 91]) gegeben. Eine Analogie der Programmsynthese zum Planen wäre etwa (sinngemäß aus [Subramanian 93]):

> "Schreibe ein PASCAL-Programm, das als Eingabe einen Blocksweltzustand und einen Block hat und als Ausgabe einen Blocksweltzustand produziert, in dem der Block frei ist. Im erreichten Zustand muß die gleiche Menge von Blöcken vorhanden sein, wie im Eingabezustand."

Solch eine Programmspezifikation würde man auch *Anforderungsspezifikation* nennen. Sie ist eine informale aber präzise Spezifikation des zu konstruierenden Programmes, etwa als Resultat einer *Anforderungsanalyse*. In der Sprache der Programmentwicklung [Pomberger & Blaschek 93] sind Pläne (z.B. für Blocksweltprobleme) nichts anderes als Prototypen, die gegenüber ihrer formalen Anforderungsspezifikation, gegeben durch Start- und Zielbedingungen, verifiziert werden können [Lafontaine et al. 91].
Formale Spezifikationssprachen sind im Softwareengineering ein Hilfsmittel, um den Programmentwicklungsprozeß präziser zu unterstützen, als das etwa von informalen Methoden geleistet werden kann. Verwendet werden sie z.B., um eine Spezifikation des Systems, das entworfen werden soll, zu erlangen (vgl. etwa die Sprache Z [Spivey 92]). Durch die Formalisierung können bereits frühzeitig Designfehler entdeckt bzw. beseitigt werden. Auf einer nächsten Stufe werden formale Spezifikationsmethoden im Entwicklungsprozess verwendet, z.B. zur präzisen stufenweisen Verfeinerung einer Spezifikation in ein ausführbares Programm (vgl. etwa die Sprache VDM [Fields 92, Bicarregui et al. 94]). Zunehmende Bedeutung erlangen formale Methoden bei der Entwicklung sicherheitskritischer Softwaresysteme [Bowen & Stavridou 93]. Beispielweise unterstützt das VSE-System [Hutter et al. 96] die formale Entwicklung von Software. Gegenüber konventionellen CASE-Tools bietet es zusätzlich die Möglichkeit, mit Hilfe formaler Spezifikations-

und Verifikationsmethoden die Korrektheit von Software-Systemen oder Teilen davon zu beweisen.
Einen Spezialfall formaler Spezifikationen bilden ausführbare formale Spezifikationen (vgl. [Fuchs 92, Fromherz 93]). Sie zeichnen sich nicht nur durch ihre Präzision aus, sondern erlauben auch ihre direkte Validierung durch das Ablaufenlassen in einem entsprechenden Interpreter. Logische Programmiersprachen bilden eine typische Klasse von ausführbaren Spezifikationssprachen.

Logisches Programmieren

Logik wird traditionellerweise z.B. dazu verwendet, um Programmspezifikationen zu formalisieren, um sie danach verifizieren zu können [Kowalski 91, Francez 92]. Für solche Zwecke hat logisches Programmieren entscheidende Vorteile: Programme und Spezifikationen sind im gleichen logischen Formalismus repräsentiert. Die Programmausführung, die Verifikation und die Synthese von Programmen können alle mit ähnlichen Methoden des logischen Schließens vollzogen werden.
Programmspezifikationen beschreiben die Funktionalität von Programmen auf deklarative, klare Weise, was aber meistens bedeutet, daß sie nicht effizient ausführbar sind, während die Programme hingegen effizient ausführbar sein sollten. Transformationstechniken überführen dabei Spezifikationen in Programme (siehe z.B. [Fuchs & Fromherz 92]).

Entsprechend der oben erwähnten Korrespondenz zwischen Programmen und Plänen entspricht eine formale Programmspezifikation einer Planspezifikation. Gemäß der in Abschnitt 2.2.1.1 beschriebenen Verbindung von Taktiken und Metaprogrammen kann eine Taktikspezifikation als Metaprogrammspezifikation betrachtet werden. Sind eine Taktik und ihre Spezifikation identisch, hat man eine ausführbare Spezifikation definiert.

2.4 Benutzermodellierung

Wenn Systeme zunehmend komplexer werden bzgl. dem großen Umfang von Informationen, die übermittelt werden, und einer großen Anzahl von Aufgabenstrukturen, die mit der Systemnutzung bzw. -bedienung assoziiert sind, dann ist es wichtig, daß ein System seinem Benutzer intelligente Führung und Hilfestellung geben kann, damit er die Funktionalität des Systems voll ausnutzen kann. Dies bedeutet auch, daß ein absoluter Neuling als *Lernender* in Form von *Intelligentem Tutoring* [Cyranek 91] adäquat unterstützt werden kann.
Da verschiedene Benutzer nicht alle die gleichen Probleme und Wünsche haben und der Wissensstand eines Benutzers sich während der Zeit ändert, ist es wünschenswert, daß sich Schnittstellen zu komplexen Systemen auf den Bedarf individueller Benutzer anpassen und ihre Anforderungen unterstützen können. Zur Repräsentation und Verwaltung der individualisierten Daten benötigt man ein *Benutzermodell* (engl. *user model*) [Kobsa & Wahlster 89]. Befindet man sich im Kontext eines intelligenten Tutoringsystems so spricht man meist von einem *Lernermodell* (engl. *student model*) [Elsom-Cook 93].
Die *Benutzermodellierung* beschäftigt sich nun mit dem Prozeß des Zusammenführens von Informationen über Benutzer eines Systems und der Nutzung dieser Information, um Leistungen und Informationen, die den spezifischen Anforderungen individueller Benutzer angepaßt sind, zur Verfügung zu stellen.

McTear [McTear 93] und Kobsa [Kobsa 93] geben einen Überblick über aktuelle Entwicklungen im Bereich der Benutzermodellierung. Ihre Ontologie ist hier auszugsweise wiedergegeben, soweit sie für die Einordnung der in dieser Arbeit angesetzten Benutzermodellierungsannahmen von Bedeutung ist (vgl. auch Abschnitt 3.1.1.2).
Es werden folgende prinzipiellen Arten von Benutzermodellen unterschieden:

- Das *implizite* Modell
  Der Designer eines Systems macht sich in der Entwurfsphase einmalig Gedanken über das Profil des zukünftigen Systembenutzers und legt somit das Benutzermodell statisch fest, d.h. es wird nicht explizit repräsentiert. Man erhält somit ein Modell des System-Designers über den Benutzer.

- Das *adaptierbare* Modell
  Der Benutzer hat hier die Auswahl zwischen verschiedenen Optionen, die das Systemverhalten beeinflussen, die in einem Benutzerprofil für eine zukünftige Verwendung aufbewahrt bleiben.

- Das *adaptive* Modell
  Hier wird Wissen über den Benutzer automatisch akquiriert, auf dem neuesten Stand gehalten und benutzt, um das Systemverhalten den Anforderungen des Benutzers anzupassen.

Eine typische Klassifikation von Benutzermodellierungssystemen unterscheidet die folgenden Dimensionen:

individuell vs. kanonisch: Man unterscheidet zwischen Modellen individueller Benutzer und Modellen ganzer Benutzergruppen (*Stereotypen*).

statisch vs. dynamisch: Die Frage ist, ob ein Benutzermodell während der Arbeit mit dem System verändert werden kann.

kurzzeitig vs. langzeitig: Das Benutzermodell kann sowohl nach einer Systemsitzung gelöscht werden als auch für eine spätere Verwendung erhalten bleiben; ebenso ist eine Mischform denkbar.

explizite vs. implizite Akquisition: Der Aufbau des Benutzermodells kann durch spezifische Anforderungen von Benutzerinformationen vollzogen werden. Durch spezielle Inferenzmethoden kann das Modell aber auch automatisch aufgebaut werden, ohne mit dem Benutzer in einen Dialog zu treten.

Ein Benutzerprofil kann eine Vielzahl unterschiedlicher Informationen enthalten. Im einzelnen können dies sein: Ziele und Pläne des Benutzers, Repräsentationen seiner Gewohnheiten und Fähigkeiten, Darstellungen von Wissen und Glauben über Konzepte der Domäne, Präferenzen von Aktionen und Verhaltensmustern und schließlich Repräsentationen beschränkter interner sowie externer Resourcen (vgl. hier speziell [Wahlster et al. 95]). Bezüglich dessen, was der Benutzer über die Domäne weiß, wird aus tutorieller Sicht meist eine *expertenbasierte* Modellierung vorgenommen [Elsom-Cook 93]. Alles, was über die Domäne bekannt ist, ist auch dem Experten, dem Tutor, bekannt. Das Wissen des

Benutzers/Studenten ist unter Verwendung dieses Wissens repräsentierbar und unterscheidet sich von dem Wissen des Experten hinsichtlich der Vollständigkeit. Eine weitere Unterscheidung kann man nun dahingehend treffen, ob bzgl. des Benutzers nur Konzepte modelliert werden, die denen des Experten entsprechen, oder ob das Benutzermodell auch Mißdeutungen von Konzepten enthalten kann.
Der offensichtlichste Zweck eines Benutzermodelles besteht darin, die Systemausgabe so zu erstellen und zu gestalten, daß sie zur Erfüllung der Bedürfnisse des Benutzers maßgeschneidert ist (siehe auch [Sarner & Carberry 92]). Im Hinblick auf die Erstellung eines Planes für den Benutzer[22] besteht die typische Situation häufig darin, daß das System ausgiebigeres und genaueres Domänenwissen als der Benutzer hat, während hingegen der Benutzer Wissen über seine Intentionen und Präferenzen hat, die entweder Restriktionen an oder potentielle Beeinflußung für einen zu konstruierenden Domänenplan sind. Präferenzen beeinflußen dabei die Selektion und Instantiierung von Aktionen, die die Ziele des Benutzers erreichen [Elzer et al. 94].
Ein Benutzermodell kann aber auch hilfreich sein, wenn die Eingaben des Benutzers vom System verstanden werden sollen, etwa im Sinne der Erkennung der Ziele und Pläne des Benutzers.[23] Zu suboptimalen Lösungen können dann z.B. Verbesserungsvorschläge vom System gemacht werden [Elzer et al. 94].

2.5 Planqualität

Im allgemeinen ist es unbefriedigend, wenn man zwar einen Planungsmechanismus hat, der einem erlaubt, Pläne zu generieren, aber kein Qualitätsmaßstab für die erzielbaren Ergebnisse deklariert ist. Für ein spezifiziertes Planungsproblem gibt es fast immer multiple Lösungen, wobei in einer konkreten Situation aber oft nur eine bestimmte für den beabsichtigten Plankonsumenten von Nutzen ist. In [Perez & Carbonell 94] wird eine Taxonomie von Qualitätsmetriken aufgestellt, die sich grob in drei große Bereiche gliedert (vgl. Abbildung 2.1).

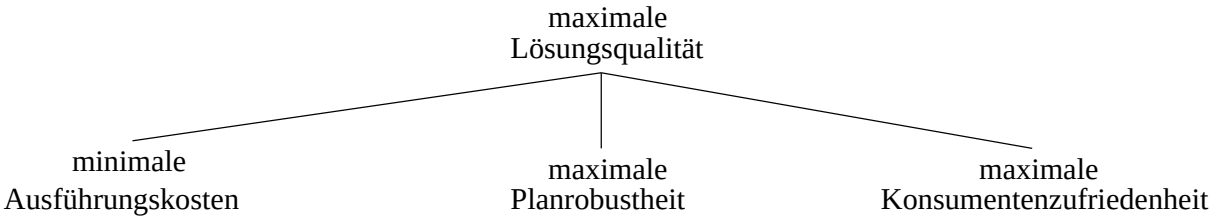


Abbildung 2.1: Partielle Taxonomie von Qualitätsmetriken

Die in der Planungsliteratur vorgeschlagenen Ansätze, die Planqualität berücksichtigen, finden sich im wesentlichen in der ersten Gruppe, der Erreichung minimaler Ausführungskosten, wieder. Darin wiederum wird häufig das Problem der Erreichung minimaler Planlänge fokussiert.
Betrachtet man Optimierungsprobleme beim Planen, darf das Komplexitätsproblem des Planens selbst nicht außer acht bleiben. Eine Vielzahl von Arbeiten untersuchen das Kom-

[22] Eine der wesentlichen Plangenerierungsanwendungen dieser Arbeit.
[23] Zur Rolle der Planerkennung in interaktiven Systemen vgl. z.B. [Goodman & Litman 92].

plexitätsproblem des Planens unter Berücksichtigung verschiedener Einschränkungen für Beschreibungsformalismen und spezifischer Probleme. Die Komplexität von domänenunabhängigem Planen in Abhängigkeit von der Art der Planungsoperatoren wird in [Erol et al. 92] untersucht. In [Bylander 92] werden verschiedene Komplexitätsklassen für das erweiterte propositionale STRIPS-Planen analysiert, eine Verallgemeinerung früherer Arbeiten [Bylander 91].
Bäckström [Bäckström 92, Bäckström & Nebel 93] legt ebenfalls eine detaillierte Untersuchung von Komplexitätsklassen für propositionales Planen vor.
Bezüglich spezifischer Blocksweltplanungsprobleme haben Gupta und Nau Untersuchungen angestellt [Gupta & Nau 92], um die Gründe der NP-Härte[24] des Findens optimaler Pläne zu entdecken.
Aufgrund der allgemeinen Komplexität des Planungsproblems gibt es zahlreiche Ansätze, die mit unterschiedlichen Methoden versuchen, eine Effizienzsteigerung im Suchprozeß des Planens zu erreichen. Es seien hier nur ein paar Forschungsrichtungen aufgezeigt:

- *Einsatz von statischem Kontrollwissen zur Suchraumbeherrschung*
  Die Verwendung von Abstraktionen ist eine akzeptierte Technik zur Organisation des Planungsprozesses. ABSTRIPS [Sacerdoti 74] war der erste abstraktionsbasierte Planer. Absteigend von der höchsten Abstraktionsebene wird dabei auf jeder der Ebenen zunächst vollständig geplant, bevor eine tiefere Ebene betreten wird. Eine spätere Formalisierung des Ansatzes und eine Analyse der Auswirkungen von Abstraktionen wird in [Knoblock et al. 91, Knoblock 94] untersucht. Eine aktuelle Untersuchung der Verwendung von Abstraktionen zum Realzeitplanen ist in [Washington 94] beschrieben.
  Das automatische Auffinden von Ordnungen zwischen zu planenden Teilzielen zur Vermeidung von negativen (zeitverschwendenden) Interaktionen zwischen geplanten Konkretisierungen ist eine weitere prinzipielle Technik zur Effizienzsteigerung des Planungsprozesses (vgl. etwa die Ansätze von [Cheng & Irani 89, Etzioni 91]).
  Ein anderer Ansatzpunkt der Beeinflussung des Planungsvorganges ist die Auswahl von Aktionen zur Erfüllung von Teilzielen. Hier setzt z.B. der Ansatz von Fink und Yang [Fink & Yang 93] an, die eine Aktion nur dann zur Erreichung eines bestimmten Effektes präferieren, wenn dieser als *Primäreffekt* der Aktion analysiert wurde.

  Das PRODIGY-Planungssystem (vgl. [Carbonell & the PRODIGY Group 92], [Fink & Veloso 95]) ermöglicht die Definition von expliziten zustandsbasierten *Kontrollregeln* zur Suchraumkontrolle. Der verwendete (fixe) Planungsalgorithmus sieht vor, daß verschiedene Regeln in Abhängigkeit von Eigenschaften des gerade zu expandierenden Knotens des Suchraumes aktiviert werden können. Entscheidungen, die dabei getroffen werden können, betreffen die Auswahl einer noch offenen Entscheidung zur weiteren Bearbeitung, die Fokussierung eines der aktuell offenen Ziele, die Auswahl eines Operators (Aktionsschemas) zur Erreichung eines bestimmten Zieles, die Auswahl einer Variablenbindung, um einen ausgewählten Operator zu instantiieren und zuletzt die Wahl, einen anwendbaren Operator tatsächlich anzuwenden oder mit der Teilzielverfeinerung (im Sinne einer *Means-ends-analysis*) fortzufahren. Da die Kontrollregeln eine Vielzahl von "Metaprädikaten" enthalten können,

[24] Zur Begriffserläuterung siehe etwa [Papadimitriou 94].

die mit der speziellen Gestaltung des Planungssystems in Verbindung stehen, ist ein effektiver Einsatz von Kontrollregeln nur bei genauer Kenntnis des Systems möglich.

- *"speed-up"-Lernen von Planerkontrollwissen*
  Durch den Einsatz der EBL-Technik[25] [Minton et al. 89] wird Kontrollwissen des Planers gelernt, um das Suchverhalten nach Lösungen zu optimieren, d.h. der Lernmechanismus wird von Effizienzzielen getrieben. Man sucht dabei nach Gründen, warum eine Lösung besser ist als eine andere, und setzt die Begründung in explizite Kontrollregeln um. Aufbauend auf dem PRODIGY-Planungssystem existieren dazu verschiedene Ansätze (vgl. [Etzioni 93, Veloso & Carbonell 93] und neuere Arbeiten in [Iwamoto 94] bzw. [Perez & Carbonell 94]).

- *"caching" und Planadaption*
  Zur Verbesserung der Laufzeit deduktiver Verfahren, die auch zum Planen genutzt werden können, gibt es Ansätze, bereits gefundene Teillösungen oder auch Fehlschläge temporär zu speichern und bei der weiteren Problemlösung zu verwenden [Elkan 89, Segre & Scharstein 93] (vgl. auch [Calistri-Yeh & Segre 94a], wo eine innovative Planungssystemarchitektur vorgestellt wird, die u.a. "caching" verwendet). Die Planadaption, d.h. die Modifikation oder die Reparatur eines alten Planes, so daß er ein neues Problem löst, kann auch einen Beitrag zur Effizienzsteigerung bei der Suche leisten. In [Hanks & Weld 95] gibt es einen Überblick über verschiedene planadaptierende Systeme. Der Einsatz von Wiederverwendungstechniken im Umfeld des deduktiven Planens wird in [Koehler 94] beschrieben.
  Es gibt weiterhin Ansätze, einen bereits erstellten Plan dadurch zu verbessern, daß er in einen kürzeren *transformiert* wird, der das gleiche Ziel erreicht. In [Nau et al. 90a, Nau et al. 90b] ist solch ein Ansatz für Domänen, in denen nur bestimmte Interaktionen zwischen Aktionen betrachtet werden, beschrieben; er basiert auf dem Mischen von Teilplänen zu einem Gesamtplan. In [Foulser et al. 91] wird eine ähnliche Technik verwendet, wobei versucht wird, Gruppen von Aktionen durch eine einzige Aktion mit dem gleichen Effekt zu ersetzen, indem man die Ordnung bestimmter Aktionen verändert. Eine andere Idee ist, Operatoren in einem Plan zu gruppieren, die bzgl. gemeinsamer Vorbedingungen und Effekte überlappen; dadurch werden möglicherweise andere Operatoren überflüssig (vgl. [Hayes 89]).

- *Reaktivität*
  Zunehmende Beachtung erlangen aktuell Untersuchungen, die einen integrativen Planungsansatz verfolgen, um insbesondere unter Zeitbeschränkungen, großer Problemquantität oder in dynamischen Umgebungen zu brauchbaren Ergebnissen zu kommen.

  Vielversprechend ist die Möglichkeit, Planen und *automatische Kontrolle* zu integrieren [Dean & Wellman 91], um damit ein Planungssystem zu erlangen, das sowohl *überlegen* kann, um zu Entscheidungen zu kommen – also traditionell zu planen –, als auch in bestimmten Situationen sofort *reaktiv* eine geeignete Aktion ausführen kann [Schoppers 87, Drummond 89].[26] Eine Verwendung dieser Technik z.B. zur Simulation von LKW-Depots wird in [Lee et al. 93] beschrieben. Die Kombination

[25] EBL steht für *explanation-based learning.*

[26] Es gibt in der Literatur den Begriff *deliberative planner* im Gegensatz zu *reactive planner.* Reaktives

von Planen und reaktivem Verhalten kann im Sinne eines *Anytime*-Algorithmus [Boddy & Dean 89] verstanden werden. Wenn eine starke Zeitbeschränkung zur Lösung einer Planungsaufgabe vorhanden ist oder auf eine früher erstellte Lösung zurückgegriffen werden kann, dann wird in vorprogrammierter Weise gehandelt. Wenn mehr Zeit vorhanden ist oder kein Plan wiederverwendet werden kann, dann wird *normal* geplant. In [Blythe & Reilly 93] wird dies zur Simulation eines Haushaltsroboters angewendet.

Eine andere Möglichkeit, Planen und Reaktivität zu verbinden, wird in einem Ansatz von Gervasio und DeJong [Gervasio & DeJong 92] beschrieben. Das Ergebnis des Planens ist dabei kein Plan im klassischen Sinne, also eine Menge von Aktionen mit mehr oder weniger eingeschränkter Ordnung zwischen den einzelnen Aktionen, sondern eine *Zielstruktur* mit der Eigenschaft, daß es zu dieser Struktur mindestens eine Konkretisierung gibt, d.h. Aktionen, die die *aufgeschobenen* Ziele erfüllen. Die Auswahl einer konkreten Aktion bzw. die Ordnung zwischen Aktionen bleibt einer reaktiven Komponente überlassen, die im konkreten Zeitpunkt der Planausführung flexibler entscheiden kann, was eine geeignete Aktion ist. Eine erfolgreiche Anwendung findet der Ansatz bei der Integration von Planen und *Scheduling*, insbesondere bei der Berücksichtigung von Unsicherheiten.

Unter der Berücksichtigung deduktiver Planungsverfahren wurden bisher nur sehr wenige Untersuchungen veröffentlicht, die sich mit dem Problem der Planqualität beschäftigen. Im ALPS-Projekt [Calistri-Yeh & Segre 94a] wird eine adaptive Inferenzmaschine basierend auf Resolution für deduktives Planen verwendet. Für die Erzeugung optimierter Pläne unter der zusätzlichen Vorgabe der Erreichung eines Anytime-Verhaltens benutzt man eine Technik, die sich "*Iterative Strengthening*" (vgl. [Calistri-Yeh & Segre 96]) nennt. Dabei wird zunächst eine unbeschränkte Suche nach irgendeiner Lösung durchgeführt. Danach erfolgt eine erneute Suche, wobei neue Lösungen besser als die bereits gefundenen bzgl. eines definierbaren Qualitätskriteriums sein müssen.

Bezüglich der bereits oben beschriebenen Beziehung zwischen Plänen und Programmen sind Beiträge zur Programmoptimierung mit Hilfe deduktiver Methoden erwähnenswert. Madden [Madden 92] transformiert den Synthesebeweis eines Originalprogrammes in einen anderen, der mit einem Programm korrespondiert, das von der Berechnung her effizienter ist. Die Transformation wird automatisch durch Zusatzinformationen gesteuert, die zwar im Synthesebeweis enthalten sind, aber nicht im extrahierten Programm. Eine neuere Untersuchung zum Ansatz der Beweistransformation zur Programmoptimierung findet sich in [Anderson 94b].

Zilberstein [Zilberstein 95] weist darauf hin, daß die Nützlichkeit eines Planungssystemes (und damit der von ihm generierten Pläne) nur evaluiert werden kann, wenn man es als Bestandteil eines intelligenten Systems betrachtet. Der Planer wird als eine Informationsquelle angesehen, die von einer Ausführungsarchitektur verwendet wird, um Aktionen zu selektieren. Bewertet wird ein Planer danach, welchen Effekt er bzgl. der Performanz des operationalen Systems hat. Planen wird als Schlußfolgerungsaktivität angesehen, die einer Ausführungsarchitektur nützliche Information zur Verfügung stellt, wie Aktionen

Planen nimmt eine extreme Position des resourcenbeschränkten Planens ein: die optimale Aktion für jede mögliche Situation wird im voraus berechnet.

intelligent selektiert werden sollten. Unter diesem allgemeinen Gesichtspunkt ist ein Plan nicht mehr länger nur beschränkt auf eine Ansammlung von Basisaktionen. Planen als kontinuierlicher Zielverfeinerungsprozeß liefert als Ergebnis eher eine Menge von operationalen, kurzfristig und offensichtlich erreichbaren Zielen. Der Ausführungsarchitektur obliegt dann, aufgrund der ihr übermittelten Information, tatsächlich eine Entscheidung zu treffen, welche Aktion ausgeführt wird.
In [Dengler 94a] wird ein Planungssystem aus der Sicht seiner Integration in ein intelligentes Assistenzsystem betrachtet. Neben dem Benutzer können hier nun auch noch andere Komponenten des Gesamtsystems als Plankonsumenten auftreten, etwa eine Planerkennungskomponente oder ein benutzerunabhängiges Planausführungssystem. Die Anforderungen an die zu generierenden Pläne, ihr Detaillierungs- bzw. Abstraktionsgrad, die verwendete Verfeinerungsstrategie, usw., können je nach Plankonsument ganz unterschiedlich sein. Es wird das Modell eines adaptierbaren deduktiven Planers vorgestellt, der den erhöhten Anforderungen Rechnung trägt.

Kapitel 3

Eine formale Betrachtung von Planervarianten

Wie in Kapitel 1 erläutert, existiert eine große Variabilität von Planungsaufgaben in der Domäne intelligenter Assistenzsysteme. Der jeweilig geforderte qualitative Unterschied in den zu generierenden Plänen erfordert hohe Flexibilität im Planungsansatz. Die Effizienz und Transparenz des Planers, der zur Lösung einer spezifischen Aufgabe eingesetzt wird, sollten unter dieser Flexibilität nicht leiden. Der verfolgte Planungsansatz besteht deshalb darin, aus einem generischen Planer durch die Angabe von Merkmalen, die die gewünschten Ergebnisse beschreiben, einen spezialisierten Planer zu konfigurieren. Die entscheidenden Einflußgrößen zur Erreichung unterschiedlicher Planungsergebnisse sind das Wissen über die Entitäten der Planungsdomäne und am entscheidendsten die Kontrollstrategie, die beschreibt, wie man dieses Wissen einsetzt, um zu den gewünschten Plänen zu kommen. Abbildung 3.1 zeigt die allgemeine Struktur des verfolgten Planungsansatzes.
Den Kern bildet eine generische Kontrollstrategie, die zu unterschiedlichen Varianten konkretisiert werden kann. Die Variantenbildung erfolgt dabei zum einen unter Berücksichtigung von unterschiedlichem Planungswissen, das in jeweils unterschiedlichen Verarbeitungsmethoden resultiert. Zum anderen können unterschiedliche Sichten auf die Planungsdomäne in Form von unterschiedlichen Zugriffsmethoden als Variantenmerkmale einfließen. Die entstehende Kontrollstrategievariante bildet dann einen spezialisierten Planer, der aus Problemspezifikationen konsumentengerechte Pläne erzeugt.
Um den angesprochenen Prozeß aussagekräftig beschreiben zu können, wird nun ein formaler Ansatz zum Umgang mit Planervarianten vorgestellt. Zunächst wird dazu das Wissen über die Anwendungsdomäne des Planers beschrieben. Relevante Domänen sind dadurch charakterisiert, daß unterschiedliche Plankonsumenten unterschiedliche Anforderungen an den Planer stellen (vgl. Abschnitt 1.1.2).

3.1 Strukturierung des Konsumenten- und Domänenwissens

Die wesentlichen Wissenseinheiten für den heterogenen Einsatz eines Planers in einer interaktiven Anwendungsdomäne, in der sich Plankonsumenten mit unterschiedlichem Wissen,

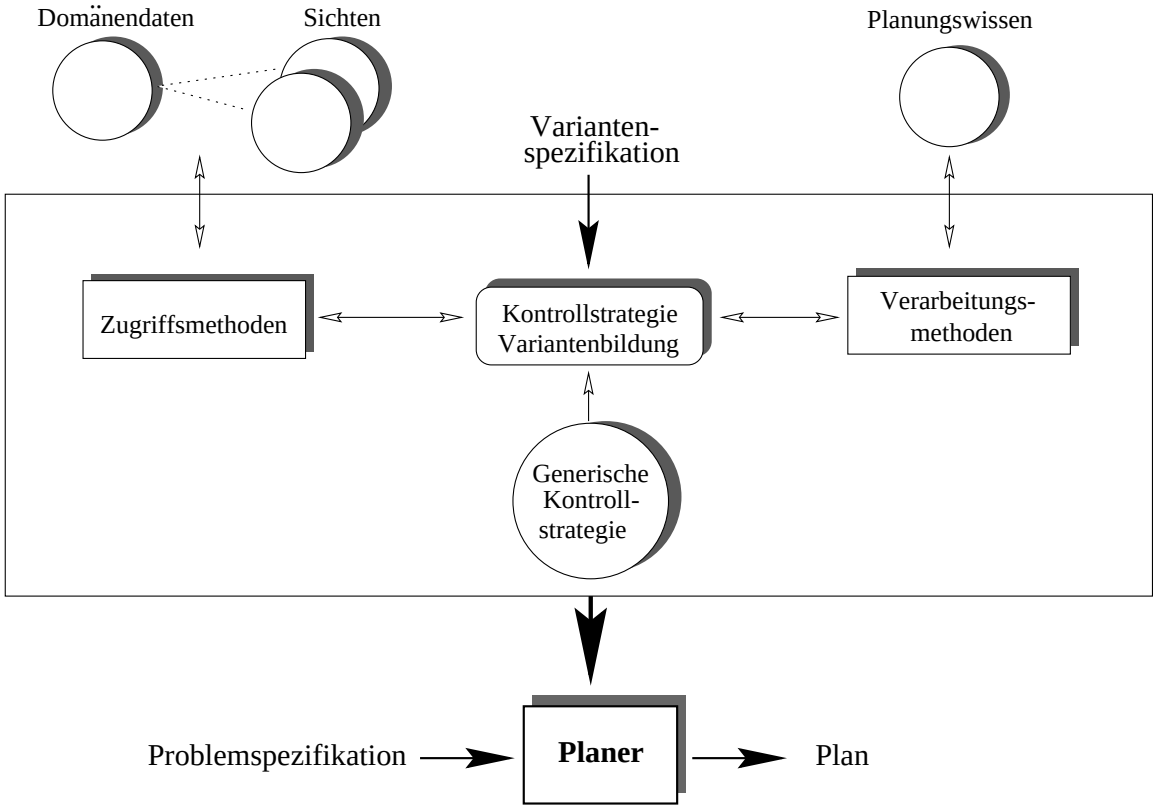


Abbildung 3.1: Grobe Skizze des Planungsansatzes

unterschiedlichen Fähigkeiten und mit unterschiedlichen Aufgaben bewegen, werden nun charakterisiert.

3.1.1 Plankonsumenten und ihre Modellierung

Nach einer Analyse der unterschiedlichen Beziehungen zwischen einem Planer und verschiedenen Plankonsumenten werden Annahmen und Modellierungsanforderungen insbesondere bzgl. des Plankonsumenten "Benutzer" dargestellt.

3.1.1.1 Die Rollen eines Planers in einem Assistenzsystem

Der mögliche strukturelle Aufbau eines Assistenzsystemes, wie ihn bereits Abbildung 1.3 zeigte, verdeutlicht, daß unterschiedliche Kommunikationskanäle unter der Beteiligung von Plankonsumenten in einem IAS existieren. Im folgenden werden die wesentlichen Beziehungen einzelner Kommunikationsteilnehmer charakterisiert.

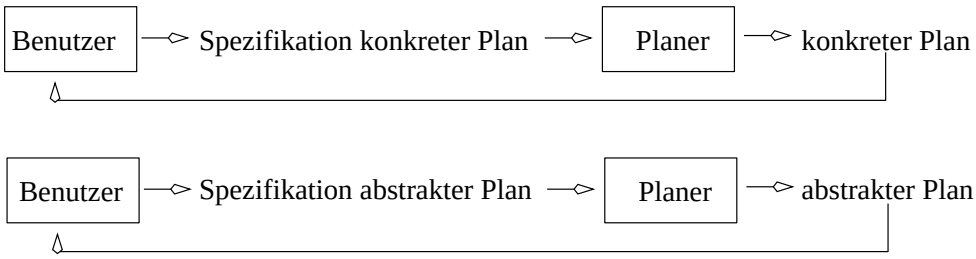
Beziehung "Planer – Benutzer"

1. *Planer simuliert Benutzer*
   Der Planer erzeugt Pläne, die dem Wissensstand des Benutzers entsprechen, damit sie für ihn leicht verständlich sind. Es werden Aktionen verwendet, die ihm bekannt sind und seine Präferenzen bzgl. der Aktionsauswahl werden berücksichtigt. Weiterhin werden nur ihm bekannte Kontrollstrukturen verwendet, und sein früheres

Handlungsverhalten wird berücksichtigt, z.B. in welcher Reihenfolge er domänenspezifisch unabhängige Ziele bearbeitet.

Der Plan berücksichtigt die aktuelle Situation, d.h. er enthält möglicherweise zusätzliche informationsspendende Aktionen, die es dem Benutzer erlauben, den Plan zu verstehen und selbst auszuführen; das unvollständige Wissen des Benutzers über den aktuellen Systemzustand wird berücksichtigt (für konkrete Pläne notwendig).

Die Basis für die Entscheidung, ob ein konkreter oder abstrakter Plan generiert werden muß, ergibt sich aus der syntaktischen Form der Spezifikation.



Basis für die Generierung ist nur das Benutzerwissen, d.h. das Benutzermodell des Systems. Möglicherweise ist auf dieser Grundlage keine Generierung erfolgreich durchführbar.

2. *Planer berücksichtigt das Wissen über den Benutzer nur partiell*
   Es gibt Einschränkungen bzgl. der Verwendung bestimmter Teile des Benutzermodells; die fehlenden Wissensquellen werden durch entsprechende Teile des Systemwissens ersetzt. Es ergeben sich z.B. folgende Optionen:

   (a) Keine Verwendung des benutzerspezifischen Problemlösungswissens.

   (b) Keine Berücksichtigung früheren Benutzerverhaltens.

   (c) Keine ausschließliche Verwendung des Aktions- bzw. Kontrollstrukturwissens; bei Bedarf wird dieses Wissen ergänzt.

   (d) Keine Verwendung des Aktions- bzw. Kontrollstrukturwissens.

3. *Planer arbeitet in Kooperation mit dem Benutzer*

   Es besteht die Möglichkeit zum interaktiven Planen:

   (a) Wenn dem Planer Information für die aktuelle Auswahl zwischen Operatoren fehlt (z.B. fehlende Indizierung der aktuellen Situation im Benutzerprofil), wird eine Entscheidung vom Benutzer gefordert.

   (b) Wenn mehrere Lösungswege existieren, kann der Benutzer in den Entscheidungsprozeß einbezogen werden. Der Planer präsentiert die Alternativen und zeigt Konsequenzen auf (durch temporale Projektion); der Benutzer wählt eine Alternative aus.

Es gibt in der Beziehung zwischen Planer und Benutzer mögliche Aufgaben, die nicht als Planungsaufgaben im klassischen Sinne anzusehen sind, jedoch können sie von einem

flexiblen Planer miterledigt werden. Als Benutzer kommt in diesem Fall nicht nur der typische Endbenutzer eines Anwendungssystems in Frage, sondern auch der IAS-Ingenieur, der etwa eine Planbibliothek erstellt bzw. wartet.

1. *Planer verifiziert Benutzervorschläge*
   Der Benutzer kann das IAS verwenden, um die Korrektheit selbst erstellter oder von Dritten erhaltener Pläne zu überprüfen. Der Benutzer gibt dabei eine Spezifikation und einen Plan vor. Der Planer verifiziert dann, ob dieser Plan die vorgegebene Spezifikation erfüllt.

2. *Planer validiert Benutzervorschläge*
   Eine ähnliche Aufgabe ist die Überprüfung der Anwendbarkeit eines vom Benutzer vorgegebenen Planes in einer aktuellen Situation.

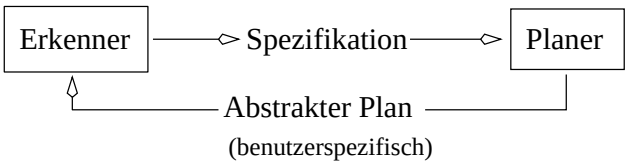
Beziehung "Planinterpreter – Benutzer"

Planer interpretiert vorgegebene abstrakte Pläne
Der Planer erzeugt unter Anwendung der Interpretationsrelationen für Abstraktionen und unter Berücksichtigung des Benutzermodells aus einem abstrakten Plan einen konkreten Plan. Offene Entscheidungen an Adaptionspunkten innerhalb eines Planes können nichtdeterministisch, situationsabhängig unter Berücksichtigung jeweiliger Konsequenzen oder in Kooperation mit dem Benutzer entschieden werden.

Beziehung "Planer – Erkenner"

1. *Planer liefert vollständige Pläne*

   Der Planerkenner benötigt für die Beobachtung und Interpretation des Benutzerverhaltens Hypothesen in Form von Plänen. Abstrakte Pläne haben den Vorteil, daß sie eine Menge konkreter Pläne in kompakter Form vereinigen. Dies reduziert den Suchaufwand des Planerkenners.

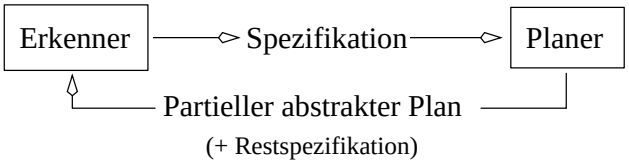


   Vollständig sind die Pläne in dem Sinne, daß sie alle Ziele einer gegebenen Planspezifikation erfüllen. Die Hypothese des Planerkenners beschreibt damit ein zusammenhängendes Verhalten des Benutzers. Im allgemeinen wird der Planerkenner mehrere Hypothesen verfolgen.

2. *Planer liefert partielle Pläne*

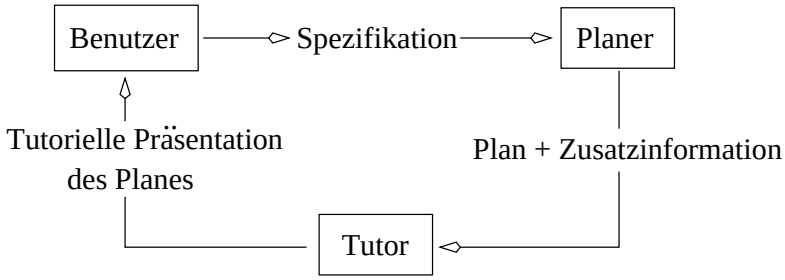
   Ist der Planerkenner in der Lage mehrere Hypothesen gleichzeitig zu verfolgen, kann es effektiver sein, wenn der Planer nur partielle Pläne als Hypothesen zur Verfügung stellt. Partiell müssen die Pläne in dem Sinne sein, daß sie nur ein (zeitlich gesehenes) Anfangsstück eines Planes bezogen auf die Spezifikation beschreiben; der minimale partielle Plan würde z.B. nur die unmittelbar nächste Aktion voraussagen. Der

Planerkenner kann dann nach einem Erkennungszyklus entscheiden, welcher partielle Plan inkrementell erweitert werden soll.



Beziehung "Planer – Tutor – Benutzer"

Betrachten wir den Fall, daß für einen bestimmten Benutzer ein Plan bzgl. einer vorgegebenen Spezifikation generiert werden soll. Wenn man nun versucht, Pläne unter Berücksichtigung des Benutzermodells der Planungsdomäne zu erstellen, kann sich die Situation einstellen, daß unter der ausschließlichen Verwendung dieses Wissens zur gegebenen Planspezifikation kein Plan gefunden werden kann.
Ein Ausweg besteht darin, das Wissen des Benutzer zu erweitern. Man verwendet zusätzlich gemäß dem Benutzermodell unbekanntes Wissen, um einen Plan generieren zu können. Dieses Wissen kann dann dem Benutzer tutoriell aufbereitet vermittelt werden. Der Planer muß also in der Lage sein, über die Verwendung von Planungs- und Domänenwissen Buch zu führen. Weiterhin muß der Planer in der Lage sein, eine Protokollierung seines Vorgehens zur Verfügung zu stellen, aus der eine Erklärung für sein Verhalten ableitbar ist.



1. *Verschiedene Arten der Wissensstanderweiterung*

   (a) Die "sanfte" Methode: Grundlage für das Planen ist das Benutzermodell. Es wird nur soviel an neuem Wissen hinzugenommen, wie nötig ist, um einen Plan generieren zu können.

   (b) Die "rücksichtslose" Methode: das Planen erfolgt auf der Grundlage des Systemwissens. Nach dem Planen wird das verwendete Wissen mit dem Benutzermodell abgeglichen. Dabei auftretende Abweichungen werden registriert.

2. *Tutoriell ausgerichtete Pläne*

   Der Detaillierungsgrad eines Planes kann das Verstehen dieses Planes beim Benutzer beeinflussen.

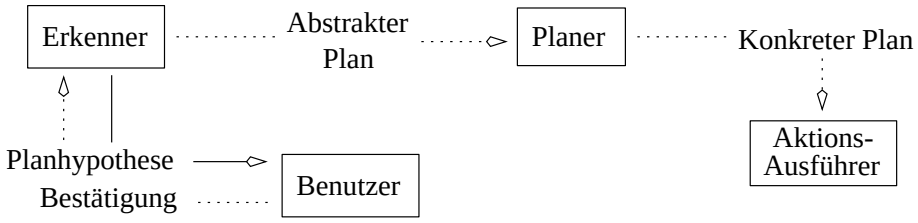
   (a) Plan enthält Abstraktionen
   Abstraktionen erlauben zum einen eine kompaktere Darstellung des Planes

(z.B. die Verwendung eines Zieles anstelle der Disjunktion aller es erreichenden Aktionen) und zum anderen veranschaulichen sie seine impliziten Zusammenhänge (z.B. die Unabhängigkeit bestimmter Aktionen ausgedrückt durch Nichtlinearität).

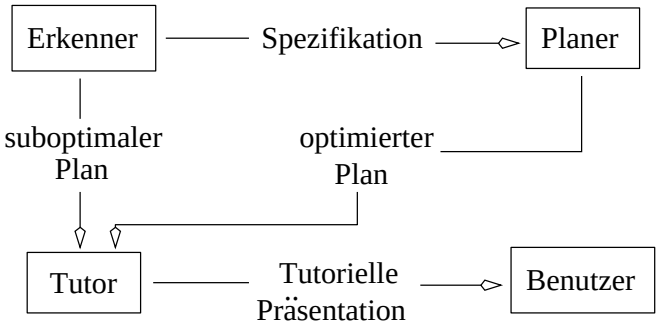
(b) Abstraktionsgrad ändert sich mit dem Planablauf
Der Abstraktionsgrad des Planes nimmt gegen Ende des Planes zu; der Beginn des Planes ist detailliert bis hin zu primitiven Aktionen. Ein Ratschlag für den Benutzer in Form eines solchen Planes kann leichter zu verstehen sein, er ist auch weniger empfindlich gegen Änderungen während der Planausführung. Details des Planendes können z.B. später durch den Benutzer vom Planer angefordert werden (durch Interpretation des abstrakten Endstückes). Dieses zeitlich verzögerte Planen kann ein sich ständig änderndes Benutzermodell genauer berücksichtigen.

Beziehung "Planer – Erkenner – Benutzer – Aktionsausführer"

Die Verwaltung der Hypothesen über mögliches Benutzerverhalten durch den Planerkenner kann dazu führen, daß eine Hypothese eine hohe Wahrscheinlichkeit erhält, daß sie das aktuelle Benutzerverhalten simuliert. Dem Benutzer kann diese Hypothese präsentiert werden mit dem Ziel, bei Akzeptanz die verbleibenden Aktionen des Planes automatisch auszuführen. Der Planer muß den nur partiell konkreten, abstrakten Plan interpretieren, um einen konkreten Plan für die automatische Aktionsausführung zu erstellen. Dabei besteht die Möglichkeit, den konkreten Plan auf der Grundlage des Benutzerwissens zu generieren, oder das gesamte Systemwissen zu verwenden.



Es kann nun aber auch die Möglichkeit bestehen, daß die präferierte Hypothese suboptimales Aktionsverhalten darstellt. In diesem Fall könnte der Planer versuchen bzgl. der Spezifikation des noch nicht ausgeführten Teiles der Hypothese einen optimierten Plan zu generieren.
In diesem Zusammenhang ist es auch möglich, daß das bisher beobachtete Benutzerverhalten suboptimales Aktionsverhalten darstellt und als solches vom Planerkenner klassifiziert wird. Dann kann bzgl. der zugehörigen Spezifikation ein optimierter Plan erstellt werden, der dann zusammen mit seinem suboptimalen Pendant dem Benutzer tutoriell präsentiert werden kann.



Beziehung "Planer – Benutzer – Aktionsausführer (– Tutor)"

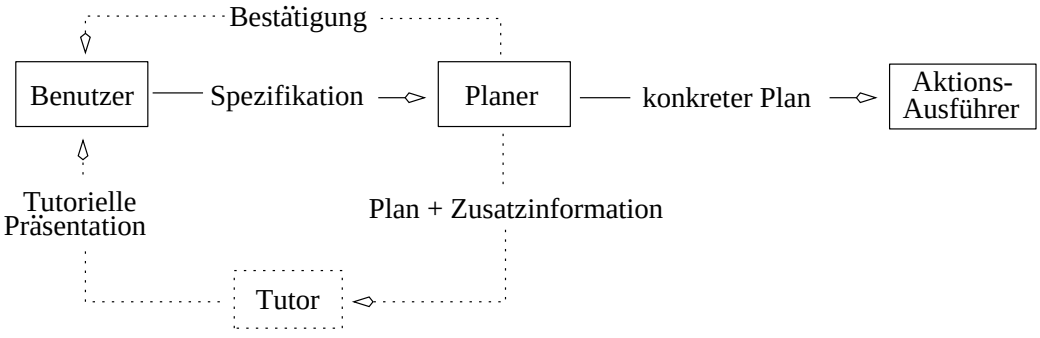
Der Benutzer initiiert die Lösung einer Planungsaufgabe durch sein Verlangen nach der Erreichung eines von ihm spezifizierten Zieles. Der Planer erzeugt einen Plan, der von dem Aktionsausführer verarbeitet werden kann. Man kann zwei mögliche Ausführungsmodi unterscheiden:

1. Der Benutzer bekommt nur eine Bestätigungsinformation über Erfolg oder Mißerfolg des Vorganges. Es kann dann ein optimierter Plan bzgl. des gesamten Systemwissens erstellt werden.

2. Der Benutzer erhält ein Protokoll der Planausführung zu dem Zweck, daß er den Planablauf nachvollziehen möchte. In diesem Fall muß das Benutzerwissen bei der Generierung berücksichtigt werden.

   (a) Die Generierung beschränkt sich auf die ausschließliche Verwendung des Benutzerwissens, benutzt lediglich geeignete Optimierungsstrategien.

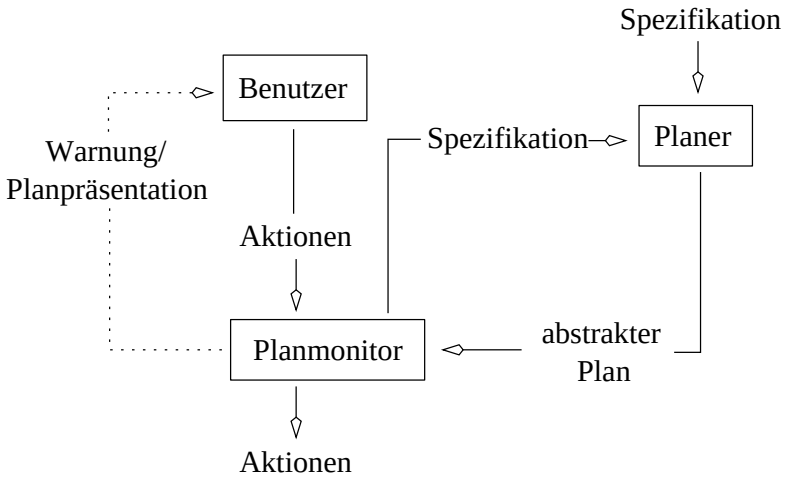
   (b) Die Generierung läßt auch eine Wissensstanderweiterung beim Benutzer zu. Dies führt dann gegebenenfalls zur Einbeziehung des Tutors.



Beziehung "Planer – Planmonitor – Benutzer"

Die Ausführung eines spezifischen Planes durch den Benutzer wird überwacht. Grundlage für die Überwachung ist ein Plan mit besonderem Abstraktionsgrad. Er abstrahiert sowohl von der Ausführungsreihenfolge bestimmter Aktionen, als auch von der Ausschließlichkeit ihres Vorkommens, d.h. es sind auch Unterbrechungen des Planablaufs durch das opportunistische Einschieben neuer Pläne erlaubt. Aus Sicht der Benutzerunterstützung kann es wünschenswert sein, die Unterbrechungen bzgl. ihrer Wirkung auf die Ausführbarkeit des restlichen Planes zu untersuchen. Es könnte passieren, daß die Effekte des eingeschobenen

Planes die Bedingungen der schadenlosen Unterbrechung des ursprünglichen Planes verletzen, etwa indem notwendige Vorbedingungen für den Restplan nicht mehr gelten. Wird solch ein Fall erkannt, kann nach einem Plan gesucht werden, der diese Vorbedingungen wieder herstellen könnte; es erfolgt also eine Generierungsaufforderung an den Planer. Mit dem jeweiligen Ergebnis kann eine entsprechende Interaktion mit dem Benutzer eingeleitet werden.



Die Vielfalt der dargestellten Beziehungen zwischen Plangenerierung und Plankonsumierung und den damit einhergehenden unterschiedlichen Merkmalen der erzeugten bzw. verwendeten Pläne bildet den konzeptionellen Ausgangspunkt für die Modellierung von Variantenspezifikationen zur Adaptierung der generischen Planungsstrategie (vgl. Abbildung 3.1).

3.1.1.2 Der Benutzer als Plankonsument

Der Plankonsument in unserer Betrachtung, der eindeutig das vielschichtigste Profil aufweist, ist der menschliche Benutzer des Anwendungssystems, mit dem das intelligente Assistenzsystem interagiert. Es werden deshalb vor allem benutzerspezifische Aspekte herausgestellt, die unmittelbar auf den Planungsprozeß und die gewünschten Ergebnisse Einfluß nehmen. Die Methoden, wie man bestimmtes Benutzerwissen akquiriert, modelliert und repräsentiert, sind im Rahmen dieser Arbeit nicht von entscheidender Bedeutung. Vielmehr ist nur der Informationsgehalt des Benutzermodelles, der für das Planen genutzt werden kann von Interesse. d.h. bestimmte Information steht bereit und kann beim Planen genutzt werden. Das Profil nichtmenschlicher Plankonsumenten kann jeweils als Spezialfall des allgemeinen Benutzerprofils angesehen werden.

Annahmen über die Benutzermodellierung Wir nehmen an, daß ein *individuelles* Benutzermodell, das *dynamisch* angepaßt wird und auf *langzeitigen* Beobachtungen beruht, Information über einen spezifischen Benutzer, der als Plankonsument auftreten kann, verfügbar macht (vgl. etwa die Klassifikation in [McTear 93] bzw. Abschnitt 2.4). Unter tutoriellen Gesichtspunkten erfolgt die Annahme einer *expertenbasierten* Modellierung domänenspezifischer Information, die dem Benutzer hier keine den tatsächlichen Gegebenheiten widersprechenden Konzepte unterstellt.

Die folgenden Arten von Information über einen Benutzer werden als verfügbar angenommen:

- Wissen über *Konzepte* der Anwendungsdomäne, d.h. welche Kontrollstrukturen, Abstraktionen, Aktionen und Beziehungen zwischen Aktionen (negative sowie positive Beeinflussungen) werden als bekannt angenommen.

- *Situationsgebundenes* Verhalten
  Hier erfolgt die Berücksichtigung der Verwendung bestimmter Verhaltensmuster des Benutzers in einer spezifischen Situation; die Situation ist gegeben durch ein zu lösendes Problem, sowie domänenspezifische und benutzerindividuelle Merkmale.

- *Fähigkeiten* zum strukturierten Vorgehen
  Wissen über die zeitliche und kausale Anordnung von Teilzielen eines übergeordneten Planungszieles.

- *Erweitertes Benutzerprofil*
  Benutzerspezifische nicht-domäneninhärente Information (z.B. Bewertungen) über Domänenkonzepte.

- Annahmen über allgemeine und situationsspezifische *Präferenzen* bzgl. der Auswahl von Aktionen, Kontrollstrukturen und Abstraktionen, Verhaltensmuster und Fähigkeiten.

Andere Plankonsumenten

Neben Benutzern gibt es auch noch andere Plankonsumenten (vgl. Abschnitt 1.1.2). In unserer Betrachtung sind dies andere Systemkomponenten, für die es dann ein spezielles *Konsumentenmodell* gibt. Bezogen auf die oben genannten Informationsarten für die Charakterisierung eines Benutzers werden die übrigen Konsumenten lediglich durch ihr jeweiliges Wissen über Aktionen, Abstraktionen und Kontrollstrukturen charakterisiert.[1]

Die Berücksichtigung von Präferenzen

Wenn Pläne für einen bestimmten Benutzer generiert werden, und das Ziel dabei ist, für den Benutzer möglichst verständliche Pläne zu erzeugen, die seinem (wahrscheinlichen) Wissensstand entsprechen, dann sollten allgemeine sowie situationsspezifische Präferenzen des Benutzers bzgl. der Verwendung seines Wissens berücksichtigt werden können. Es kann auch eine Anforderung an die Plangenerierung formuliert werden, Pläne zu generieren, die das Verhalten des Benutzers simulieren, z.B. um mit diesen Plänen Voraussagen über das zukünftige Verhalten des Benutzers zu machen oder sein tatsächliches Verhalten auf bestimmte Ziele abzubilden.
Es werden nur allgemeine Anforderungen an einen Formalismus, der Präferenzen repräsentiert und sie verarbeitet, gestellt. Welcher konkrete Formalismus dann jeweils verwendet wird, ist für unsere Betrachtung nicht weiter von Belang. Der Formalismus muß folgendes leisten:

[1] Im allgemeinen werden diesen Konsumenten alle Basisaktionen bekannt sein und es wird Einschränkungen in den verarbeitbaren Abstraktionen und Kontrollstrukturen geben. Andere strukturelle Aspekte von Plänen spielen meist keine Rolle.

- Grundlage ist eine Menge von Alternativen. Es ist möglich, eine Rangfolge zwischen Elementen der Alternativenmenge festzulegen, unter deren Berücksichtigung eine vollständige Ordnung der Alternativenmenge definiert ist, d.h. eine deterministische Zugriffsfunktion wird angeboten, die *Backtrack*-fähig ist.

Eine formale Logik in Form einer (zunächst nicht näher spezifizierten) logischen Sprache $\mathcal{L}$ bildet die Basis für die Repräsentation der Planungsdomäne, der Planungsaufgaben und schließlich der Pläne. Zunächst werden die Basiseinheiten eines jeden Planes untersucht, die Basisaktionen.

3.1.2 Basisaktionen

Die Menge der verfügbaren Basisaktionen der Planungsdomäne ist durch eine Menge von Axiomen beschrieben; dabei beschreibt ein Axiom eine abstrakte Basisaktion. Abstrakt wird im Sinne einer Objektabstraktion verstanden. Es wird angenommen, daß Basisaktionen mit Objektreferenzen parametrisiert werden können. Die Sprache $\mathcal{L}$ wird als eine Sprache erster Ordnung gewählt.
Eine Aktion ist in ihrer Axiomatisierung charakterisiert durch Vorbedingungen, d.h. Bedingungen, deren Gültigkeit das Ausführen der Aktion ermöglichen, und Effekte, d.h. Bedingungen, die nach Ausführung der Aktion gelten.[2]
Die Menge der zulässigen Aktionen einer Anwendungsdomäne kann folgende aus Planungssicht interessante Eigenschaften zeigen:

- Eine Aktion kann unter verschiedenen Bedingungen, verschiedene Effekte herbeiführen, d.h. sie impliziert *bedingte Effekte*. Dabei kann es sich um kummulierende Effekte handeln – aufgrund einer zusätzlichen Vorbedingung folgen auch zusätzliche Effekte – oder aber auch um abweichende Effekte – die Schnittmenge der Effekte in unterschiedlichen Situationen ist echt kleiner als jeder der Einzeleffekte.

- Es gibt Aktionen, die bei gleichen Vorbedingungen genau die gleichen Effekte haben. Solche Aktionen sind dann *äquivalent* und prinzipiell gleichwertig zu verwenden. Unterschiedliche Plankonsumenten können hierbei unterschiedliche Präferenzen haben.

- Es gibt Aktionen, die gemeinsame Effekte haben. Sie können sich in ihren Vorbedingungen und ihren zusätzlich erreichbaren Effekten unterscheiden. Eine Differenzierung bzw. eine Entscheidung zwischen diesen Aktionen kann auf verschiedene Arten vollzogen bzw. gefällt werden:

  - Es wird eine wertfreie Auswahl zwischen Aktionen getroffen, die einer reinen Vorliebe eines späteren Plankonsumenten entspricht. Die Auswahl geschieht auf der Ebene der Aktionsnamen.

[2]Es gibt erweiterte Ansätze zur Aktionsrepräsentation, die zusätzlich zu Vorbedingungen und Effekten z.B. auch noch Bedingungen einbeziehen, die während der Ausführung von Aktionen nicht verletzt werden dürfen ([Sandewall & Rönnquist 86]). Solche Ansätze zugrunde zu legen, bringt für den Gegenstand unserer Untersuchung keine qualitative Veränderung.

  - Die Auswahl zwischen Aktionen geschieht auf der Ebene ihrer zusätzlichen Effekte, etwa durch Berücksichtigung der Anzahl dieser Effekte, Bewertung ihrer allgemeinen positiven/negativen Auswirkungen, usw.
  - Die Auswahl geschieht unter Berücksichtigung der qualitativen Position eines die Aktionen indizierenden Effektes (vgl. etwa die Verwendung von *Primäreffekten* zur Suchraumreduzierung beim Planen [Fink & Yang 93]).
  - Die Auswahl zwischen Aktionen geschieht durch Vergleich ihrer Vorbedingungen, etwa durch die Bewertung von deren unmittelbaren Erreichbarkeit.
  - Die Auswahl zieht sowohl Effekte als auch Vorbedingungen mit ein.

  Auch wenn die Beziehungen zwischen konkurrierenden Aktionen außer Acht bleiben, kann eine Entscheidung herbeigeführt werden:

  - Die Auswahl einer Aktion geschieht rein situationsgebunden und entspricht z.B. einem reaktiven Verhalten.

- Es gibt Domäneneigenschaften, die in Aktionen nur innerhalb deren Vorbedingungen auftreten. Solche Bedingungen charakterisieren in der Regel ein *statisches* Szenario und dienen dazu, die Anwendbarkeit von Aktionen nur in bestimmten domänenspezifischen Situationen zuzulassen. Im Gegensatz zu diesen *statischen* Eigenschaften werden die Eigenschaften, die von Aktionen verändert werden können, als *dynamische* Eigenschaften bezeichnet.[3]

- Es gibt voneinander unabhängige Aktionen.

3.1.3 Pläne

Pläne werden hier verstanden als ein Beschreibungswerkzeug, das es erlaubt, nach bestimmten Gesichtspunkten zusammenhängende Basisaktionen zu einer Einheit zusammenzufassen. Aktionen werden meist zusammengefaßt, um auszudrücken, daß mit ihrer Ausführung ein bestimmtes Ziel erreicht werden kann. Im allgemeinen gibt es viele Konstellationen von Aktionen, die ein vorgegebenes Ziel durch ihre Ausführung wahr werden lassen.[4] Es wird angenommen, daß keine zwingend notwendige parallele Ausführung von Aktionen beschrieben werden muß.
Die minimalen Anforderungen an einen Plan sind, daß er festlegt, welche Aktionen an ihm beteiligt sind, welche (notwendigen) zeitlichen Beziehungen zwischen den Aktionen gelten müssen und daß er eine Garantie für die Erreichung bestimmter Ziele abgibt. Eine weitere Anforderung kann sein, daß nicht nur notwendige sondern auch mögliche Aktionen spezifiziert werden können. Die naheliegendste Form, Aktionen zeitlich strukturiert zusammenzufassen, ist, sie in einer streng sequentiellen Struktur anzuordnen, d.h. Aktionssequenzen zu bilden.

[3] In einem realitätsnahen Roboterszenario als Planungsdomäne sind z.B. die die fixe Umgebung beschreibenden (räumlichen) Eigenschaften als statisch anzusehen.

[4] Falls das Ziel mit dem Vorrat an Aktionen überhaupt erreicht werden kann.

Präzisierung 3.1.1 *Ein Plan P mit assoziiertem Ziel Z ist dann ganz allgemein eine Beschreibung*[5] *für eine Äquivalenzklasse AA von Aktionssequenzen. Die Äquivalenzrelation R_{AA} zwischen zwei Aktionssequenzen as_1 und as_2 gilt dabei, wenn die individuelle Ausführung jeder der beiden Sequenzen unter gleichen Voraussetzungen das Ziel Z erreicht. Enthält AA nur Sequenzen, deren letzte Aktion notwendig ist, um Z zu erreichen, heißt P ein notwendiger Plan, sonst ein hinreichender Plan.*[6]

Eine solche Plansicht erlaubt es, neben einer eindeutigen Charakterisierung eines Aktionsplanes auch eine Vielzahl von Freiheitsgraden darzustellen. Dies schlägt sich dann sowohl in der Größe der Äquivalenzklasse nieder als auch in den einzelnen Unterschieden zwischen ihren Vertretern.
Kontrollstrukturen und Abstraktionen stellen nun auf der Beschreibungsebene ein Instrument dar, um die Äquivalenzklasse eines Planes zu charakterisieren. Sie beschreiben zeitliche Beziehungen und Abhängigkeiten zwischen Aktionen oder definieren zustandsbezogene Randbedingungen, die der Plan erfüllt. Die Kontrollstrukturen werden mit Hilfe von Operatoren[7] der zugrundeliegenden logischen Sprache $\mathcal{L}$ gebildet; Abstraktionen werden ebenfalls in $\mathcal{L}$ selbst beschrieben. In der Menge CSA der verfügbaren Kontrollstrukturen und Abstraktionen können zwei Teilmengen, denen eine besondere Bedeutung zukommt, definiert werden:

1. Eine Menge CSA_e von *expliziten* Kontrollstrukturen, d.h. Strukturen, die unmittelbar als Kontrollstruktur in der Anwendungsdomäne vorkommen.

2. Eine Menge CSA_a *abstrakter* Kontrollstrukturen und Abstraktionen, die in abstrakten Plänen vorkommen und für alle Plankonsumenten, die nicht nur auf direkt ausführbare Pläne fixiert sind, relevant sind.

3.1.4 Entscheidungsfunktionen

Während des Planens läuft ein ständiger Entscheidungsfindungsprozeß ab und es gibt eine Menge von Auswahlpunkten, die den entsprechenden Suchraum markieren. Diese lassen im allgemeinen mehrere alternative Entscheidungen zu, z.B. in der Frage, welche Aktion gewählt werden soll, um ein aktuelles Teilziel zu erreichen. Ein allgemeiner Ansatz, die Entscheidungsfindung zu unterstützen, ist, geeignete Entscheidungsfunktionen für Auswahlpunkte zu definieren.

Definition 3.1.1 *Entscheidungsfunktionen für Auswahlpunkte haben die folgende schematische Struktur:*

$$ef_i : Ziel \times Randbedingungen \rightarrow \langle\, Planklasse_{Ziel}\,,\ Präferenzen \rangle$$

ef_i wird folgendermaßen interpretiert: Gegeben ein aktuelles Ziel (atomar oder strukturiert) und entsprechende Randbedingungen (an die aktuelle Situation, den vorgesehenen Plankonsumenten oder die übergeordnete aktuelle Zielspezifikation) gelten, dann bildet die

[5] Die nicht zwangsweise explizit Aktionen enthalten muß.
[6] Wenn im folgenden nicht anders markiert, werden Pläne im Sinne von notwendigen Plänen angesehen.
[7] Auch abgeleiteten.

(partielle) Funktion ef_i ab auf ein Tupel bestehend aus einer Äquivalenzklasse von Plänen[8]*, die das aktuelle Ziel erreichen können, und einer assoziierten Präferenzstruktur, die eine Ordnung über der Klasse definiert und/oder Bewertungen der Elemente vornimmt.*[9] *R_i sei die zugehörige Äquivalenzrelation. Die verfügbaren Entscheidungsfunktionen werden zusammengefaßt in der Menge* $\mathbf{EF} = \{ef_i \mid i = 1, \ldots, n\}$.

Es gibt eine Funktion $choose_i$, die festlegt, wie die Elemente des Bildbereiches einer Funktion ef_i zu interpretieren sind und die die Auswahl eines Planes gewährleistet:

$$choose_i : \langle \text{ Planklasse , Präferenzen} \rangle \rightarrow Plan \ .$$

$choose_i$ ist eine nichtdeterministische Funktion, deren sukzessiver Aufruf eine Aufzählung der Elemente aus "Planklasse" gemäß den definierten Präferenzen bewirkt. Ist die Aufzählung vollständig, so ändert sich der Wert von $choose_i$ in "undefiniert". Werden die Präferenzen z.B. durch eine aufsteigende totale Ordnung dargestellt, so selektiert $choose_i$ bei geeigneter Interpretation etwa das bzgl. der Ordnung jeweils verbleibende "kleinste" Element aus der Planäquivalenzklasse.

3.1.4.1 Aktionsspezifische Präferenzen

Wenn die in Abschnitt 3.1.2 beschriebenen Eigenschaften und verschiedenen Arten von Auswahlkriterien während des Planungsprozesses genutzt werden sollen, um eine Entscheidungsfindung zu unterstützen, so legt dies nahe, daß es entsprechende Indizierungen der Aktionsmenge in Form von Entscheidungsfunktionen gibt.
Betrachten wir die folgende partielle Entscheidungsfunktion als Konkretisierung von Definition 3.1.1:

$$ef_{akt} : \text{ Atomares Ziel } \times nil \rightarrow \langle \text{ Aktionsklasse}_{akt} \ , PS_{akt} \rangle \ ,$$

wobei gilt:

- R_{akt} beschreibt dann die Relation "Auftreten von *Atomares Ziel* als ein gemeinsamer Effekt".[10]
- $\text{Aktionsklasse}_{akt}$ enthält als Elemente Basispläne, die nur aus einer expliziten Aktion bestehen.
- PS_{akt} sei irgendeine Präferenzstruktur, die die Auswahl eines Klassenrepräsentanten unterstützen kann.
- *nil* symbolisiert, daß keine Randbedingungen zu berücksichtigen sind.

Mit Hilfe von ef_{akt} kann man nun einerseits darstellen, von welchen Zielen man überhaupt weiß, wie sie durch Aktionen zu erreichen sind, und andererseits sind die entsprechenden Aktionen gruppiert.
In Abhängigkeit von der verwendeten Präferenzstruktur und einer Restriktion der Äquivalenzklasse erhält man Spezialfälle von ef_{akt}:

[8] In der allgemeinen Form, wie sie in Abschnitt 3.1.3 eingeführt wurden.

[9] Zum Beispiel eine partielle Ordnung zum Ausdruck relativer Beziehungen, oder Wahrscheinlichkeiten, die auch eine absolute Bewertung eines Kandidaten zulassen – wenn etwa die Werte unterschiedlicher Entscheidungsfunktionen miteinander verglichen werden sollen.

[10] Es können auch noch jeweils zusätzliche Effekte auftreten.

- Zur Repräsentation des Wissens über äquivalente Aktionen kann die Äquivalenzrelation eingeschränkt werden auf "Identität von Vorbedingungen und Effekten". Die Präferenzstruktur kann hier die Aufgabe haben, bestimmte Vorlieben über Klassenmitglieder auszudrücken.
  Eine andere Einschränkung von R_{akt} könnte sein, z.B. eine bestimmte Höchstgrenze zusätzlicher Effekte festzulegen.

- Varianten von ef_{akt} können gebildet werden, indem man die Intention der Präferenzstruktur verändert; sie kann z.B. ausdrücken:

  - Vorlieben des Plankonsumenten *Benutzer*;
  - eine Ordnung bzgl. der Anzahl zusätzlicher Effekte;
  - eine Ordnung bzgl. der Ausführungskosten;
  - eine Ordnung bzgl. der Komplexität von Vorbedingungen;
  - eine Ordnung bzgl. negativer Auswirkungen zusätzlicher Effekte[11].

  Man wird dann im allgemeinen jeweils unterschiedliche Präferenzstrukturen zugrundelegen.

Eine unmittelbare Erweiterung von ef_{akt} stellt die Entscheidungsfunktion ef_{makt} dar:

$$ef_{makt}: \text{ Atomare Ziele } \times nil \rightarrow \langle \text{ Aktionsklasse}_{makt} , PS_{makt} \rangle .$$

Der Definitionsbereich von ef_{makt} wird gebildet von Konjunktionen atomarer Ziele. Man kann damit einen Index auf singuläre Aktionen bilden, die bestimmte konjunktive Effekte haben. Einschränkungen sind für ef_{makt} analog zu ef_{akt} durchführbar.

Durch die zusätzliche Spezifikation von Randbedingungen können die Funktionen ef_{akt} bzw. ef_{makt} spezialisiert werden. Um etwa ausgehend von einem atomaren Ziel die positiven Auswirkungen der Verwendung einer bestimmten Aktion zur Erreichung anderer, noch offenen Ziele zu beschreiben, kann folgende Instanz einer Entscheidungsfunktion verwendet werden:

$$ef_{aktpos}: \text{ Atomares Ziel } \times \text{ Atomare Ziele } \rightarrow \langle \text{ Aktionsklasse}_{aktpos} , PS_{aktpos} \rangle .$$

"Atomare Ziele" sei eine Randbedingung an die aktuelle Planspezifikation in Form einer Konjunktion von Zielen, die noch zu erfüllen ist. Die positive Auswirkung einer Aktion aus Aktionsklasse$_{aktpos}$ ist, daß durch ihre Verwendung zur Erreichung von "Atomares Ziel" die Erreichung von "Atomare Ziele" erleichtert wird.[12]
Analog zu ef_{aktpos} kann auch eine Entscheidungsfunktion ef_{aktneg} definiert werden, die eine Zuordnung von Aktionen zu Zielen unter Berücksichtigung negativer Auswirkungen vornimmt.

Die Indizierungen des Aktionswissens werden zusammenfassend nicht nur benutzt, um benutzerrelevante, oder allgemein konsumentenrelevante, Aspekte des Aktionswissens darzustellen, sondern auch, um allgemeine Heuristiken des Planers zu unterstützen. Zu bemerken ist, daß die Wertebereiche von Entscheidungsfunktionen wie ef_{akt}, ef_{makt}, ef_{aktpos}

[11] Es könnten z.B. irreversible Effekte erreicht werden.

[12] Weil z.B. Effekte erzeugt werden, die eine nachfolgende Ausführung einer Aktion ermöglichen, um die Ziele in "Atomare Ziele" unmittelbar zu erreichen.

und ef_{aktneg} bezogen auf eine gegebene Menge von Aktionsaxiomen automatisch bestimmt werden können.

3.1.4.2 Abstraktionsspezifische Präferenzen

Ähnlich der Strukturierung der Aktionsmenge können auch die einzelnen Kontrollstruktur- bzw. Abstraktionsmengen (vgl. Abschnitt 3.1.3) restringiert bzw. mit einer Präferenzstruktur versehen werden. Die entsprechenden Entscheidungsfunktionen haben folgende Struktur:

$$ef_{CSA_*} :\rightarrow \langle \text{ KAklasse}_{CSA_*} , PS_{CSA_*} \rangle .$$

Man kann damit ausdrücken, welche Kontrollstrukturen bzw. Abstraktionen verfügbar sein sollen, und ob Präferenzen zwischen ihnen bestehen, d.h. es gibt Strukturen PS_{CSA_e} bzw. PS_{CSA_a}, wenngleich ihnen im allgemeinen eine geringere Bedeutung zukommt als den Präferenzen bzgl. der Aktionsauswahl.
Speziell aus tutorieller Sicht kann aber eine Rangfolge von Kontrollstrukturen und Abstraktionen bzgl. eines steigenden Schwierigkeitsgrades hilfreich sein.[13] Ein Beispiel für solch eine Rangfolge zeigt Abbildung 3.2. Formal entspricht eine solche Rangfolge $rang_{CSA}$ einer geordneten Teilmenge von 2^{CSA}:

$$rang_{CSA} = \langle A, < \rangle, \quad A \subset 2^{CSA}, \; < \text{ eine totale Ordnung.}$$

{Implizite Aktionsdetermination}
$\vee$
{Iteration}
$\vee$
{Unterbrechbarkeit, Nichtlinearität}
$\vee$
{Konditionale}
$\vee$
{Aktionsnichtdeterminismus, Aktionsabstraktion}
$\vee$
{Objektabstraktion}
$\vee$
{Sequentialität}

Abbildung 3.2: Beispielhafte Rangfolge von Abstraktionen

[13] Es kann z.B. notwendig sein, einen Lernenden, der Anfänger im Umgang mit interaktiver Software bzw. mit Computern ist, zunächst nicht mit zuvielen Abstraktionen bei der Präsentation von Plänen zu überfordern.

3.1.4.3 Planspezifische Präferenzen

Die vorgestellten aktionsspezifischen Präferenzstrukturen drücken lokale Entscheidungen – die Auswahl einer Aktion betreffend – als Folge des Auftretens eines bestimmten Zieles oder einer Menge von Zielen aus. Um benutzerspezifische Aspekte der Verwendung von Aktionen und Plänen in ausreichendem Maße berücksichtigen zu können, betrachtet man als Erweiterung und Verallgemeinerung der Aktionsmengenstrukturierung die Repräsentation von *typischem Planverhalten* des Benutzers (vgl. Abschnitt 3.1.1.2 "Situationsgebundenes Verhalten") als eine Form, planspezifische Präferenzen auszudrücken. Dabei bilden strukturierte Ziele – zeitliche Relationen werden berücksichtigt – in Verbindung mit Randbedingungen einen Index auf Klassen von Plänen. Diese Pläne können sowohl konkret als auch abstrakt sein.
Das typische Planverhalten kann wiederum durch eine entsprechende Instanz der allgemeinen Entscheidungsfunktion gemäß Definition 3.1.1 dargestellt werden:

$$ef_{planver} : \text{SZe} \times \text{C} \rightarrow \langle \text{PK}_{planver} , PS_{planver} \rangle ;$$

es gilt:

- SZe beschreibt strukturierte Ziele, d.h. neben einer reinen Konjunktion von Zielen können auch zeitliche Beziehungen zwischen den Zielen spezifiziert sein.
- C beschreibt Randbedingungen bzgl. der übergeordneten Planspezifikation, des aktuellen Zustandes oder des fokussierten Plankonsumenten.[14]
- $\text{PK}_{planver}$ ist eine Klasse von Plänen[15] bzgl. der allgemeinen Äquivalenzrelation "Erreichung des Zieles SZe".

Es werden nun Instanzen des Konzeptes "Typisches Planverhalten" vorgestellt:

Realisierung konjunktiver Ziele: Eine Entscheidung, die bei dem Auftreten konjunktiver Ziele zu treffen ist, betrifft die angenommene zeitliche, partielle Ordnung der bestehenden Teilziele, d.h. in welcher Reihenfolge werden die Teilziele durch entsprechende Teilpläne erreicht. Aussagekräftige Spezialfälle von $ef_{planver}$ kann man bilden, indem man für die Mitglieder von $\text{PK}_{planver}$ bestimmte Abstraktionsniveaus voraussetzt:

1. $ef_{planver^{IA}}$ – "Implizite Aktionsdetermination"
   Hier wird einem gegebenem Ziel ein neues konkreteres Ziel zugeordnet. Konkreter kann das Ziel bzgl. der zeitlichen Beziehungen einzelner Teilziele oder im Sinne einer Spezialisierung der Teilziele sein. Als Mitglied des Graphen von $ef_{planver^{IA}}$ kann man sich folgendes Tupel vorstellen:

   $$(\Diamond(g_1 \wedge g_2), nil, \langle\{p1 : \Diamond(g_2 \wedge \Diamond g_1), p2 : \Diamond(g_3;g_4 \wedge \Diamond g_1)\}, \{p2 < p1\}\rangle).$$[16]

[14] Zum Beispiel benutzerspezifische Bedingungen über Objekte, die mit einem Plan in Beziehung stehen (vgl. Abschnitt 3.1.1.2 "Erweitertes Benutzerprofil").

[15] Hier nun nicht mehr nur eingeschränkt auf Basispläne.

[16] Die Symbole $\Diamond$, $\wedge$ und ";" stellen logische Operatoren dar. Ihre genaue Bedeutung wird in Abschnitt 4.1 erklärt.

Die Interpretation dieses Eintrages lautet: Gegeben ein Ziel bestehend aus zwei Teilzielen g_1 und g_2 und es sind keine Randbedingungen spezifiziert, dann stehen zwei Pläne $p1$ bzw. $p2$ zur Verfügung, die mögliche Vorgehensweisen des Benutzers beschreiben. $p1$ führt eine zeitliche Ordnung zwischen den Teilzielen ein, während $p2$ eine Spezialisierung des Teilzieles g_2 vornimmt, es wird zerlegt in eine sequentielle Folge zweier Teilziele g_3 und g_4. Es wird angenommen, daß eine Präferenz von $p2$ gegenüber $p1$ besteht.
Man hat mit einer solchen Indizierung nun die Möglichkeit, bestimmte Fähigkeiten des Benutzers darzustellen: Welches Wissen besteht beim Benutzer über das Erkennen und Lösen kritischer Zielbeschreibungen; kritisch können sie in dem Sinne sein, daß eine spezielle Anordnung der Teilziele notwendig ist, um zu einem befriedigenden Plan[17] zu kommen.

2. $ef_{planver^{EA}}$ – "Explizite Aktionen"
Einem gegebenen Ziel wird ein konkreter Plan zugeordnet, der zur Erreichung des Zieles verwendet wird. Als Mitglied des Graphen von $ef_{planver^{EA}}$ könnte auftreten:

$$(\Diamond(g_1 \wedge g_2), nil, \langle\{p1 : ex(action_{g_{21}}); ex(action_{g_{22}}); ex(action_{g_1})\}, \{p1\}\rangle).$$

Die Interpretation lautet: Gegeben ein Ziel bestehend aus zwei Teilzielen g_1 und g_2 und es sind keine Randbedingungen spezifiziert. Ein Plan $p1$ kann verwendet werden, um das Ziel zu lösen; zunächst werden dazu zwei Aktionen zur Erfüllung von g_2 verwendet und dannach eine Aktion zur Erreichung von g_1.

Triggern hinreichender Pläne: In Abhängigkeit von bestimmten Randbedingungen können Aktionen ausgelöst werden, die nicht notwendig sind, um ein vorgegebenes Ziel zu erreichen, sondern deren Einführung exogen bedingt ist[18]. Die Entscheidungsfunktion $ef_{planver^{reak}}$ beschreibt etwa solches Verhalten. Ein Mitglied des Funktionsgraphen kann folgendermaßen aufgebaut sein:

$$(\Diamond(g(x)), \phi(x), \langle\{p1 : ex(action_g); ex(action_\phi)\}, \{p1\}\rangle).$$

Ein Ziel $g(x)$ sollte erreicht werden, wobei eine Randbedingung $\phi(x)$ gilt, die einen Spezialfall des Objektes x abfängt. Der Benutzer bevorzugt, das Ziel durch eine Aktion $action_g$ gefolgt von einer Aktion $action_\phi$ zu erreichen, die nicht notwendig ist zur Erreichung von $g(x)$, sondern ein reaktives Verhalten, ausgelöst durch die Bedingung $\phi(x)$, darstellt.

Die Einbeziehung von typischem Planverhalten des Benutzers in den Plangenerierungsprozeß dient dazu, das Verhalten des Benutzers in gewisser Weise zu simulieren. Es können Pläne erstellt werden, die für ihn verständlich sind und seinen Fähigkeiten entsprechen.

Unabhängig von typischem Planverhalten eines Benutzers können planspezifische Präferenzen aber auch dazu dienen, kontrollspezifische Heuristiken des Planers zu beschreiben, z.B. Zielordnungsheuristiken zur Unterstützung einer Strategie für den Umgang mit konjunktiven Zielen. Diese Strategie kann z.B. versuchen, bei mehreren offenen Teilzielen

[17] Oder überhaupt zu einem Plan.
[18] Sie entsprechen z.B. einer persönlichen Vorliebe des Benutzers.

solche zur nächsten Bearbeitung auszuwählen, die in der aktuellen Situation *am ehesten* erreichbar sind. Als Maß für den Abstand zwischen aktuellem Zustand und gefordertem Ziel kann man z.B. domänenspezifisch eine gewisse *Kontextübereinstimmung* wählen.[19] Ziele, die den aktuellen Kontext betreffen, sind hierbei am ehesten zu erreichen. Ist ein Zielzustand z.B. gegeben als eine Konjunktion von einzelnen Teilzielen, geschrieben als $\Diamond[g_1 \wedge g_2 \wedge \ldots \wedge g_n]$, so kann ein Element des Graphen einer entsprechenden Entscheidungsfunktion $ef_{current\_context}$ z.B. gegeben sein als

$$(\Diamond[g_1 \wedge g_2(c) \wedge \ldots \wedge g_n], \phi(c), \langle\{p1 : \Diamond[g_2(c) \wedge g_3 \wedge \ldots \wedge g_1 \wedge g_n]\}, \{p1\}\rangle).$$

Wenn wir annehmen, daß eine Planungsstrategie verfügbar ist, die, gegeben ein konjunktives Ziel $\Diamond[g_1 \wedge g_2(c) \wedge \ldots \wedge g_n]$, versucht, die einzelnen Teilziele g_i in der durch die Folge $\langle 1, 2, \ldots, n\rangle$ bestimmten Reihenfolge zu erreichen, um schließlich das Gesamtziel erreicht zu haben, dann triggert $\Diamond[g_2(c) \wedge g_3 \wedge \ldots \wedge g_1 \wedge g_n]$ ein verändertes Verhalten der Strategie. $\phi(c)$ markiere hier eine Bedingung, die den aktuellen Kontext charakterisiert; g_2 sei das einzige Teilziel, das im Kontext c zu erreichen ist. Die weitere Bearbeitung des transformierten Zieles verhindert nun einen notwendigen Kontextwechsel.

Die Verwendung der oben beschriebenen Entscheidungsfunktionen kann auch so interpretiert werden, daß sie mit den Konzepten des *fallbasierten Planens* [Hammond 89], des *reaktiven Planens* [Georgeff & Lansky 87] und des Planens durch Zuhilfenahme *hierarchischer Aufgabenreduktionen* [Erol et al. 94] in enger Beziehung stehen. Mit der Technik des fallbasierten Planens wird im Sinne des *"planning from second principles"* versucht, adäquate bereits erstellte Pläne durch eine geeignete Indizierung in neuen Problemsituationen wiederzuverwenden, gegebenenfalls nach einer Modifikation (vgl. etwa [Koehler 94]). Das heißt, bezogen auf ein gegebenes Ziel und als Randbedingung aktuell geltende Vorbedingungen erhält man durch eine geeignete Entscheidungsfunktion mit einer Rangfolge versehene Pläne oder Planschemata[20] zurück, die zur Lösung des Zieles (möglicherweise) verwendet werden können.
Betrachtet man nur den Aspekt, daß die wiederverwendeten Planschemata den noch notwendigen weiteren Verfeinerungsprozess steuern, so stellt die Technik, hierarchisch repräsentierte Aufgabenverfeinerungen zur Problemreduktion während des Planens zu verwenden, eine Verallgemeinerung dar. Die Verfeinerungen repräsentieren die vielen Anwendungsdomänen inhärent gegebene Eigenschaft, daß komplexere Aufgaben sukzessive auf bestimmte Weisen in Teilaufgaben untergliedert werden können, die letztendlich dann zu primitiven Aktionen führen.
Reaktives Planen erlaubt die situationsbedingte Auswahl einer Aktion zur Erreichung eines Zieles und verwendet dazu eine Form von *Situations-Aktions-Regeln*. Für ein gegebenes Ziel ist dabei festgelegt, mit welcher Aktion dieses Ziel in einer bestimmten Ausgangssituation erreicht werden soll. Dieses Konzept findet sich direkt in dem Entscheidungsfunktionsschema wieder.

[19] Als einen solchen Kontext kann man beispielsweise in einer Planungsumgebung, in der ein autonomer Roboter verschiedene Aufgaben in einem Raum mit abgetrennten Zonen zu erfüllen hat, *eine* solche Zone ansehen.

[20] Verhaltensmuster, die noch mit nicht näher spezifizierten Lücken versehen sind.

3.1.5 Anwenderspezifisches Domänenwissen

Das Wissen über eine Anwendungsdomäne kann man von (mindestens) zwei Gesichtspunkten aus betrachten.

1. Man beschränkt sich auf domäneninhärente Aspekte, d.h. die betrachteten Entitäten, Eigenschaften und Beziehungen haben ein direktes Abbild in der Domäne. Man betrachtet die Domäne "von Innen heraus".

2. Man betrachtet die Domäne aus der Sicht des Anwenders, desjenigen, der in der Domäne operiert. Neben den domäneninhärenten Aspekten existieren dann auch noch anwenderspezifische Aspekte, wobei indirekte Abbildungen auf Domänenkonzepte existieren. Man definiert hier etwa zusätzliche Objekteigenschaften, für die aus Domänensicht keine Notwendigkeit besteht, die aber den Umgang des Benutzers mit der Domäne erklärbar machen.

Benutzereigene *Bewertungen* von Objekten der Domäne definieren eine Art von nicht domäneninhärentem Wissen. Diese Bewertungen haben eine subjektive Bedeutung für das Verhalten des Benutzers in der Domäne, sie können z.B. bestimmte Reaktionen auslösen.

3.1.6 Wissen über Mensch-Maschine-Interaktion

Ein Punkt, dem bei einem interaktiven Anwendungssystem große Bedeutung zukommt, ist der des Informationsretrievals und der Informationspräsentation. Wenn es etwa parametrisierte Aktionen in der Anwendungsdomäne gibt, so muß der Benutzer entsprechende Parameterinstantiierungen oft auf Grund von Systemzustandsdaten vornehmen, d.h. die Aktionen sind abhängig vom jeweiligen Systemzustand und nicht nur von dem Ziel, das durch sie erreicht werden soll. Wenn Systemzustandsdaten nicht ständig für den Benutzer unmittelbar verfügbar sind[21], so gibt es zusätzliche sensorische Aktionen, die sie präsentieren können. Die Daten können, wenn sie benutzerrelevant sind, z.B. durch ein Anzeigen in einem bestimmten Fenster auf dem Bildschirm präsentiert werden. Ausgelöst wird dieser Vorgang durch eine geeignete informationsproduzierende Aktion.
Wenn man benutzerorientierte Pläne generieren möchte, so ist die Einführung solcher Aktionen oft unverzichtbar, um einen für den Benutzer verständlichen Plan zu erzeugen. Betrachten wir dazu das folgende abstrakte Beispiel:

> Die Ausgangssituation ist: ein Benutzer möchte das Ziel $goal(y)$ erreichen und ein Plan, der dies tut, soll dem Benutzer präsentiert werden.
>
> Es gebe dazu beispielsweise eine Aktion $action(x, y)$, die dieses Ziel erreicht, wobei x ein situationsabhängiger Parameter sei. Ein reines Präsentieren der Aktion $action(inst(x), y)$,[22] die das geforderte Ziel erreicht, ist für den Benutzer unbefriedigend, da er u.U. nicht weiß, wie er $inst(x)$ selbst bestimmen kann. Sinnvoller ist deshalb, zusätzlich eine informationsspendende Aktion $show_x$ mit anzugeben, die eine Bestimmung von $inst(x)$ in der aktuellen Situation erlaubt.

[21] Etwa indem sie auf dem Bildschirm sichtbar sind.
[22] $inst(x)$ ist die aktuelle Instantiierung des Parameters x.

Aus Plangenerierungssicht kann sich ein Problem bzgl. der Einführung von informationsproduzierenden Aktionen ergeben: Sie sind oft nicht notwendig, um das spezifizierte Ziel zu erreichen, da die Effekte, die sie erzeugen, möglicherweise gar nicht als Vorbedingungen für andere Aktionen gefordert sind. Um dennoch auch Pläne erzeugen zu können, die solche Aktionen enthalten, muß eine entsprechende Ansteuerung zur Generierung hinreichender Pläne erfolgen. Man muß die Lücke zwischen den Effekten der informationsproduzierenden Aktionen und den übrigen Aktionen dann durch die Verwendung entsprechend angepaßter Planverhaltensindizes (vgl. Abschnitt 3.1.4.3) schließen.

3.1.7 Kompilation von Aktionswissen

Durch eine Analyse der Aktionsaxiome ist es möglich, bestimmte Beziehungen zwischen einzelnen Zielen, die durch verschiedene Aktionen erreicht werden, zu erkennen. Für das Planen ist vorallem die Aufspürung negativer Interaktionen von Bedeutung, da ihr Auftreten während des Planens möglichst verhindert werden sollte. Betrachten wir zwei Aktionen a_1 und a_2 mit Vorbedingungen pre_1 bzw. pre_2 und Zielen $goal_1$ bzw. $goal_2$. Gilt nun

$$goal_1 \rightarrow \phi \text{ und } pre_2 \rightarrow \neg\phi^{23}$$

dann besteht eine negative Interaktion zwischen $goal_1$ und $goal_2$ bzgl. der Aktionsfolge $a_1;a_2$, d.h. die Erreichung von $goal_1$ durch die Aktion a_1 erschwert die Erreichung von Ziel $goal_2$ durch a_2.
Wenn nun die axiomatisierten Aktionen einer Anwendungsdomäne gegeben sind, so kann man für die darin vorkommenden Einzelziele eine Relation

$$erschwert(goal_a, goal_b, action_a, action_b)$$

automatisch aufbauen, die Information über negative Interaktionen zwischen Zielen, wie sie oben beschrieben wurden, widerspiegelt. In [Cheng & Irani 89] wird eine formale Repräsentation von Teilzielordnungsbedingungen angegeben. Cheng und Irani erweitern die oben genannte Relation, indem sie negative Beziehungen zwischen Mengen von Zielen betrachten.

Die Relation kann bei der Lösung von Planungsproblemen, die durch konjunktive Ziele beschrieben sind, als Heuristik zum Ordnen der einzelnen Teilziele verwendet werden. Ordnen bedeutet dabei, zu bestimmen, in welcher Reihenfolge die Teilziele erreicht werden sollen. Eine Umformung der Relation in Entscheidungsfunktionen[24], die die Umordnung von Teilzielen repräsentieren, ist offensichtlich durchführbar.
Ist ein Zielzustand z.B. gegeben als eine Konjunktion von einzelnen Teilzielen, geschrieben als $\Diamond[g_1 \wedge g_2 \wedge \ldots \wedge g_n]$ so kann ein Element des Graphen einer entsprechenden Entscheidungsfunktion $ef_{erschwert}$ z.B. gegeben sein als

$$(\Diamond[g_1 \wedge g_2 \wedge \ldots \wedge g_n], nil, \langle\{p1 : \Diamond[g_2 \wedge g_3 \wedge \ldots \wedge g_1 \wedge g_n]\}, \{p1\}\rangle).$$

Bezüglich $ef_{erschwert}$ nehmen wir an, daß für das transformierte Ziel weniger Instanzen der Relation *erschwert* existieren mit: es gilt $erschwert(g_i, g_j, *, *)$ für $i < j$. Auf diese Art

[23] "$\rightarrow$" drücke die logische Implikation aus.
[24] Wie sie auf Seite 56 ff. beschrieben sind.

und Weise kann die Entscheidungsfunktion $ef_{erschwert}$ als Zielordnungsheuristik während eines Planungsprozesses bei entsprechender Aktivierung eingesetzt werden und so u.U. verhindern, daß Konflikte während des Planens auftreten.

Als zweites Beispiel zur Umsetzung kompilierten Wissens betrachten wir die Entscheidungsfunktion $ef_{synergy}$. Ein Element ihres Graphen kann z.B. gegeben sein als

$$(\Diamond[g_1 \wedge g_2 \wedge \ldots \wedge g_n], nil, \langle\{p1 : \Diamond[g_2 \wedge g_3 \wedge g_1 \wedge g_4 \wedge \ldots \wedge g_n]\}, \{p1\}\rangle).$$

Die Intention ist, daß im transformierten Ziel Teilziele *gruppiert* sind hinsichtlich der Eigenschaft, daß es Aktionen gibt, die Konjunktionen spezifizierter Teilziele erreichen können. Beispielsweise kann es eine Aktion geben, die die Ziele g_2 und g_3 erreicht und eine, die g_1 und g_4 erreicht. Die Umordnung der einzelnen Ziele geschieht dabei unter Verwendung der aktionsspezifischen Entscheidungsfunktion ef_{aktpos} (vgl. Abschnitt 3.1.4.1).

3.2 Die Spezifikation von Planungsproblemen

Betrachten wir nun das zugrundeliegende Verständnis der Plangenerierung, d.h. einerseits wie Planungsprobleme, Planungszustände und Lösungen repräsentiert sind und andererseits wie der Prozeß des Planens zu verstehen ist. Ausgehend von einem Planungsproblem, dessen Beschreibung einen gesuchten Plan zu Vorbedingungen und Zielen in Beziehung setzt, wird durch das mit Hilfe von Transformationen bewirkte Erreichen von verschiedenen Planungszuständen eine Lösung des Planungsproblems erlangt, d.h. Planen bedeutet Suche im Raum möglicher Planungszustände. Für die Steuerung der Transformationen und damit der Übergänge zwischen Planungszuständen ist ein Planungsalgorithmus verantwortlich.
Die Basis für die Repräsentation von Vorbedingungen, Zielen und Plänen bildet eine logische Planungssprache $\mathcal{L}$, eine modale (temporale) prädikatenlogische Sprache erster Ordnung. Eine Teilsprache $\mathcal{L}_{Plan}$ charakterisiert zulässige Pläne, d.h. Pläne sind selbst Formeln in der Sprache $\mathcal{L}$. $\mathcal{L}_{NP} = \mathcal{L} \setminus \mathcal{L}_{Plan}$ bildet die Sprache der Nichtplanformeln.
Es gibt Erweiterungen $\mathcal{L}^{Mv}_{Plan}$ und $\mathcal{L}^{Mv}_{NP}$ der Sprachen $\mathcal{L}_{Plan}$ bzw. $\mathcal{L}_{NP}$, die die Signaturen dieser Sprachen um eine Menge $Mv = Mv_{Plan} \cup Mv_{NP}$ spezieller Propositionssymbole, sogenannte *Metavariablen*, erweitern. Diese sind Platzhalter für Formeln der jeweiligen Objektsprache und werden wie atomare Formeln in dieser Sprache verwendet. Aus einer Formel $\phi \in \mathcal{L}^{Mv}$ kann durch Demodulation bzgl. einer geeigneten Menge

$$\sigma = \{f_i/mv_i | i = 1, \ldots, n \;\; f_i \in \mathcal{L}^{Mv}, mv_i \in Mv\}$$

von Substitutionen eine Formel $\phi' \in \mathcal{L}$ gewonnen werden, $\phi' = demod_\sigma(\phi)$. Es ist zu bemerken, daß die Metavariablen lediglich ein Hilfsmittel darstellen, um zu einem bestimmten Zeitpunkt einfache, noch unspezifizierte Beziehungen zwischen Formeln zu repräsentieren.

Definition 3.2.1 *Ein Planungszustand S als ein definierter Zwischenzustand des verwendeten Planungsalgorithmus ist nun definiert als ein 6-Tupel*

$$\langle V, MvP, Z, MvPSubst, MvSubst, Th\rangle,$$

mit

- $V \in \mathcal{L}_{NP}^{Mv}$, *eine Formel, die aktuell geltende Vorbedingungen beschreibt;*
- $MvP \in \mathcal{L}_{Plan}^{Mv}$, *eine Planformel;*
- $Z \in \mathcal{L}_{NP}^{Mv}$, *eine Formel, die die zu erreichenden Ziele beschreibt;*
- *MvPSubst, eine Menge von Substitutionen* f_i/mv_i, *mit* $f_i \in \mathcal{L}_{Plan}^{Mv}$;
- *MvSubst, eine Menge von Substitutionen* f_i/mv_i, *mit* $f_i \in \mathcal{L}_{NP}^{Mv}$;
- *Th, die zugrundeliegende Theorie der Planungsdomäne; beschrieben sind die verfügbaren Aktionen, Gesetzmäßigkeiten und geltenden Beziehungen in der Domäne.*

Definition 3.2.2 *Ein initiales Planungsproblem ist spezifiziert durch einen Planungszustand* $S_{init} = \langle V, MvP, Z, MvPSubst, MvSubst, Th\rangle$, *wobei gilt* $V, Z \in \mathcal{L}_{NP}$, $MvP \in Mv$ *und* $MvPSubst = MvSubst = \emptyset$. *Man spricht auch von einem initialen Planungszustand.*

Definition 3.2.3 *Ein Planungsalgorithmus plangen wird als partielle Funktion über initialen Planungszuständen charakterisiert:*

$$plangen: \ M_S \rightarrow 2^{M_{Subst}},$$

M_S *die Menge der initialen Planungszustände,* M_{Subst} *die Menge möglicher Substitutionen für Elemente aus* Mv. *plangen muß die Eigenschaft haben, daß durch*

$$plangen(\langle V, MvP, Z, MvPSubst, MvSubst, Th\rangle) = MvPSubst'$$

ein Plan $P = demod_{MvPSubst'}(MvP)$, $P \in \mathcal{L}_{Plan}$ *bestimmt wird, so daß durch die Ausführung von* P *unter der Bedingung, daß die Vorbedingung* V *gilt, die Ziele* Z *garantiert erreicht werden.* P *wird Lösung des Planungsproblems genannt.*

Durch die garantierte Erreichung der Ziele Z schränkt man den Planungsalgorithmus so ein, daß nur bzgl. ihrer Spezifikation und der Theorie Th *korrekte* Pläne erzeugt werden.

Definition 3.2.4 *Die Realisierung von plangen durch Suche im Raum der zulässigen Planungszustände bezeichnen wir als Planen, bzw. Plangenerierung.*

Die Theorie Th aus Definition 3.2.1 wird als statisch betrachtet, d.h. sie wird während des Planens nicht verändert.
Die Wahl des Mechanismus, der Zustandsübergänge im Raum der Planungszustände beschreibt und sie zueinander in Beziehung setzt, hat eine wesentliche Auswirkung darauf, inwieweit Begriffe, wie *Korrektheit* des generierten Planes bzgl. seiner Spezifikation, formuliert und bewiesen werden können. Kann man die Zustandsübergänge etwa durch Inferenzen in einem logischen Kalkül beschreiben, so gewinnt man einerseits eine Semantik für den Planungsprozeß selbst und andererseits erhält man eine gesicherte Aussage darüber, was die durch den Planungsprozeß erzielbaren Lösungen mit der ursprünglichen Problemstellung zu tun haben, sie sind *beweisbar korrekte* Lösungen. Die Korrektheit des Planungsprozesses und der dadurch generierten Pläne wird automatisch sichergestellt, sie folgt direkt aus der Korrektheit des verwendeten logischen Kalküls.

Ein weiterer Begriff, der mit einem Planungsalgorithmus in Verbindung gebracht werden kann, ist der der *Vollständigkeit*. Die Frage ist, ob der Algorithmus in der Lage ist, zu allen initialen Planungszuständen, zu denen eine Lösung existiert, diese Lösung auch zu finden.

Analog zu der Charakterisierung eines Planungsproblems in Definition 3.2.2 lassen sich auch damit verwandte Probleme charakterisieren.

Definition 3.2.5 *Ein Planungszustand* $S = \langle V, P, Z, MvPSubst, MvSubst, Th \rangle$ *mit* $V, Z \in \mathcal{L}_{NP}$, V *konsistent,* $P \in \mathcal{L}_{Plan}$ *spezifiziert eine Planverifikationsaufgabe; es gilt dann* $plangen(S) = \emptyset$.

Die Aufgabe besteht dann darin, nachzuprüfen, ob durch die Ausführung von MvP unter der Bedingung V die Ziele Z erreicht werden, d.h. ob MvP seine Spezifikation erfüllt.

Definition 3.2.6 *Ein Planungszustand* $S = \langle V, P, Z, MvPSubst, MvSubst, Th \rangle$, *mit* $V, Z \in \mathcal{L}_{NP}^{Mv}$, $P \in \mathcal{L}_{Plan}$ *und*
$MvSubst = \emptyset$ *charakterisiert eine Planvalidierungsaufgabe; es gilt dann* $plangen(S) = \emptyset$.

Gesucht ist hier nach einer Menge $MvSubst'$ mit $V' = demod_{MvSubst'}(V)$, $V' \in \mathcal{L}_{NP}$ konsistent, so daß die vollständige Ausführung von P in V' möglich ist.

Der Schlüssel zur Lösung eines spezifizierten Planungsproblems wird im allgemeinen in einer geeigneten Aufteilung in Teilprobleme liegen. Wenn zwischen diesen Teilproblemen Abhängigkeiten bestehen, etwa daß die Ziele des einen Teilproblems Vorbedingungen eines anderen Teilproblems darstellen, können für diese möglicherweise zunächst noch unbekannten Bedingungen Metavariablen eingesetzt werden. Die Lösung eines Teilproblems ist assoziiert mit dem Finden eines Planes, der einen Teilplan für die Lösung des initialen Planungsproblems darstellt. Die unterschiedlichen Teilpläne können im Idealfall sofort zu einem Gesamtplan zusammengesetzt werden, nämlich dann, wenn Abhängigkeiten zwischen Teilproblemen bereits während der Erstellung der Teilpläne berücksichtigt wurden. Im allgemeinen sind aber noch Modifikationsoperationen notwendig, um aus den Teilplänen einen geeigneten Gesamtplan zu erstellen, etwa die Elimination von Redundanzen.

3.3 Die Spezifikation von Kontrollstrategien

Die unterschiedlichen Wissensquellen, die eine Anwendungsdomäne beschreiben, müssen nun koordiniert und gezielt in einem Planungsalgorithmus eingesetzt werden, um gegebene Planungsprobleme zu lösen. Dieser Algorithmus wird, um der geforderten Adaptivität gerecht zu werden, in seinem Kern durch eine konfigurierbare Kontrollstrategie realisiert. Mit Hilfe einer formalen Spezifikation wird dabei ein konzeptuelles Modell der Kontrollstrategie erstellt, das die Basis für deren kontrollierte Adaption bildet. Mit dem Zusatzanspruch der Ausführbarkeit der Spezifikation repräsentiert man gleichzeitig auch ein Verhaltensmodell des zu realisierenden Planers ([Fuchs 92]). Abbildung 3.3 zeigt die prinzipielle Konzeption der adaptierbaren Kontrollstrategien. Es liegt eine generische Strategie vor, die über verschiedene Adaptionspunkte Freiheitsgrade definiert, deren unterschiedlichen Instanziierungen durch einen geeigneten Interpreter eine Planerrealisierung ermöglichen. Die Art der Adaption (und damit die Auswahl der Instanz) wird durch die aktuelle

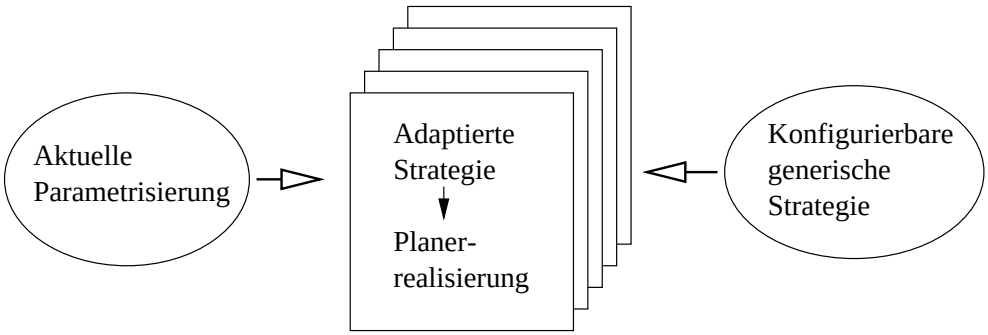


Abbildung 3.3: Prinzip der Kontrollstrategieadaption

Parametrisierung bestimmt, zum einen durch die gewünschte Merkmalsspezifikation der verfügbaren Adaptionspunkte, zum anderen durch ein aktuell zu lösendes Problem.[25]

3.3.1 Adaptionspunkte

Man kann eine Reihe von adaptierbaren Stellen, sogenannte *Adaptionspunkte*, in einer Kontrollstrategie für einen Planer bezogen auf die Planungsproblematik, wie sie in Abschnitt 3.2 beschrieben ist, lokalisieren:

- AP_{split}: Aufteilung in Teilprobleme
  Die Lösung eines Planungsproblems wird reduziert auf die Lösung von Teilproblemen und die Berücksichtigung von Beziehungen zwischen diesen; die Kontrollstruktur zur Verbindung von Teillösungen wird hier ebenfalls festgelegt.

- AP_{foc}: Bearbeitungsfokus
  Wenn mehrere noch nicht gelöste Teilprobleme vorhanden sind, so muß entschieden werden, welches als nächstes bearbeitet wird.

- AP_{imp}: Realisierung der Teilprobleme
  Es gibt unterschiedliche Strategien, wie die Lösung eines Teilproblems gefunden werden kann, etwa unter Verwendung von Abstraktionen oder domänenspezifischen Heuristiken. Gedacht ist hier an Verarbeitungsschritte, die einer Lösung des Teilproblemes unmittelbar vorausgehen, oder sie unmittelbar herbeiführen[26].

- $AP_{probtrans}$: Transformation der Teilprobleme
  Um ein Problem adäquat zu lösen, kann es notwendig oder hilfreich sein, das Problem zunächst umzuformen oder sogar zu erweitern. Umformungen können z.B. dazu dienen, spätere Konflikte zu vermeiden. Es handelt sich also eher um mittelbare Verarbeitungsschritte, die zur Realisierung von Teilproblemen führen (können).

[25]Bei einer domänenabhängigen Kontrollstrategie ist die Domänenbeschreibung in einem Vorverarbeitungsschritt einmalig verwendet worden, um die Kontrollstrategie entsprechend anzupassen.

[26]Etwa die direkte Wiederverwendung vorhandener Lösungen zum gleichen Problem.

- AP_{act}: Aktionsauswahl
  Wenn elementare Teilprobleme erreicht sind, muß entschieden werden, welche Aktion unter welchen Variablenbindungen als Teilplan zum Lösen dieses Problemes verwendet werden soll.[27]

- $AP_{plantrans}$: Plantransformation
  Ein Plan, der bestimmte Merkmale erreichen soll, kann oft besser in zwei Phasen generiert werden. Nach einer Generierungsphase folgt eine mehr oder weniger aufwendige Transformationsphase, die den Plan in seine endgültige Form überführt.

Definition 3.3.1 *Die Adaptionspunkte AP_{split}, AP_{foc}, AP_{imp}, $AP_{probtrans}$, AP_{act} und $AP_{plantrans}$ sind die notwendigen Adaptionspunkte, die ein Planungsalgorithmus zur Erzeugung vollständiger Pläne berücksichtigen muß.*[28] *Sie werden zusammengefaßt in der Menge AP_NEC.*

Jeder der Adaptionspunkte in AP_NEC kann im allgemeinen auf verschiedene Weise konkretisiert werden. Indem bestimmte Alternativen selektiert, zurückgewiesen oder präferiert werden, kann Einfluß auf den Suchprozeß und die Qualität der Ergebnisse genommen werden.
Unter Berücksichtigung der oben genannten Adaptionspunkte kann das in Abbildung 3.4 dargestellte Schema eines Planungsalgorithmus entworfen werden.
Wenn S ein initiales Planungsproblem ist, dann liefert der Funktionsaufruf *planer(S)* eine Lösung dazu. Zunächst erzeugt die Funktion *generiere* einen *Rohplan* (z.B. in einer bestimmten Normalform), der dannach in seine endgültige Form transformiert wird. Während der Generierung müssen verschiedene Schritte durchlaufen werden. Ein sequentieller Algorithmus muß sich zunächst entscheiden, welches der offenen Probleme näher bearbeitet wird. Entweder geschieht dabei eine Aufteilung in handhabbare Teilprobleme, eine Transformation eines Problems zu einer veränderten Problemstellung oder es entstehen (partielle) Lösungen, die möglicherweise neue Probleme induzieren. Die Realisierungsphase wird nach Durchführung bestimmter Vorverarbeitungsschritte eine Aktion auswählen, mit deren Hilfe eine Problemreduktion stattfindet.
Eine gefundene Teillösung wird mit bereits vorhandenen Teillösungen verbunden.
Der gesamte Prozeß wird nun solange weitergeführt, bis keine offenen Probleme, deren Lösung notwendig ist,[29] mehr bestehen. Aus den zwischendurch entstandenen Teillösungen wird dann ein Plan zusammengesetzt.

Die Aufgabe einer Kontrollstrategie ist zum einen, ein Ablaufschema, wie in Abbildung 3.4 skizziert, umzusetzen, und zum anderen, die darin enthaltenen Adaptionspunkte durch ein bestimmtes Verhalten zu konkretisieren. Dazu muß bei der Verfeinerung einer Kontrollstrategie die Lokalisierung von Adaptionspunkten möglich sein und entsprechende Realisierungen der Adaptionspunkte müssen spezifizierbar sein.

[27] Eine Entscheidung kann dabei z.B. bei Optimierungsanforderungen unter globalen Gesichtspunkten, d.h. den Gesamtplan betreffend, gefällt werden.

[28] Manche der Adaptionspunkte können in verschiedenen Planungsalgorithmen mehr oder weniger trivial realisiert sein oder aber auch zusammenfallen, z.B. AP_{imp} und $AP_{probtrans}$.

[29] Bei einer global optimierenden Suche wird ein Und-Oder-Suchbaum verwendet, d.h. es genügt, einen Oder-Zweig abzuschließen, um ein Teilproblem zu lösen.

- **function** *planer(S)* $\rightarrow$

  1. *P* := *generiere(S)*
  2. **return** *Plantransformation(P)*

- **function** *generiere(S)* $\rightarrow$

  if *keine_offenen_Probleme(S)*
  then return *Plan(S)*
  else 1. $\langle AS,RS \rangle$:= *Problemfokussierung(S)*
  2. *AS'* := *Problemaufteilung(AS)*
  or
  AS' := *Problemtransformation(AS)*
  or
  AS' := *Problemrealisierung(AS)*
  3. *S'* := *AS'* $\cup$ *RS*
  4. **return** *generiere(S')*

- **function** *Problemrealisierung(SP)* $\rightarrow$

  1. *SP'* := *Vorverarbeitung(SP)*
  2. *A* := *Aktionsauswahl(SP')*
  3. **return** *Problemreduktion(SP',A)*

Abbildung 3.4: Ein allgemeines Planerschema

Es wird nun eine Spezifikationssprache für Kontrollstrategien vorgestellt, in der die Kontrollstrategie für einen Planer als eine ausführbare Spezifikation gegeben ist. Dies erlaubt einerseits eine frühe Validierung des Planers auf einer abstrakten Ebene, zum anderen dient sie als Prototyp, der es erlaubt unterschiedliche Anforderungen auch experimentell zu überprüfen.

Ein Planungsalgorithmus *plangen* – wie in Abschnitt 3.2 bzw. in Abbildung 3.4 beschrieben – wird nun realisiert durch eine ausführbare Spezifikation formuliert in einer Sprache $\mathcal{TS}$.

Der folgende Abschnitt führt zunächst den "statischen" Teil der Sprache und seine Implementierung in der logischen Programmiersprache PROLOG ein; die Erweiterung auf adaptierbare Strategien wird in Abschnitt 3.4 behandelt.

3.3.2 Eine Spezifikationssprache für Kontrollstrategien

Die Spezifikationssprache $\mathcal{TS}$ ist eine Taktiksprache in der Tradition metalogischer Beweiswerkzeuge (vgl. [Paulson 90]). Sie wird hier in Form einer algebraischen Spezifikation *Tacdef* definiert; der Beschreibungsformalismus folgt der in [Wirsing 90] verwendeten Notation. Zunächst betrachten wir die notwendige Spezifikation von Planungszustandsübergängen (vgl. Definition 3.2.4). Sie bilden die Grundfunktionen eines jeden Planungsalgorithmus und werden durch die Spezifikation *Basic* definiert.

spec *Basic* $\equiv$

extend *Bool, List(X)* **by**

signature $\Sigma_{Basic} \equiv$

sorts *tac, spec*

constructors

PZ_spec_1: $\rightarrow$ *spec*

$\vdots$

PZ_spec_k: $\rightarrow$ *spec*

empty: $\rightarrow$ *list(spec)*

empty: $\rightarrow$ *list(tac)*

functions

$basic_1$: *list(spec) list(spec)* $\rightarrow$ *tac*

$\vdots$

$basic_n$: *list(spec) list(spec)* $\rightarrow$ *tac*

cons: spec list(spec) $\rightarrow$ *list(spec)*

cons: tac list(tac) $\rightarrow$ *list(tac)*

hd: list(spec) $\rightarrow$ *spec*

hd: list(tac) $\rightarrow$ *tac*

hdl: list(spec) $\rightarrow$ *list(spec)*

tl: list(spec) $\rightarrow$ *list(spec)*

tl: list(tac) $\rightarrow$ *list(tac)*

single: list(spec) $\rightarrow$ *bool*

endsignature

axioms
hdl(cons(x,y)) $\rightarrow$ *cons(x,empty)*
single(cons(x,empty)) $\rightarrow$ 1_{bool}
single(cons(x,cons(y,z))) $\rightarrow$ 0_{bool}

endspec

Elemente der Sorte *spec* charakterisieren Planungszustände wie sie in Abschnitt 3.2 beschrieben wurden. *list(spec)* enthält entsprechend Listen von Elementen aus *spec*. Die Funktionen $basic_i$ stellen eine Relation zwischen einem Planungszustand und einer Sequenz von Planungszuständen dar. Der erste Parameter ist dabei auf eine einelementige Liste eingeschränkt; diese Zusicherung folgt später in der Spezifikation $Tacimpl$. Ein Übergang entspricht dabei einer planungsspezifischen Transformation, wie sie im Zusammenhang mit der Realisierung von Adaptionspunkten (vgl. Abschnitt 3.3.1) erwähnt wurden: der Aufteilung von Zielen in Teilziele, dem Erreichen eines Basiszieles durch Anwendung einer Aktion, dem Einführen neuer Ziele, deren Lösung hinreichend ist für die Anwendung einer Aktion, usw.
Die polymorphen Funktionen *cons, hd, tl* sind die Standardlistenoperatoren. Für die leere Liste steht die polymorphe Konstante *empty*. Die Funktion *hdl* spaltet von einer Liste das erste Element ab und bildet daraus wiederum eine Liste.

Die Spezifikation Tac stellt die Basiskontrollmechanismen für Strategien zur Verfügung.

spec $Tac \equiv$

extend *Basic* **by**

signature $\Sigma_{Tac} \equiv$

sorts *tac*

functions
then: list(tac) list(spec) list(spec) $\rightarrow$ *tac*
then2: tac tac $\rightarrow$ *tac*
orelse: list(tac) list(spec) list(spec) $\rightarrow$ *tac*
orelse2: tac tac $\rightarrow$ *tac*
or: list(tac) list(spec) list(spec) $\rightarrow$ *tac*
or2: tac tac $\rightarrow$ *tac*
repeat: tac list(spec) list(spec) $\rightarrow$ *tac*
maptac: tac list(spec) list(spec) $\rightarrow$ *tac*
iftac: tac tac tac $\rightarrow$ *tac*
applytac: tac list(spec) list(spec) $\rightarrow$ *tac*
$selectapply_1$*: tac tac list(spec) list(spec)* $\rightarrow$ *tac*
$\vdots$
$selectapply_l$*: tac tac list(spec) list(spec)* $\rightarrow$ *tac*
append: list(spec) list(spec) list(spec) $\rightarrow$ *tac*
true, false: tac
$selfunc_1$*: list(spec)* $\rightarrow$ *list(spec)*
$\vdots$

selfunc$_l$: list(spec) → list(spec)
decfunc$_1$: list(spec) → list(tac)
⋮
decfunc$_m$: list(spec) → list(tac)
gentac: list(tac) list(spec) → tac
multigentac: list(tac) list(spec) tac → tac
then_expand: list(tac) tac → list(tac)

endsignature

axioms [30]

then(x,a,b) → then2(applytac(hd(x),a,c),then(tl(x),c,b))
orelse(x,a,b) → orelse2(applytac(hd(x),a,b),orelse(tl(x),a,b))
orelse(empty,a,b) → false
or(x,a,b) → or2(applytac(hd(x),a,b),or(tl(x),a,b))
or2(x,y) → orelse2(x,y)
or2(x,y) → orelse2(y,x)
maptac(t,a,b) → then2(applytac(t,hdl(a),c),
then2(maptac(t,tl(a),d),append(c,d,b)))
maptac(t,empty,b) → true
repeat(t,a,b) → iftac(maptac(t,a,c),repeat(t,c,b),true)
applytac(then(x,a,b),c,d) → then(x,c,d)
applytac(orelse(x,a,b),c,d) → orelse(x,c,d)
applytac(or(x,a,b),c,d) → or(x,c,d)
applytac(repeat(x,a,b),c,d) → repeat(x,c,d)
applytac(maptac(x,a,b),c,d) → maptac(x,c,d)
applytac(iftac(x,a,b),c,d) → iftac(applytac(x,c,f),
applytac(a,c,d),applytac(b,c,d))
selectapply$_1$(t1,t2,s,sl) → then2(then2(applytac(t1,hdl(selfunc$_1$(s)),sl1),
append(sl1,tl(selfunc$_1$(s)),sl2)),
applytac(t2,sl2,sl))
⋮
selectapply$_l$(t1,t2,s,sl) → then2(then2(applytac(t1,hdl(selfunc$_l$(s)),sl1),
append(sl1,tl(selfunc$_l$(s)),sl2)),
applytac(t2,sl2,sl))
gentac(decfunc$_1$(a),b) → applytac(hdl(decfunc$_1$(a)),a,b)
⋮
gentac(decfunc$_m$(a),b) → applytac(hdl(decfunc$_m$(a)),a,b)
multigentac(decfunc$_1$(a),b,t) → orelse(then_expand(decfunc$_1$(a),t),a,b)
⋮
multigentac(decfunc$_m$(a),b,t) → orelse(then_expand(decfunc$_m$(a),t),a,b)
then_expand(l,t) → cons(then(cons(hd(l),t),a,b),then_expand(tl(l),t))
then_expand(empty,t) → empty

[30] Die Schreibweise $\phi \to \psi$ eines Axioms ist im Sinne einer gerichteten Gleichheit zu verstehen. Wenn es zwei Axiome $\phi \to \alpha$ und $\phi \to \beta$ gibt, so ist dies im Sinne eines Nichtdeterminismus bei der Ersetzung von ϕ zu verstehen.

endspec

Das Konstrukt *then* steht für eine abhängige Sequenzialisierung: Taktiken aus einer Liste von Taktiken werden sukzessive auf Planungszustände angewendet. Die Resultate einer Taktikanwendung bilden dabei jeweils die Eingaben für die folgende Taktikanwendung.
orelse steht für eine deterministische Auswahl und nachfolgende Anwendung von Taktiken: eine Taktikliste wird dabei sequentiell nach der ersten Taktik durchsucht, die erfolgreich auf einen Planungszustand angewendet werden konnte. Das Konstrukt *or* bezeichnet die nichtdeterministische Variante von *orelse*.
repeat fungiert als eine iterative Version von *then*. *maptac* steht für die unabhängige ("parallele") Anwendung einer Taktik auf unterschiedliche Planungszustände. *iftac* steht für die Kontrollstruktur eines vollständigen Konditionals. *applytac* beschreibt die Anwendung einer Taktik auf spezifische aktuelle Parameter.
Die Funktionen $selfunc_i$ $i = 1, \ldots, l$ stehen jeweils für eine spezielle Methode, eine Liste von Spezifikationen umzuordnen. Die Umordnung geschieht immer so, daß ein Element der Liste nach festgelegten Kriterien ausgewählt wird und an den Anfang der Liste bewegt wird. Verwendung finden diese Funktionen in dem Konstrukt *selectapply*. Dabei wird aus einer Menge von Planungszuständen ein spezieller ausgewählt, auf den eine Taktik angewendet wird. Die resultierenden zusammen mit den nicht ausgewählten Zuständen werden mit einer zweiten Taktik weiterverarbeitet.
Durch die Funktionen $decfunc_i$ $i = 1, \ldots, m$ werden die in Abschnitt 3.1.4 eingeführten Entscheidungsfunktionen realisiert. Wie sie eingesetzt werden, wird mit der Erklärung der Intention für die Taktikkonstrukte *gentac* und *multigentac* deutlich. Eine Entscheidungsfunktion übernimmt z.B. die Auswahl, mit welcher Aktion ein aktuelles, noch nicht erreichtes Ziel erfüllt werden soll, sie ist also abhängig vom aktuellen Planungszustand. Wenn diese Entscheidung getroffen ist, dann soll sie auch umgesetzt werden, d.h. der aktuelle Planungszustand muß entsprechend transformiert werden, so daß er die Anwendung der ausgewählten Aktion widerspiegelt. Die Transformation wird wiederum durch eine Taktik gesteuert. Eine Funktion $decfunc_i$ liefert nun also eine Liste von Taktiken als Ergebnis, wobei jede einzelne die Ausführung einer getroffenen Entscheidung übernehmen kann. Die Taktiken stehen in der Reihenfolge in der Liste, wie sie der durch die mit der Entscheidungsfunktion assoziierten Funktion *choose* (vgl. Seite 53) realisierten Präferenz entsprechen. Das Taktikkonstrukt *gentac* wendet nun die Taktik mit höchster Priorität[31] auf einen Planungszustand an.
Mit dem Taktikkonstrukt *multigentac* hat man die Möglichkeit, auf einen Mißerfolg, der möglicherweise durch eine Entscheidung, wie sie bei *gentac* getroffen wird, hervorgerufen wurde, zu reagieren. Als Mißerfolg wird eine Reduktion auf die Konstante *false* interpretiert. *multigentac* definiert ein *Backtracking* bzgl. aller alternativen Taktiken, die durch die verwendete Entscheidungsfunktion induziert werden, und einer jeweils nachfolgenden davon abhängigen Taktik.

Um eine sinnvolle Verwendbarkeit von Spezifikationen im Sinne der Unterstützung von prozeduralen Abstraktionen zu gewährleisten, wird die Spezifikation *Tac* um definierbare Taktiken erweitert.

spec *Tacdef* $\equiv$

[31] Entspricht dem ersten Element der produzierten Taktikliste.

extend *Tac* **by**

sorts *tacname*

constructors

$a_1, \ldots, a_n$: $\rightarrow$ *tacname*

functions

tacdef: tacname $\rightarrow$ *tac*
calltac: tacname list(spec) $\rightarrow$ *tac*
calltac: tacname list(spec) list(spec) $\rightarrow$ *tac*

axioms

tacdef(a_1) $\rightarrow$ <*ein Term der Sorte tac*>
$\vdots$
tacdef(a_n) $\rightarrow$ <*ein Term der Sorte tac*>
calltac(a,s) $\rightarrow$ *applytac(tacdef(a),s,x)*
calltac(a,s,t) $\rightarrow$ *applytac(tacdef(a),s,t)*

endspec

Die Semantik von Spezifikationen aus *Tacdef* bzw. *Tac* wird definiert durch eine Übersetzung in die logische Programmiersprache PROLOG [Clocksin & Mellish 87], deren prozedurale Interpretation dann ein Ausführungsmodell für Taktiken liefert. Die Implementierung wird ebenfalls durch eine algebraische Spezifikation, *Tacimpl*, definiert; eine Algebra *PROLOGdef* zur syntaktischen Charakterisierung von PROLOG sei gegeben. Darin sind die verfügbaren Operatoren und Prädikate (in Form von booleschen Funktionen) spezifiziert. Obwohl PROLOG eine ungetypte Sprache darstellt, ist es ohne Probleme möglich Sorteninformationen zu integrieren und zu verarbeiten.[32]

spec *Tacimpl* $\equiv$

extend *Tacdef, PROLOGdef* **by**

sorts

functions

axioms

then2(t1,t2) $\rightarrow$ *(t1 , t2)*
orelse2(t1,t2) $\rightarrow$ *if(t1,true,t2)*
or2(t1,t2) $\rightarrow$ *(rel_choose([t1,t2],?f,?s) , if(?f,true,?s))*
iftac(t1,t2,t3) $\rightarrow$ *if(t1,t2,t3)*
true $\rightarrow$ $true_{PROLOGdef}$
false $\rightarrow$ $fail_{PROLOGdef}$
$single(s) \rightarrow 1_{bool} \Rightarrow basic_i(s,sl) \rightarrow rel\_basic_i(s,sl)$
$single(s) \rightarrow 0_{bool} \Rightarrow basic_i(s,sl) \rightarrow false$
$single(u) \rightarrow 1_{bool} \Rightarrow applytac(basic_i(s,sl),u,v) \rightarrow rel\_basic_i(u,v)$

[32]Eine Prozedur $p(A,B) \leftarrow body_p$, bei der A (B) von der Sorte *sorte*1 (*sorte*2) sein soll, kann ersetzt werden durch die Prozedur $p(A,B) \leftarrow sorte1(A), sorte2(B), body_p$. *sorte*1 bzw. *sorte*2 sind Prozeduren, die die Sortenkonformität eines Datenobjektes überprüfen.

$single(u) \rightarrow 0_{bool} \Rightarrow applytac(basic_i(s,sl),u,v) \rightarrow false$

$rel\_basic_i(s,sl) \rightarrow (rel\_selfunc_j(x,?y), rel\_basic_i(s',sl))$

für alle rel_basic_i(s,sl) mit selfunc$_j$(x) ist Unterterm in s und $s' = \sigma(s)$ mit $\sigma = \{?y/selfunc_j(x)\}$

$applytac(t,s,sl) \rightarrow (rel\_decfunc_j(x,?y), applytac(t',s,sl))$

für alle $applytac$(t,s,sl) mit decfunc$_j$(x) ist Unterterm in t und $t' = \sigma(t)$ mit $\sigma = \{?y/decfunc_j(x)\}$

$applytac(tacdef(a),s,sl) \rightarrow (rel\_tacdef(a,?tac), applytac(?tac,s,sl))$

$applytac(t,s,sl) \rightarrow (?z = [a|b], applytac(t',s,sl))$

für alle $applytac$(t,s,sl) mit $cons(a,b)$ ist möglichst tief liegender Unterterm in t und $t' = \sigma(t)$ mit $\sigma = \{?z/cons(a,b)\}$

$applytac(t,s,sl) \rightarrow (l = [?z|\_], applytac(t',s,sl))$

für alle $applytac$(t,s,sl) mit $hd(l)$ ist Unterterm in t und $t' = \sigma(t)$ mit $\sigma = \{?z/hd(l)\}$

$applytac(t,s,sl) \rightarrow (l = [?z|\_], applytac(t,s',sl))$

für alle $applytac$(t,s,sl) mit $hd(l)$ ist Unterterm in s und $s' = \sigma(s)$ mit $\sigma = \{?z/hd(l)\}$

$applytac(t,s,sl) \rightarrow (l = [\_|?z], applytac(t',s,sl))$

für alle $applytac$(t,s,sl) mit $tl(l)$ ist Unterterm in t und $t' = \sigma(t)$ mit $\sigma = \{?z/tl(l)\}$

$applytac(t,s,sl) \rightarrow (l = [?z|\_], applytac(t,[s'],sl))$

für alle $applytac$(t,s,sl) mit $hdl(l)$ ist Unterterm in s und $s' = \sigma(s)$ mit $\sigma = \{?z/hdl(l)\}$

$if(t1,t2,t3) \rightarrow (\mathsf{F}, if(t1',t2,t3))$

für alle $if(t1,t2,t3)$ mit $cons(a,b)$, $hd(l)$ oder $tl(l)$ ist Unterterm in $t1$ und $t1' = \sigma(t1)$ mit
$\sigma = \{?z/cons(a,b)\} \Rightarrow (?z = [a,b])/\mathsf{F}$,
$\sigma = \{?z/hd(l)\} \Rightarrow l = [?z|\_]/\mathsf{F}$ oder
$\sigma = \{?z/tl(l)\} \Rightarrow l = [\_|?z]/\mathsf{F}$

analog für t2 und t3

$applytac(append(a,b,c)s,sl) \rightarrow append_{PROLOGdef}(a,b,sl)$

$applytac(true,c,d) \rightarrow equal(c,d)$

$applytac(false,c,d) \rightarrow fail_{PROLOGdef}$

$then(empty,a,b) \rightarrow equal(a,b)$

endspec

Die Konstanten mit dem Präfix "?" übernehmen hier die Rolle von *Skolemkonstanten*, d.h. bezeichnen jeweils ein neues Objekt. Mit $\sigma(t)$ sei eine Termersetzungsfunktion bezeichnet, die in t Ersetzungen bzgl. einer Substitutionsvorschrift σ vornimmt. Die Funktionen aus $PROLOGdef$ sind boolsche Funktionen und können somit auf natürliche Weise eine logische Interpretation in PROLOG erhalten. Die Funktion ",", Darstellung für das logische

"und", ist hier der Lesbarkeit wegen in Infixnotation geschrieben.
Zu den Funktionen *rel_choose*, rel_basic_i gibt es keine direkten Abbildungen in PROLOG-eigene Prädikate; vielmehr sind entsprechende Prädikate (Prozeduren) durch terminierende logische Programme zu definieren. Die intuitive Bedeutung von *rel_choose* ist, eine nichtdeterministische Auswahl zwischen zwei Elementen zu treffen.
Zu den Funktionen $selfunc_i$ und $decfunc_k$ gibt es zunächst entsprechende boolsche Funktionen $rel\_selfunc_i$ und $rel\_decfunc_k$, für die dann wiederum definierte Prädikate in PROLOG existieren. rel_tacdef ist die korrespondierende boolsche Funktion zu $tacdef$; auch hier existiert ein analoges Prädikat in PROLOG.[33] Es wird vorausgesetzt, daß alle Funktionen rel_* in $PROLOGdef$ spezifiziert sind.

Mit Hilfe der obigen Spezifikationen kann man nun definieren, was eine Planungstaktik ist. Zunächst wird dazu der Begriff *Taktikmodul* definiert:

Definition 3.3.2 *Ein Taktikmodul* $\mathbf{TM}_{tac}$ *ist spezifiziert durch eine Menge von Termen, die der Spezifikation Tacdef genügen, wobei gilt:*

$$\mathbf{TM}_{tac} = T \cup \{calltac(tac, s, sl)\} \text{ mit}$$
$$T \subset \{tacdef(y) |\ y \text{ definierte Konstante der Sorte tacname}\}.$$

Für alle $calltac(t, x, y)$, *Subterm von* $\tau \in \mathbf{TM}_{tac}$ *gilt* $tacdef(t) \in \mathbf{TM}_{tac}$.

$\mathbf{TM}_{tac}$ *heißt dann ein* <u>Taktikmodul</u> *der Taktik tac.*

Die Spezifikation *Tacimpl* kann nun benutzt werden, um eine Berechnung eines Taktikmoduls $\mathbf{TM}_{tac}$ bezogen auf $calltac(tac, s, sl)$ zu beschreiben. Die Axiome der Spezifikation von *Tacimpl*[34] können unmittelbar als Regeln eines Termersetzungssystems TeS interpretiert werden [Dershowitz & Jouannaud 90]. Durch die Anwendung von TeS auf $calltac(tac, s, sl)$ gelangt man aufgrund der rekursiven Anteile prinzipiell zu einem unendlichen Term $TT_{tac} \in T_{\Sigma_{PROLOGdef}}$ mit folgender Struktur:

$$TT_{tac} \equiv t_1, \ldots, t_n, \ldots\ ^{35}$$

t_i ist entweder eine der Funktionen rel_* oder die Funktion $true$, $fail$, $append$, $equal$ bzw. if. Die Argumente von if können wiederum strukturiert sein wie TT_{tac}.

Ein Term TT_{tac} kann nun unmittelbar als PROLOG-Ziel interpretiert werden, d.h. es existiert ein Homomorphismus $h_{tac\_trans} : T_{\Sigma_{PROLOGdef}} \rightarrow F_{PROLOG}$ zur Umsetzung von Termen in PROLOG-Notation in entsprechende Formeln in PROLOG-Notation. Die prozedurale Interpretation von PROLOG legt dann eine Berechnung von TT_{tac} fest.[36]

Definition 3.3.3 *Eine Berechnung eines Taktikmoduls* $\mathbf{TM}_{tac}$ *bzgl. der Parameter s und sl vom Typ list(spec) wird durchgeführt, indem zunächst ein Term* $TT_{tac} \in T_{\Sigma_{PROLOGdef}}$ *durch die Anwendung von Termersetzungen bzgl. TeS aus dem Term*

$$calltac(tac, s, sl) := \sigma(calltac(tac, x, y)),\ calltac(tac, x, y) \in \mathbf{TM}_{tac},\ \sigma = \{s/x, sl/y\}$$

[33] Bei der Realisierung handelt es sich lediglich um PROLOG-Fakten.

[34] In seiner vollständigen Form mit allen Teilspezifikationen.

[35] Auf das Setzen von Klammern wurde verzichtet; "," steht hier für das logische "und" in der PROLOG-Schreibweise.

[36] Vorausgesetzt, daß entsprechende Programme zur Realisierung der Funktionen rel_* vorhanden sind.

erzeugt wird. $h_{tac\_trans}(TT_{tac})$ *erzeugt dann ein PROLOG-Ziel, das im Sinne von PROLOG ausgeführt werden kann. Das Ergebnis einer Berechnung ist vom Typ bool; "true" bezeichnet eine erfolgreiche Berechnung und "false" eine erfolglose.*

Um die Berechnung eines Taktikmoduls praktisch durchführen zu können, muß im allgemeinen ein Wechsel zwischen einer Termersetzungsphase und einer PROLOG-Zielauswertungsphase erfolgen.[37] Eine Berechnungsstrategie wird deshalb einer *"lazy evaluation"*-Strategie (vgl. [Antoy 91, Antoy et al. 93]) folgen, d.h. Ausdrücke werden nur dann ausgewertet, wenn dies notwendig ist. Die Strategie folgt dabei auch der *"von-links-nach-rechts"*-PROLOG-Auswertungsstrategie. Termersetzungen bzgl. *Tacimpl* werden so ausgeführt, daß möglichst schnell ein Ausdruck entsteht, der mit Hilfe von $h_{tac\_trans}(TT_{tac})$ partiell in PROLOG ausgewertet werden kann. Die im folgenden schematisch beschriebene Funktion *eval* beschreibt eine entsprechende Berechnungsstrategie:

function $eval(tac) \leftarrow$

1. reduziere *tac* durch die Anwendung von TeS zu tac' bis gilt
2. **case** tac' **of**

   (if(a,b,c),Rest):
   if $eval(a) = true$ **then return**$(eval((b, Rest)))$ **else return**$(eval((c, Rest)))$

   if(a,b,c):
   if $eval(a) = true$ **then return**$(eval(b))$ **else return**$(eval(c))$

   (basic,Rest):
   %% *basic* ist entweder eine der Funktionen rel_* oder
   %% die Funktion *true*, $fail$, *append*

   if $PROLOG\text{-}eval(basic) = \langle true, \sigma \rangle$ **then return**$(eval(\sigma(Rest)))$
   else return$(fail)$

   basic:
   %% *basic* ist entweder eine der Funktionen rel_* oder
   %% die Funktion *true*, $fail$, *append*, *equal*
   if $PROLOG\text{-}eval(basic) = \langle true, * \rangle$ **then return**$(true)$ **else return**$(fail)$

   esac

Die Funktion $PROLOG\text{-}eval(Z)$ symbolisiert die Auswertung des Zieles Z in PROLOG. Sie liefert demnach ein Tupel zurück bestehend aus dem Wahrheitswert der Auswertung und einer Menge von Substitutionen für Skolemkonstanten, die als freie Variablen im PROLOG-Ziel interpretiert werden. Die Substitution wird durch eine Termersetzungsfunktion über die restlichen Ziele propagiert. *eval* beschreibt, in welchem Sinne ein Taktikmodul eine ausführbare Spezifikation darstellt.

Eine Planungstaktik kann nun wie folgt charakterisiert werden:

[37] Notwendig ist dies nicht für den Fall, daß keinerlei Rekursion auftritt. Wenn Konditionale auftreten, ist dies sehr sinnvoll, um unnötige Reduktionen zu vermeiden.

Definition 3.3.4 *Eine Planungstaktik ist ein Taktikmodul* $\mathbf{TM}_{planer}$ *mit folgenden Eigenschaften:*

- $calltac(planer, X, XL) \in \mathbf{TM}_{planer}$,
- *für* $S =< V, MvP, Z, MvPSubst, MvSubst, Th >$ *ein initialer Planungszustand und* SL *eine n-elementige Liste von Planungszuständen mit den Elementen* $S_k =< V_k, MvP_k, Z_k, MvPSubst_k, MvSubst_k, Th >$ *für* $k = 1, \ldots, n$ *gelte*
  - $Z_k = true_{\mathcal{L}}$[38] *für* $k = 1, \ldots, n$;
  - $MvP_k \in \mathcal{L}_{Plan}$ *für* $k = 1, \ldots, n$;
  - *für ein* $i \in \{1, \ldots, n\}$ *gilt* $P := demod_{MvPSubst_i}(MvP)$, $P \in \mathcal{L}_{Plan}$ *und* P *ist eine Lösung von* S;
  - *die Berechnung von* $\mathbf{TM}_{planer}$ *bzgl. der aktuellen Parameter* S *und* SL *ist erfolgreich.*

Abbildung 3.5 verdeutlicht die Zusammenhänge in dem bisher eingeführten Konzept der Kontrollstrategierealisierung durch Taktiken. Der Metainterpreter ist durch das Termer-

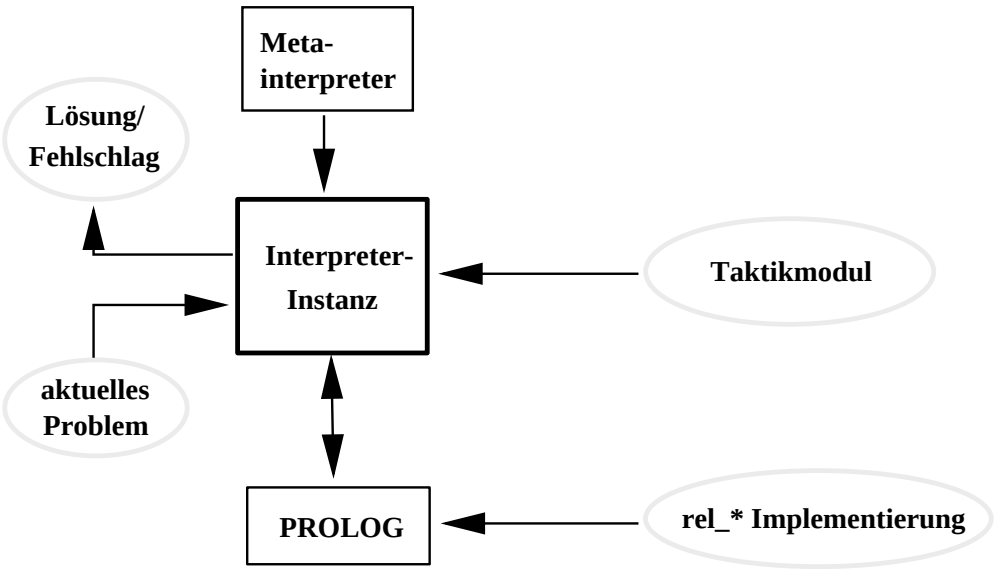


Abbildung 3.5: Prinzipielle Taktikverarbeitung

setzungssystem TeS und die realisierte Berechnungsstrategie beschrieben. Initialisiert mit einem Taktikmodul, das einer Planungstaktik entspricht, ergibt sich eine Interpreterinstanz, die die Kontrollstrategie eines Planers realisiert. Sie verarbeitet ein aktuelles Problem und erzeugt Lösungen oder zeigt Fehlschläge an. Zur Auswertung von taktikspezifischen Basisschritten stehen die rel_*-Implementierungen zur direkten Ausführung als PROLOG-Prozeduren zur Verfügung.
Im folgenden Abschnitt erfolgt die Erweiterung des Taktikkonzeptes hin zu konfigurierbaren Taktiken.

[38] $true_{\mathcal{L}}$ steht für die Formel *true* in der logischen Sprache $\mathcal{L}$.

3.4 Konfigurierbare Strategien

Ein Planungsalgorithmus kann durch eine spezifische Planungstaktik wie sie oben eingeführt wurde realisiert werden. Solch eine Planungstaktik wird implizit auf eine bestimmte Weise konkrete Realisierungen der Adaptionspunkte AP_NEC (vgl. Seite 65) enthalten. Dies führt dazu, daß die Pläne, die man durch Verwendung des Planungsalgorithmus erhalten wird, auch zu einer konkreten Planklasse mit spezifischen Merkmalen gehören. Das wiederum bedeutet, daß, wenn Repräsentanten verschiedener Planklassen gefordert sind (vgl. dazu das Kapitel 1), jeweils unterschiedliche Planungsalgorithmen notwendig sind. Die Vielzahl möglicher Varianten rechtfertigt die Realisierung von generischen Planungsalgorithmen, die gemäß der abstrakten Spezifikation ihres Verhaltens an Adaptionspunkten zu konkreten Planungsalgorithmen konfiguriert werden können (vgl. nochmals Abbildung 3.3).

3.4.1 Eine erweiterte Spezifikationssprache

Zunächst wird zu dem Zweck der Variantenbildung von Kontrollstrategien die Spezifikation *Tacdef* auf eine Spezifikation *Tackonf* (siehe unten) erweitert. Es können dann Adaptionspunkte in eine Taktik eingeführt werden, die später auf verschiedene Weise konkretisiert werden können. Zu bemerken ist, daß eine Taktik mit noch nicht konkretisierten Adaptionspunkten nicht ausgeführt werden kann.

spec *Tackonf* $\equiv$

extend *Tacdef* **by**

sorts *adaptation*

constructors

$AP_{split}, AP_{foc}, AP_{imp}, AP_{probtrans}, AP_{act}, AP_{plantrans}: \rightarrow adaptation$

functions

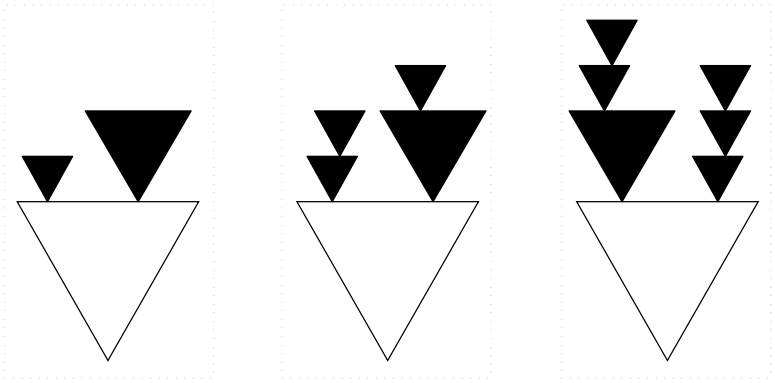
choicetac: adaptation $\rightarrow$ *tac*

axioms

endspec

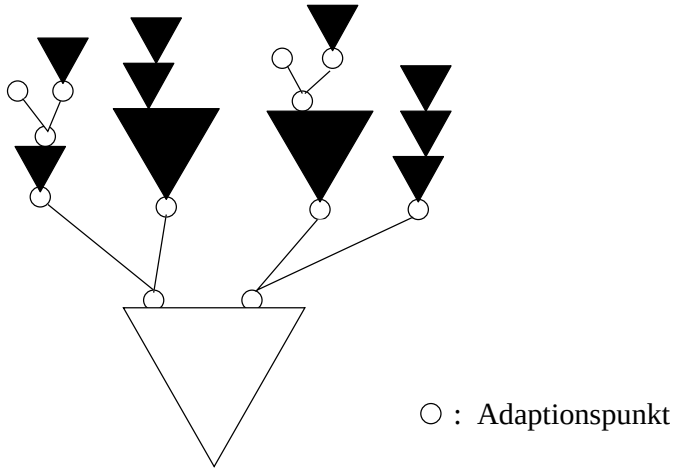
Die Intention der Verwendung von Funktionen *choicetac* zum Aufbau von Kontrollstrategien ist, eine Strukturierung der Strategien zu unterstützen und zunächst eine zusätzliche Abstraktion zu erreichen. Die Abbildung 3.6 verdeutlicht grob den Zusammenhang.
In Teil (a) der Abbildung sind drei Kontrollstrategien symbolisiert, die strukturelle und kodierungsbezogene Gemeinsamkeiten haben. Die weißen Dreiecke sollen einen allen Strategien gemeinsames Skelett symbolisieren; bestimmte Stellen werden aber unterschiedlich konkretisiert. Teil (b) symbolisiert, wie man die drei Strategien in einem Oder-Baum zusammenfassen kann. Da durch die Abstraktion nun i.a. zunächst mehr Möglichkeiten der Konkretisierung entstehen als als Basis zur Verfügung standen, ist nun ein zusätzlicher Mechanismus (siehe unten: *Regeln zur Merkmalsumsetzung*) notwendig, um nicht erlaubte Kombinationen auszuschließen.

(a)



3 unabhängige Planungsstrategien

(b)



Abstraktion und Strukturierung

Abbildung 3.6: Einführung von Adaptionspunkten

Der Aufbau von Kontrollstrategien sollte nun so geschehen, daß man möglichst nur die Situation (b) anstrebt, d.h. man muß geeignete Adaptionspunkte innerhalb von Strategien definieren.

In Abschnitt 3.3.1 wurden charakteristische Adaptionspunkte genannt, um eine Planervariante zu charakterisieren; sie waren in der Menge

$$AP\_NEC = \{AP_{split}, AP_{foc}, AP_{imp}, AP_{probtrans}, AP_{act}, AP_{plantrans}\}$$

zusammengefaßt. Um nun festzulegen, wie sie konkretisiert werden können, stehen jedem Adaptionspunkt zunächst eine Menge von Basismerkmalen

$$features(AP_i) = \{m_1^i, \ldots, m_k^i\},\ i \in \{split, foc, imp, probtrans, act, plantrans\},\ 1 \leq k$$

zur Verfügung. Ergänzend gibt es zusätzlich Funktionen über den Merkmalen, um (mit bestimmten Einschränkungen) neue Qualitätsmerkmale zu definieren. Jedem Adaptionspunkt AP_i kann eine Menge von Kombinationsoperatoren O_i zugeordnet werden:

$$O_i \subseteq \{prefer\_to, before, and, some\_of, adapt\_to\_when\},$$
$$i \in \{split, foc, imp, probtrans, act, plantrans\}.$$

Σ^i_{Adapt} sei die Signatur, die durch $features(AP_i)$ und O_i gebildet wird. Die aktuelle Spezifikation $spec(AP_i)$ eines Adaptionspunktes AP_i ist dann ein Term aus $T(\Sigma^i_{Adapt})$. Da es im allgemeinen nicht für jeden Term aus $T(\Sigma^i_{Adapt})$ eine sinnvolle Interpretation gibt, kann man *zulässige Kombinationen* definieren.

Definition 3.4.1 *Für jeden Adaptionspunkt $AP_i \in AP\_NEC$ und jeden Operator $o_{ij} \in O_i$ gibt es eine partielle Funktion*

$$kombi_{ij} : P_{ij} \times Q_{ij} \rightarrow R_{ij} \cup \{nil\}$$
$$mit\ P_{ij}, Q_{ij}, R_{ij} \subseteq features(AP_i),$$

Die Gesamtheit $Kombi_{Adapt}$ der Funktionen $kombi_{ij}$[39] charakterisiert die zulässigen Terme aus $T(\Sigma_{Adapt})$.

Mit einer Menge

$$APM = \{spec(AP_i) |\ i \in \{split, foc, imp, probtrans, act, plantrans\}\}$$

von Termen aus *Adapt* wird dann eine Planungstaktikvariante auf einem geeigneten Abstraktionsniveau spezifiziert. Die folgende Spezifikation, *Adapt*, beschreibt nun neben den Merkmalen selbst auch die Syntax der Merkmalsfunktionen.

spec *Adapt* $\equiv$

extend *Tacimpl* **by**

signature $\Sigma_{Adapt} \equiv$

[39]Dem 3-stelligen Operator *adapt_to_when* wird nur eine zweistellige Kombinationsfunktion zugeordnet, da das dritte Argument kein Adaptionsmerkmal darstellt.

sorts *feature, condition, condition_type*

constructors

$m_1, \ldots, m_n$: *feature*
nil: feature
$c_1, \ldots, c_m$: *condition*
$t_1, \ldots, t_k$: *condition_type*

functions

prefer_to: feature feature $\rightarrow$ *feature*
before: feature feature $\rightarrow$ *feature*
and: feature feature $\rightarrow$ *feature*
some_of: feature feature $\rightarrow$ *feature*
adapt_to_when: feature feature condition $\rightarrow$ *feature*
valid_combination: feature $\rightarrow$ *feature*
nec_subterm: feature $\rightarrow$ *feature*
cond_type: condition $\rightarrow$ *condition_type*

endsignature

axioms

$valid\_combination(before(m_i, m_j)) \rightarrow m_k$
$\vdots$
$nec\_subterm(before(m_i, m_j)) \rightarrow nec\_subterm(m_i)$
$nec\_subterm(before(m_i, m_j)) \rightarrow nec\_subterm(m_j)$
$nec\_subterm(and(m_i, m_j)) \rightarrow nec\_subterm(m_i)$
$nec\_subterm(and(m_i, m_j)) \rightarrow nec\_subterm(m_j)$

$nec\_subterm(m_i) \rightarrow m_i$
$\vdots$
$cond\_type(c_1) \rightarrow t_{c_1}$
$\vdots$
$cond\_type(c_m) \rightarrow t_{c_m}$

endspec

Die Axiome der Funktion *valid_combination* in der Spezifikation *Adapt* beschreiben die Graphen der Funktionen $kombi_{ij}$. $Kombi_{Adapt}$ gibt an, welche Merkmale mit einem Operator o_{ij} (sinnvoll) kombiniert werden können und wie das Resultat dieser Kombination in neuen Kombinationen verwendet werden kann; z.B. $kombi_{ij}(m_a^i, m_b^i) = m_a^i$ drückt aus, daß die Merkmale m_a^i und m_b^i mit dem Operator o_{ij} kombiniert werden können und das Resultat dieser Kombination kann genau so weiterverwendet werden, wie m_a^i alleine. Eine Abbildung von $kombi_{ij}$ auf das Symbol *nil* bedeutet, daß das Resultat einer Kombination zweier Merkmale nicht für weitere Kombinationen verfügbar ist. Die intuitive Bedeutung der in *Adapt* definierten Kombinationsfunktionen wird wie folgt angegeben:

- Der Operator *prefer_to* steht für eine deterministische Auswahl im Sinne von
  "erreiche wenn möglich eher Merkmal 1 als Merkmal 2" oder
  "versuche zuerst Methode 1 sonst Methode 2"

- Der Operator *before* steht für eine Sequenzialisierung:
  "wende Methode 1 an und danach Methode 2"
- Der Operator *and* steht für Gemeinsamkeit:
  "es sind nur Lösungen erlaubt, die nach Methode 1 und Methode 2 entstehen"
- Der Operator *some_of* steht für nichtdeterministische Auswahl:
  "versuche eine beliebige von zwei Methoden"
- Der Operator *adapt_to_when* steht für die Anpassung einer Methodenverwendung, sobald eine spezifizierte Bedingung gilt.

Die Funktion *nec_subterm* beschreibt die Zerlegung von determinierten Merkmalkombinationen in ihre Einzelteile. *cond_type* klassifiziert eine Bedingung als zu einem bestimmten Typ gehörig. Ein Typ stellt dabei im wesentlichen eine Abstraktion mehrerer parametrisierter Konkretisierungen dar.
Das Ziel besteht nun darin, aus der Menge APM automatisch eine Menge von Axiomen zu gewinnen, die *choicetac*-Terme reduzieren können, d.h. Terme aus $T(\Sigma_{Adapt})$ werden dabei zu Taktiken transformiert.

Die Realisierung eines spezifizierten aktuellen Merkmales $spec(AP_i) \in APM$, also die Taktikumsetzung, die ihm entspricht, kann durch eine Menge von (konditionalen) Termersetzungsregeln und speziellen Anweisungen zur dynamischen Veränderung von spezifizierten Merkmalen beschrieben werden. Diese Veränderung führt dann letztendlich wiederum zu Termersetzungsregeln, deren Entstehen weiter unten (vgl. Seite 85 ff.) näher erläutert wird. Die Ersetzungsregeln sind äquivalent zu entsprechenden Axiomen einer algebraischen Spezifikation $Tacadapt$, die die Verbindung von Taktiken, Adaptionspunkten und Merkmalsspezifikationen herstellt (vgl. Seite 83 ff.).
Im folgenden werden zunächst die berücksichtigten Arten von Regeln bzw. Beziehungen zwischen Merkmalen dargestellt. Da sie möglicherweise einen "*dynamischen*" Teil enthalten, können sie nicht direkt als Axiome in die Spezifikation *Tacadapt* aufgenommen werden, sondern müssen zunächst geeignet interpretiert werden.
Es folgt nun eine Typisierung der Merkmalsumsetzungen sowohl für Basismerkmale als auch unter Berücksichtigung von Merkmalskombinationen.

1. (a) Ein Basismerkmal m_j^i von AP_i wird realisiert, indem für seinen primären Adaptionspunkt (AP_i) eine Taktik angegeben wird und ergänzend eine (möglicherweise leere) Sequenz von Anweisungen zur Erweiterung von Spezifikationen anderer (sekundärer) Adaptionspunkte[40]:

$$feature\_tac(m_j^i) \rightarrow tac_j^i \text{ und } (\mathbf{extend}(AP_{\pi(k)}, spec_k))_{k=1}^{n^j}$$

   Es gilt:

$$\{\pi(k) | k = 1, \ldots, n^j\} \subseteq \{split, foc, imp, probtrans, act, plantrans\} \setminus \{i\}.$$

[40] Eine Erklärung der **extend**-Anweisungen zur Erweiterung von Adaptionspunktspezifikationen erfolgt weiter unten.

Eine Realisierung heißt *kontextfrei*, wenn sie keine sekundären Adaptionspunkte spezifiziert, sonst *kontextsensitiv*. Die **extend**-Anweisungen drücken eine notwendige Anpassung anderer Adaptionspunkte aus. Wenn diese Anpassungen mit den bereits vorliegenden Spezifikationen nicht vereinbar sind, dann ersetzen sie diese.

(b) Eine zweite Möglichkeit besteht darin, eine von der aktuellen Spezifikation eines zweiten Merkmales abhängige Realisierung eines Basismerkmals zu spezifizieren:

$$
\begin{array}{l}
current\_features(AP_l) \rightarrow m \Rightarrow \\
\quad nec\_subterm(m) \rightarrow m_1^l \Rightarrow \\
\quad\quad feature\_tac(m_j^i) \rightarrow tac_{j^1}^i \text{ und } (\mathbf{extend}(AP_{\pi(1,k)}, spec_{1k}))_{k=1}^{n^1} \\
\vdots \\
current\_features(AP_l) \rightarrow m \Rightarrow \\
\quad nec\_subterm(m) \rightarrow m_z^l \Rightarrow \\
\quad\quad feature\_tac(m_j^i) \rightarrow tac_{j^z}^i \text{ und } (\mathbf{extend}(AP_{\pi(z,k)}, spec_{zk}))_{k=1}^{n^z}
\end{array}
$$

d.h. in Abhängigkeit des notwendigen Vorhandenseins eines bestimmten Merkmals in der aktuellen Spezifikation eines zweiten Adaptionspunktes verändert sich die Realisierung des Merkmals m_j^i; die Funktion $current\_features$ beschreibt die Verbindung zwischen Adaptionspunkt und aktueller Merkmalsspezifikation. Es gilt dabei:

$$\{\pi(r,k) | k = 1, \ldots, n^r\} \subseteq \{split, foc, imp, probtrans, act, plantrans\} \setminus \{i\}.$$

Bemerkung: Die Terme tac_*^* können selbst auch wieder $feature\_tac$-Terme sein. Man kann in diesem Sinne *abstrakte* Merkmale definieren, um sie zur gezielteren Steuerung der Merkmalsumsetzungen zu verwenden.

2. Ist die Spezifikation eines Adaptionspunktes AP_i durch eine Merkmalskombination gegeben, so kann mit Hilfe von $Kombi_{Adapt}$ zunächst ihre Zulässigkeit entschieden werden. Eine zulässige Spezifikation kann dann wie folgt realisiert werden:

   Bei der Realisierung von $o_{ij}(m_a^i, m_b^i)$ bzw. $o_{ij}(m_a^i, m_b^i, c)$ werden folgende Fälle unterschieden:

   (a) $feature\_tac(o_{ij}(m_a^i, m_b^i)) \rightarrow tac_{o_{ij}}[feature\_tac(m_a^i), feature\_tac(m_b^i)]$
   $tac_{o_{ij}}$ beschreibt ein Taktikschema, das den Operator o_{ij} realisiert. Für m_a^i und m_b^i gibt es Realisierungen, die das Taktikschema $tac_{o_{ij}}$ parametrisieren. Wenn für m_a^i (m_b^i) eine kontextsensitive Realisierung resultiert, dann können die darin spezifizierten sekundären Adaptionspunkte im allgemeinen zunächst nicht vollständig realisiert werden, da ihre Merkmalskombination sich zur Laufzeit verändern kann.

   (b) $feature\_tac(o_{ij}(m_a^i, m_b^i)) \rightarrow feature\_tac(m_c^i)$
   Die Kombination der beiden Merkmale m_a^i und m_b^i mit dem Operator o_{ij} führt zu dem gleichen Resultat, wie wenn das Merkmal m_c^i alleine verwendet wird,

d.h. es existiert z.B. eine äquivalente, syntaktisch komplexere[41] Darstellung eines Merkmales.

(c) $feature\_tac(o_{ij}(m_a^i, m_b^i)) \rightarrow feature\_tac(o_{ij'}(m_a^i, m_b^i))$
Die Verwendung des Operators o_{ij} kann bei der Anwendung auf bestimmte Merkmale nur ein *Alias* für einen anderen Operator sein.[42]

(d) $feature\_tac(o_{i\ and}(m_a^i, o_{i\ and}(m_b^i, \ldots, m_r^i)\ldots)) \rightarrow$
$tac[intersection(decfunc_a, intersection(decfunc_b, \ldots, decfunc_r)\ldots)]$
Dies beschreibt den Spezialfall für den Operator *and* in bestimmten Situationen. Es gilt dabei, daß eine kontextfreie Realisierung für m_a^i,..., m_r^i durch mit einer Entscheidungsfunktion parametrisiertes Taktikschema $tac[decfunc_a]$,..., $tac[decfunc_r]$ gegeben ist. Für die m_x^i, $x \in \{a, \ldots, r\}$ existieren dabei gleiche Taktikschemata. Der Operator *and* führt dann zur Schnittbildung der assoziierten Entscheidungsfunktionen bei Beibehaltung des gleichen Taktikschemas.

(e) $feature\_tac(o_{ij}(m_a^i, m_b^i)) \rightarrow tac_{o_{ij}}[tac_a^i[feature\_tac(m_c^i)]]$
Die Realisierung des Merkmals m_a^i hat hier nur zu einer weiteren Spezialisierung des Adaptionspunktes i beigetragen. Es kann dabei auch der Spezialfall auftreten, daß keine explizite Realisierung von o_{ij} vorgenommen wird; es gilt dann:
$feature\_tac(o_{ij}(m_a^i, m_b^i)) \rightarrow tac_a^i[feature\_tac(m_c^i)]$

(f) Beim Vorkommen des Operators *adapt_to_when* kann es eine von dem Typ der spezifizierten Bedingung c abhängige Realisierung geben, die für jeden einzelnen Typ veränderte Merkmale vorsehen kann.

$$
\begin{array}{l}
cond\_type(c) \rightarrow t_1 \Rightarrow \\
\quad feature\_tac(o_{i\ adapt\_to\_when}(m_a^i, m_b^i, c)) \rightarrow \\
\qquad applytac(iftac(cond\_real(c), feature\_tac(m_b^{i\,1}), feature\_tac(m_a^{i\,1})), s, sl) \\
\vdots \\
cond\_type(c) \rightarrow t_z \Rightarrow \\
\quad feature\_tac(o_{i\ adapt\_to\_when}(m_a^i, m_b^i, c)) \rightarrow \\
\qquad applytac(iftac(cond\_real(c), feature\_tac(m_b^{i\,z}), feature\_tac(m_a^{i\,z})), s, sl)
\end{array}
$$

Die Realisierung von Merkmalen muß den Test der Bedingung c auch ermöglichen, d.h. z.B. für den Fall, daß die Bedingung c einen Schwellwert für den Verbrauch einer bestimmten Resource festlegt, die Realisierung der Merkmale die Protokollierung des spezifischen Resourcenverbrauches auch vornehmen müssen.
Zwischen Merkmalen $m_j^{i\,k}$, die aus Merkmalen m_j^i unter der Bedingung eines Types t_k entstehen, besteht also ein funktionaler Zusammenhang der Art:

$$m_j^{i\,k} := feature\_mod(m_j^i, t_k)$$

[41] Möglicherweise aber leichter verständlich, da weniger abstrakt.

[42] Der Operator *and* kann u.U. verwendet werden wie der Operator *before*, d.h. *and* wird so realisiert wie *before*.

Hierin ist $feature\_mod$ eine Funktion, die die Umwandlung beteiligter Merkmale vornimmt (auch innerhalb von Merkmalskombinationen); ihr Graph ist individuell zu definieren.

Für alle bzgl. Σ_{Adapt} definierten Merkmale und zulässigen Merkmalskombinationen sind Regeln gemäß den Fällen $1.(a) - 2.(f)$ zu definieren. Für den Fall 2(a) können oft folgende Standardübersetzungen verwendet werden:

$feature\_tac(prefer\_to(a, b)) \rightarrow orelse(cons(feature\_tac(a), feature\_tac(b)), s, sl)$
$feature\_tac(before(a, b)) \rightarrow then(cons(feature\_tac(a), feature\_tac(b)), s, sl)$
$feature\_tac(some\_of(a, b)) \rightarrow or(cons(feature\_tac(a), feature\_tac(b)), s, sl)$

Es ergeben sich Möglichkeiten der dynamischen Einflußnahme auf die Adaptionspunktrealisierungen:

- Durch die Verwendung von **extend**-Anweisungen hat man einen *Propagierungsmechanismus* zur Verfügung, d.h. wird ein bestimmtes Merkmal realisiert, kann es Anpassungen an anderen Adaptionspunkten bewirken (Typ 1(a)).
- Durch die Verwendung von bedingten Realisierungen (Typ 1(b) ohne **extend** und Typ 2(f)) steht ein *Reaktionsmechanismus* zur Verfügung; man kann damit u.a. auch erreichen, daß die Realisierung eines Merkmales die Anpassung der Realisierung eines zweiten Merkmales indirekt bewirkt.
- Verwendet man Typ 1(b)-Realisierungen mit **extend**-Anweisungen, so kann man gleichzeitig reagieren und propagieren.
- Anpassungen von Merkmalen können auch deren Deaktivierung bedeuten; realisiert wird dann beispielsweise ein Merkmal durch die Angabe einer *leeren* Taktik.
- Um im Falle einer Typ 2(f)-Realisierung auch propagieren zu können, muß man die beteiligten Merkmale in Varianten, *indiziert* mit dem Typ der verwendeten Bedingung, vorhalten; bei der Realisierung des indizierten Merkmales kann dann auch eine Propagierung stattfinden.

Die Spezifikation $Tacadapt$ kann nun schrittweise beginnend mit ihrem Grundgerüst aufgebaut werden. Zur Erzeugung der Axiome zur Taktikumsetzung wird danach ein inkrementelles Verfahren angegeben.

spec $Tacadapt \equiv$

extend $Tackonf, Adapt$ **by**

signature $\Sigma_{Tacadapt} \equiv$

sorts

constructors

functions

current_features: adaptation → feature
feature_tac: feature → tac
intersection: list(tac) list(tac) → list(tac)
cond_real: condition list(spec) list(spec) → tac

endsignature

axioms

applytac(t,s,sl) → (rel_intersection(x,y,z),applytac(t',s,sl))

für alle *applytac*(t,s,sl) mit intersection(x,y) ist Unterterm in t und $t' = \sigma(t)$ mit $\sigma = \{z/decfunc_j(x,y)\}$

choicetac(a) → t ⇒
applytac(choicetac(a),s,sl) → applytac(t,s,sl)
applytac(feature_tac(m),s,sl) → (rel_feature_tac(m,?t),applytac(?t,s,sl))
cond_real(c,s,sl) → rel_cond_real(c,s)

endspec

Die Funktion *current_features* stellt die Verbindung zwischen den aktuellen Merkmalen und dem Adaptionspunkt, den sie charakterisieren, dar. Axiome dieser Funktion werden dynamisch erzeugt.
rel_intersection und *rel_cond_real* bezeichnen logischen Programmen entsprechende boolsche Funktionen aus der Spezifikation $PROLOGdef$, ebenso die Funktion $rel\_feature\_tac$. *rel_intersection* hat die Aufgabe, den geordneten Durchschnitt zwischen zwei geordneten Listen zu berechnen und evaluiert nur dann zu *true*, wenn ein nichtleerer Durchschnitt entsteht.[43]

Die für den Adaptionspunkt AP_i bzgl. Σ^i_{Adapt} definierten Termersetzungsregeln und **extend**-Anweisungen werden in einem *erweiterten* Termersetzungssystem $ETeS^i$ zusammengefaßt. Die Gesamtheit aller $ETeS^i$, $i \in \{split, foc, imp, probtrans, act, plantrans\}$ wird durch das System $ETeS_{Adapt}$ beschrieben. Das Teilsystem, das nur konditionale Regeln gemäß 1.(b) enthält, sei mit $ETeS^{KER}_{Adapt}$ benannt.
Wenn in einer der möglichen Realisierungen eines Adaptionspunktes AP_i eine Anweisung der Form **extend**$(AP_j, spec_k)$ vorkommt, dann heißt AP_j ein *dynamischer* Adaptionspunkt, sonst ein *statischer* Adaptionspunkt. Wenn die Spezifikation eines Adaptionspunktes eine *adapt_to_when*-Operation enthält, dann heißt der Adaptionspunkt *selbstadaptiv*.
Eine Funktion

$$status : \{ETeS^i\}_{i \in \{split,foc,imp,probtrans,act,plantrans\}} \rightarrow \{\text{statisch, dynamisch, selbstadaptiv}\}$$

vollziehe diesen Test.

[43] Die Stellung eines Elementes innerhalb einer Liste drückt dabei direkt eine Präferenzordnung aus. Die Stellung $s_{l_1,l_2}(E)$ eines Elementes E in der Durchschnittsliste ergibt sich durch die Gewichtung der einzelnen Bewertungen, beispielsweise gemäß $s_{l_1,l_2}(E) := f(R, (s_{l_1}(E) + s_{l_2}(E)) * min(s_{l_1}(E), s_{l_2}(E)))$. Die Funktion f normalisiert dabei die sich ergebende Bewertungszahl bzgl. der sich für die übrigen Elemente R der Durchschnittsliste ergebenden Bewertungen, um konkrete Listenpositionen zu erhalten.

Aus $ETeS_{Adapt}$ können nun nicht unmittelbar Axiome zur Erweiterung von $Tacadapt$ gewonnen werden, da die enthaltenen **extend-**Anweisungen zunächst umgesetzt werden müssen. Durch das gleichzeitige Vorhandensein konditionaler Taktikumsetzungsregeln ergibt sich die Notwendigkeit eines inkrementellen Vorgehens.

3.4.1.1 Die Adaptionspunktrealisierung

Ein Adaptionspunkt ist während der gesamten Taktikumsetzungsphase durch zwei Eigenschaften charakterisiert:

1. durch seine aktuelle vollständige Merkmalspezifikation und
2. durch seinen aktuellen Merkmalsfokus als Teil von 1., für den eine Realisierung ansteht.[44]

Eingabe für den inkrementellen Algorithmus ist demnach der Merkmalsfokus eines Adaptionspunktes und die vollständigen Spezifikationen aller Adaptionspunkte. Als Ausgabe ergibt sich bei Erfolg ein Axiom zur weiteren Realisierung des Merkmalsfokus und möglicherweise haben die Spezifikationen der Adaptionspunkte Änderungen erfahren.
Aufgrund der durch die Taktikumsetzungsregeln gemäß den Formen $1.(a)-(b)$ möglichen Definition wechselseitiger Abhängigkeiten setzen wir um Konflikte zu vermeiden voraus, daß eine totale Ordnung $<_{AP}$ zwischen den Adaptionspunkten in AP_NEC besteht, die einen steigenden Grad von *Wichtigkeit* zwischen Adaptionspunkten repräsentiert.
Es folgen zunächst noch einige Festlegungen. Ein Adaptionspunktzustand für den Adaptionspunkt a ist während des Algorithmusablaufs beschrieben durch ein Tupel

$$\langle c\_spec, foc\_spec, foc\_real, c\_extend\rangle_a$$

mit *c_spec* die aktuelle vollständige Adaptionspunktspezifikation, foc_spec der Merkmalsfokus, foc_real das Resultat der bereits durchgeführten Realisierung des Merkmalsfokus (initial *nil*) und *c_extend* die noch nicht vollzogenen **extend-**Anweisungen des Adaptionspunktes (initial *nil*). $APM(i)$ sei die Menge der einzelnen Adaptionspunktzustände mit einem spezifizierten Merkmalsfokus auf Adaptionspunkt i.
Durch eine Prozedur $konfig$ wird nun eine Menge $APM(i)'$ berechnet, die sich von $APM(i)$ dadurch unterscheidet, daß eine Transformation von Adaptionspunktspezifikationen bzgl. $ETeS_{Adapt}$ so weit wie möglich durchgeführt wurde. Wenn für

$$\langle c\_spec, foc\_spec, foc\_real, c\_extend\rangle_i \in APM(i)'$$

die Eigenschaft $foc\_real \neq nil$ gilt, dann konnte eine Verfeinerung des Adaptionspunktes i vorgenommen werden; sonst konnte eine automatische Adaptionspunktrealisierung nicht durchgeführt werden.

procedure $konfig(APM(i)) \leftarrow$

while $\exists\ \langle c\_s, foc\_s, foc\_r, c\_e\rangle_j \in APM(i)$ mit $c\_s \neq nil$ **and**
i möglichst hohe Priorität bzgl. $<_{AP}$ **and**
$\exists\ R \in ETeS_{Adapt}$ mit $applicable(R, foc\_s)$

[44] Initial entsprechen sich 1. und 2.

do if $c\_e = nil$

then $\langle\langle c\_s, foc\_s', foc\_r', c\_e\rangle_j, EXT\rangle := apply(R, \langle c\_s, foc\_s, foc\_r, c\_e\rangle_j)$
$APM(i) := (APM(i) \setminus \{\langle c\_s, foc\_s, foc\_r, c\_e\rangle_j\}) \cup \{\langle c\_s, foc\_s', foc\_r', c\_e\rangle_j\}$
$integriere(EXT, APM(i))$

else if c_e unzulässig bzgl. $Kombi_{Adapt}$

then abort

else if $and(c\_s, c\_e)$ zulässig bzgl. $Kombi_{Adapt}$

then $APM(i) := (APM(i) \setminus \{\langle c\_s, foc\_s, foc\_r, c\_e\rangle_j\}) \cup$
$\{\langle and(c\_s, c\_e), and(foc\_s, c\_e), foc\_r, nil\rangle_j\}$

else $APM(i) := (APM(i)\setminus\{\langle c\_s, foc\_s, foc\_r, c\_e\rangle_j\})\cup\{\langle c\_e, c\_e, nil, nil\rangle_j\}$

od

if $\{\langle c\_s, foc\_s, foc\_r, c\_e\rangle_i \in APM(i)$ mit $foc\_r \neq nil$

then return

else abort

Das Prädikat $applicable(R, focus)$ überprüft, ob eine Taktikumsetzungsregel auf den aktuellen Merkmalsfokus des entsprechenden Adaptionspunktes vollständig anwendbar ist.

predicate $applicable(R, focus) \leftarrow$

case R **of**

$feature\_tac(focus) \rightarrow *$ und $* :$
return $true$

$feature\_tac(focus) \rightarrow * :$
return $true$

$current\_features(A) \rightarrow m \Rightarrow nec\_subterm(m) \rightarrow m' \Rightarrow feature\_tac(focus) \rightarrow * :$

if $\langle m, fs, fr, ce\rangle_A \in APM(i)$ **and** $nec\_subterm(m) \rightarrow m'$ ableitbar in $Adapt$
then return $true$
else return $false$

$c \rightarrow t \Rightarrow feature\_tac(focus) \rightarrow * :$

if $c \rightarrow t$ ableitbar in $Adapt$
then return $true$
else return $false$

esac

return $false$

Die Funktion $apply(R, \langle c\_s, foc\_s, foc\_r, c\_e\rangle_j)$ verändert nun den Fokus des Adaptionspunktes j gemäß der Regel R.

function $apply(R, \langle c\_s, focus, nil, c\_e\rangle_x) \leftarrow$

case R **of**

$feature\_tac(focus) \rightarrow T$ und Ext **:**
return $\langle\langle c\_s, nil, T, c\_e\rangle_x, Ext\rangle$

$feature\_tac(focus) \rightarrow feature\_tac(m)$ **:**
return $\langle\langle c\_s, m, nil, c\_e\rangle_x, nil\rangle$

$feature\_tac(focus) \rightarrow T$ **:**
return $\langle\langle c\_s, nil, T, c\_e\rangle_x, nil\rangle$

$* \rightarrow * \Rightarrow * \rightarrow * \Rightarrow feature\_tac(focus) \rightarrow T$ **:**
return $\langle\langle c\_s, nil, T, c\_e\rangle_x, nil\rangle$

$* \rightarrow * \Rightarrow feature\_tac(focus) \rightarrow T$ **:**
return $\langle\langle c\_s, nil, T, c\_e\rangle_x, nil\rangle$

esac

Die Prozedur $integriere(EXT, APM(i))$ verteilt die in EXT vorhandenen **extend**-Anweisungen auf die entsprechenden Adaptionspunkte.

procedure $integriere(EXT, APM(i)) \leftarrow$

forall extend$(AP_i, spec_i) \in EXT$

do $\langle cs, fs, fr, ce\rangle_i \in APM(i)$
if $ce = nil$
then $ce' := spec_i$
else $ce' := and(ce, spec_i)$
$APM(i) := (APM(i) \setminus \{\langle cs, fs, fr, ce\rangle_i\}) \cup \{\langle cs, fs, fr, ce'\rangle_i\}$

od

Die Initialisierung der Menge $APM(i)$ wird aktuell getriggert durch die Realisierungsnotwendigkeit eines Merkmalsfokus m_i. Sie geschieht gemäß

$$APM(i) := \{\langle c\_s, c\_s, nil, nil\rangle_a | current\_features(a) \rightarrow c\_s \text{ Axiom in } Tacadapt,\ a \neq i\} \cup \{\langle c\_s, m_i, nil, nil\rangle_i | current\_features(i) \rightarrow c\_s \text{ Axiom in } Tacadapt\}$$

als Resultat einer Prozedur $init\_APM(m\_i)$.
Wenn nun die Prozedur $konfig$ zu einer erfolgreichen Verfeinerung des aktuellen Merkmalsfokus m_i führt, so können die entsprechenden Axiome zur Anpassung von $Tacadapt$ leicht aus $APM(i)$ extrahiert werden; die Funktion $genKAX$ bildet die Axiome:

function $genKAX(APM(i)) \leftarrow$
Tmp:= {}

for each $\langle c\_s, foc\_s, foc\_r, c\_e\rangle_j \in APM(i)$

do if $i = j$

then $Tmp := Tmp \cup \{feature\_tac(m_i) \rightarrow foc\_r\} \cup \{current\_features(i) \rightarrow c\_s\}$

else $Tmp := Tmp \cup \{current\_features(j) \rightarrow c\_s\}$

od

return Tmp

Die entstandenen Axiome müssen nun für eine Aktualisierung von $Tacadapt$ verwendet werden. Durch die inkrementelle Realisierung jeweils nur eines aktuellen Merkmales, wird erreicht, daß während der Berechnung eines Taktikmodules jeweils aktuell angepaßte Adaptionspunktrealisierungen vorliegen.
Als Erweiterung von Definition 3.3.2 kann nun definiert werden, was ein *konfigurierbares* Taktikmodul ist.

Definition 3.4.2 *Ein konfigurierbares Taktikmodul* $\mathbf{KTM}_{tac}$ *ist beschrieben durch ein Tupel*

$$\langle \mathbf{TM}_{tac}, \Sigma_{Adapt}, Kombi_{Adapt}, ETeS_{Adapt}, <_{AP}\rangle,$$

wobei $\mathbf{TM}_{tac}$ *ein Taktikmodul bzgl.* $\Sigma_{Tackonf}$ *ist.*

Die Erweiterung auf konfigurierbare Taktikmodule erfordert auch eine Anpassung der Berechnungsstrategie zur Umsetzung einer ausführbaren Spezifikation (vgl. dazu die Berechnung eines Taktikmoduls auf Seite 73). Die Basis für eine Berechnung bildet der aktuelle Zustand eines *dynamischen* Termersetzungssystems $DTeS$, das den in der Spezifikation $Tacadapt$ verfügbaren Axiomen entspricht. $DTeS$ enthält eine statische Teilmenge $DTeS_{stat}$ von Axiomen, die der Spezifikation $Adapt$ entstammen. Entsprechend bezeichnet $DTeS_{dyn} = DTeS \setminus DTeS_{stat}$ den dynamischen Teil von $DTeS$.
Um eine Berechnung durchführen zu können, ist nun ein Wechsel zwischen drei verschiedenen Phasen notwendig:

1. einer Termersetzungsphase zur Reduktion von Taktiken,
2. einer PROLOG-Zielauswertungsphase und
3. einer Merkmalrealisierungsphase, die zu einer Veränderung des zugrundeliegenden Termersetzungssystems $DTeS$ führt.

Gegeben sei eine Spezifikation der Adaptionspunkte eines konfigurierbaren Taktikmoduls $\mathbf{KTM}_{tac}$ als die Menge von Paaren

$$\{(AP_1, spec_1), \ldots, (AP_n, spec_n)\}.$$

Die Initialisierung von $DTeS$ erfolgt dann als

$$\begin{aligned} DTeS := DTeS_{stat} \quad & \cup \{choicetac(AP_i) \rightarrow feature\_tac(spec_i) \mid i = 1, \ldots, n\} \\ & \cup \{current\_features(AP_i) \rightarrow spec_i \mid i = 1, \ldots, n\} \end{aligned}$$

Die folgende Funktion *keval* beschreibt die verwendete Berechnungstrategie für partiell konfigurierte Taktiken und koordiniert die oben genannten drei Phasen:[45]

function $keval(tac) \leftarrow$

1. reduziere tac durch die Anwendung von $DTeS$ zu tac' bis gilt
2. **case** tac' **of**

$basic$**:**
%% $basic$ ist entweder eine der Funktionen rel_* (außer $rel\_feature\_tac$) oder
%% die Funktion $true$, $fail$, $append$
if $PROLOG\text{-}eval(basic) = \langle true, * \rangle$ **then return**$(true)$ **else return**$(false)$

$(basic, Rest)$**:**
%% $basic$ ist entweder eine der Funktionen rel_* (außer $rel\_feature\_tac$) oder
%% die Funktion $true$, $fail$, $append$
if $PROLOG\text{-}eval(basic) = \langle true, * \rangle$ **then return**$(keval(\sigma(Rest)))$ **else return**$(false)$

$(if(a, m_u^i, m_v^i), Rest)$**:**
%% m_u^i und m_v^i sind Merkmale gemäß $T(\Sigma_{Adapt})$
if $eval(a) = true$ **then return**$(keval((feature\_tac(m_u^i), Rest)))$
else return$(keval((feature\_tac(m_v^i), Rest)))$ [46]
return$(keval(tac'))$

$(feature\_tac(m_j^i), Rest)$**:**
%% m_j^i ist Merkmal gemäß $T(\Sigma_{Adapt})$ und es existieren konditionale
%% Ersetzungsregeln für m_j^i
$konfig(init\_APM(m_j^i))$
$DTeS := axiom\_update(genKAX(APM(i)) \backslash \{feature\_tac(m_j^i) \rightarrow T\}, DTeS)$
return$(keval((T, Rest)))$[47]

$(feature\_tac(m_j^i), Rest)$**:**
%% m_j^i ist Merkmal gemäß $T(\Sigma_{Adapt})$ und es existieren keine konditionalen
%% Ersetzungsregeln für m_j^i
$konfig(init\_APM(m_j^i))$
$DTeS := axiom\_update(genKAX(APM(i)), DTeS)$
return$(keval(tac'))$

$(rel\_feature\_tac(m_j^i, tac), Rest)$**:**
%% m_j^i ist Merkmal gemäß $T(\Sigma_{Adapt})$ und es existieren konditionale
%% Ersetzungsregeln für m_j^i
$konfig(init\_APM(m_j^i))$
$DTeS := axiom\_update(genKAX(APM(i)) \backslash \{feature\_tac(m_j^i) \rightarrow T\}, DTeS)$
$Rest' := \sigma(Rest)$, mit der Substitution $\sigma = \{T/tac\}$[48]
return$(keval(Rest'))$

[45] Verwendet wird dabei auch die Funktion *eval* von Seite 74.

[46] Damit ist es möglich, *nec_subterm*-Anfragen auf selbstadaptive Adaptionspunkte zu behandeln.

[47] Mit diesem Vorgehen erreicht man, daß das Merkmal m_j^i bei jeder Interpretation neu realisiert wird und sich damit an möglicherweise veränderte Umstände anpassen kann.

[48] Da das Symbol *tac* die Rolle einer Variable übernimmt, kann man hier von Substitution sprechen.

$(rel\_feature\_tac(m_j^i, tac), Rest)$**:**
$\%\%$ m_j^i ist Merkmal gemäß $T(\Sigma_{Adapt})$ und es existieren keine konditionalen
$\%\%$ Ersetzungsregeln für m_j^i
$konfig(init\_APM(m_j^i))$
$DTeS := axiom\_update(genKAX(APM(i)), DTeS)$ und $\exists feature\_tac(m_j^i) \rightarrow T \in genKAX(APM(i))$
$Rest' := \sigma(Rest)$, mit der Substitution $\sigma = \{T/tac\}$
return$(keval(Rest'))$

$if(a, b, c)$**:**
if $eval(a) = true$ **then return**$(keval(b))$ **else return**$(keval(c))$

$(if(a, b, c), Rest)$**:**
if $eval(a) = true$ **then return**$(keval((b, Rest)))$ **else return**$(keval((c, Rest)))$

endcase

Die Funktion *keval* verwendet die Funktion *axiom_update*, die wie folgt definiert ist.

function $axiom\_update(S, DTeS) \leftarrow$

if $S = \{\}$

then return DTeS

else $S := S \setminus A$

case A **of**

$current\_features(a) \rightarrow m$ **:**
$DTeS := (DTeS \setminus \{current\_features(a) \rightarrow *, choicetac(a) \rightarrow feature\_tac(*)\}) \cup \{A, choicetac(a) \rightarrow feature\_tac(m)\}$

$feature\_tac(m) \rightarrow T$ **:**
$DTeS := (DTeS \setminus \{feature\_tac(m) \rightarrow *\}) \cup A$

esac

return $axiom\_update(S, DTeS)$

In Abbildung 3.7 ist nun zusammenfassend dargestellt, wie mit konfigurierbaren Taktiken gearbeitet werden kann.
Einmalig festzulegen sind dabei

- die Definition der Merkmalssprache, d.h. welche Merkmale gibt es zur Realisierung von welchem Adaptionspunkt und welche Kombinationen von Adaptionspunkten sind erlaubt (dargestellt durch $Kombi_{Adapt}$);
- die Umsetzung von Merkmalen in Taktiken und Anweisungen zur Erweiterung anderer Merkmalspezifikationen (zusammengefaßt im Ersetzungssystem $ETeS_{Adapt}$);
- eine Ordnung zum Ausdruck von Prioritäten zwischen den einzelnen Adaptionspunkten (gegeben durch $<_{AP}$);

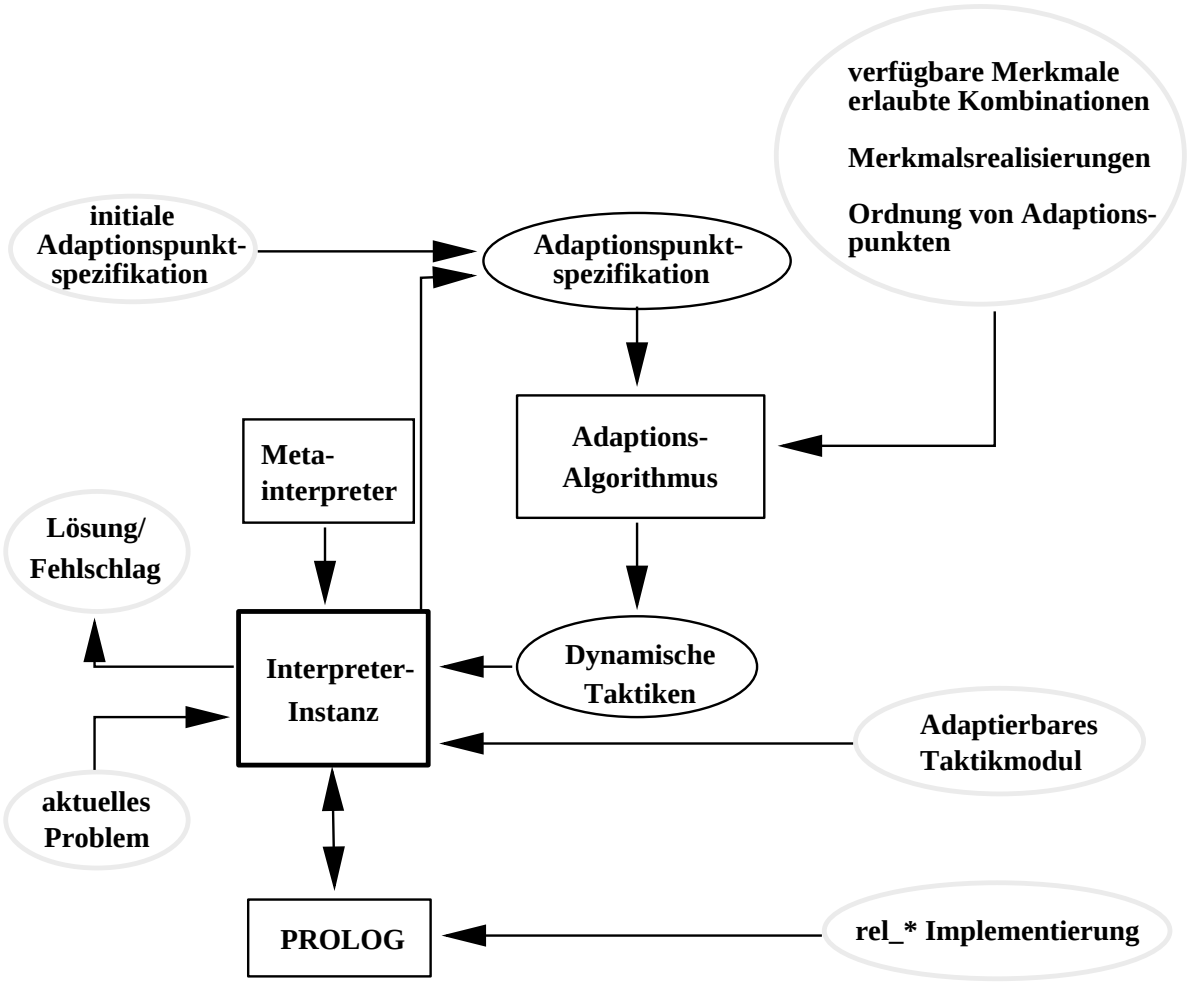


Abbildung 3.7: Konfigurierbare Taktiken

- schließlich ein Taktikmodul, das Adaptionspunkte enthält.

Durch die Vorgabe einer aktuellen Adaptionspunktspezifikation charakterisiert man nun eine spezifische Variante einer Planungstaktik. Der Adaptionsalgorithmus kann dann eine möglicherweise dynamische Menge von Taktiken zur Ergänzung des vorgegebenen Taktikmoduls erzeugen. Ein Interpreter, der die Berechnungsstrategie realisiert, führt mit dem resultierenden dynamischen Taktikmodul bzgl. eines aktuellen Problems eine Berechnung durch. Während dieser Berechnung kann es zu erneuten Taktikadaptionsphasen kommen, bevor letztendlich eine Problemlösung oder ein Fehlschlag resultieren. Die grundsätzliche Voraussetzung für die Durchführung einer Berechnung ist das Vorhandensein von PROLOG-Implementierungen aller Funktionen rel_*.

Die Gesamtstruktur der Sprache

Abbildung 3.8 zeigt einen Überblick, wie die algebraischen Spezifikationen zur Beschreibung konfigurierbarer Taktiken zusammenhängen. Durch die Implementierungsbeziehungen wird die Eigenschaft der Ausführbarkeit von Taktikspezifikationen herbeigeführt.

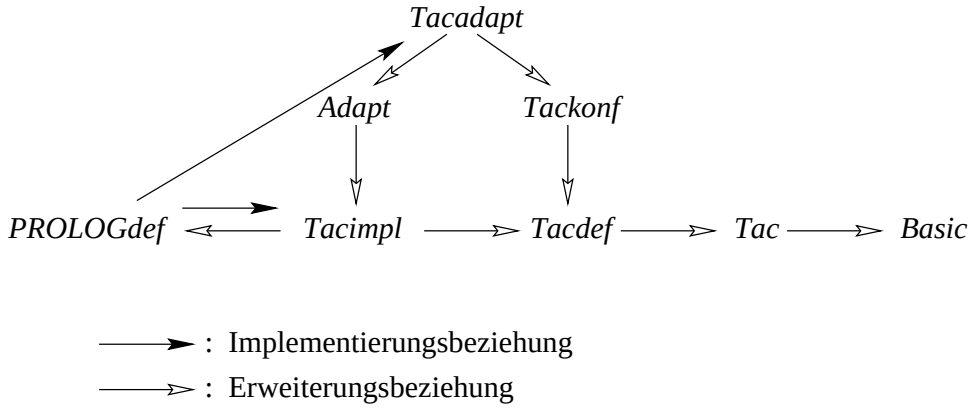


Abbildung 3.8: Zusammenhang der einzelnen Spezifikationen

Die Spezifikation *PROLOGdef* kann unmittelbar in die Programmiersprache *PROLOG* umgesetzt werden und bildet die Basis für die Ausführbarkeit der Taktikspezifikationen.

3.4.2 Kontrollstrategien als Metaprogramme

Wenn man von dem Konzept einer Kontrollstrategie einmal absieht, so könnte man den Planungsprozeß im Prinzip auch alleine auf der Basis der Implementierung von Planungszustandübergangsfunktionen – repräsentiert durch die Funktionen rel_basic_* – ausführen. Diese Funktionen sind auf der Implementierungsebene – in unserem Falle die logische Programmiersprache PROLOG – realisiert durch entsprechende logische Programme rel_basic_*. Man kann nun einen einfachen Interpreter in PROLOG spezifizieren, der ausgehend von einem initialen Planungszustand versucht, durch die Verwendung von rel_basic_*-Programmen einen Planungsalgorithmus (vgl. Abschnitt 3.2) zu realisieren. Allein unter Verwendung der PROLOG-eigenen, starren Berechnungsstrategie ist dieses Vorgehen im allgemeinen nur bedingt erfolgreich und wenn, dann nur unter

enormem Backtracking-Aufwand. Ein Ausweg liegt darin, Einfluß auf die Berechnungsstrategie zu nehmen, d.h. von einer starren hinüber zu einer selbst definierbaren Berechnungsstrategie zu gehen. Genau dies erreicht man mit der Verwendung einer Planungstaktik. Sie beschreibt letztendlich, welche rel_basic_*-Programme in welcher Situation verwendet werden, stellt also eine Berechnungsstrategie dar. Die Umsetzung einer Planungstaktik geschieht demzufolge durch die Realisierung eines entsprechenden PROLOG-*Metaprogrammes*[49]. Durch die Realisierung konfigurierbarer Taktiken ergeben sich insgesamt drei Sprachebenen (siehe Abbildung 3.9).

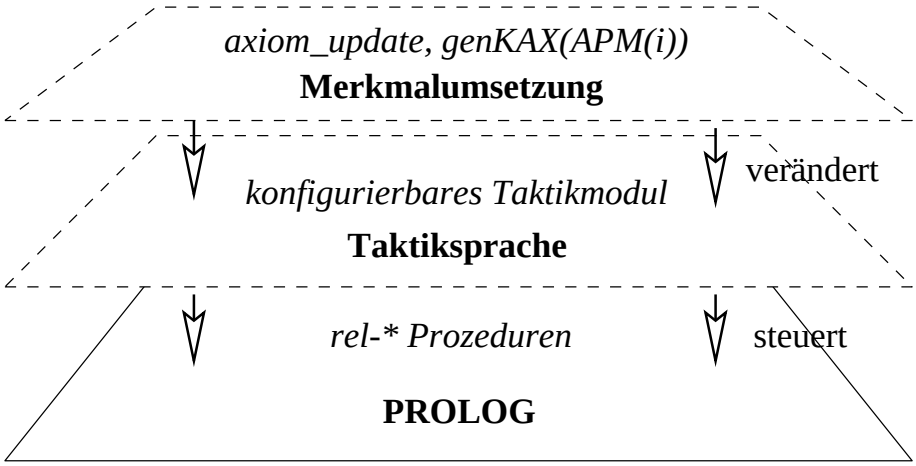


Abbildung 3.9: Drei Sprachebenen einer Kontrollstrategie

Auf der Objektebene befindet sich PROLOG mit seinen *built-in*-Prädikaten und den Realisierungen der Funktionen rel_*. Auf der ersten Metaebene befindet sich ein Interpreter, der die Taktiksprache gemäß der Spezifikation *Tacadapt* umsetzt, d.h. die Axiome aus *Tacadapt* werden durch geeignete Programmklauseln realisiert; dieses Vorgehen wird durch die Verwendung gerichteter Gleichungen bei der Formulierung der Axiome erleichtert. Auf der darüberliegenden zweiten Metaebene wird die Adaptionspunktrealisierung durchgeführt. Eine Interpretation auf dieser Ebene bewirkt eine Veränderung des Metaprogrammes auf der ersten Metaebene.

Die Umsetzung des dynamischen Termersetzungssystems in ein Metaprogramm

Der aktuelle Zustand des verwendeten dynamischen Ersetzungssystems *DTeS* entspricht der aktuellen Spezifikation eines Metaprogrammes *reduce*, das eine Realisierung der Berechnungsstrategie *keval* darstellt. Die Programmklauseln von *reduce* teilen sich entsprechend der Aufteilung von *DTeS* in einen statischen und einen dynamischen Teil. Die Realisierung des statischen Teiles wird exemplarisch durch die Umsetzung typischer Axiome gemäß *Tacadapt* gezeigt. Die Umsetzung der drei folgenden Axiome gibt bereits einen Einblick in die Verbindung von erster Metaebene und Objektebene.

[49]Zur allgemeinen Vorgehensweise und Anwendung der Metaprogrammierung im Kontext logischen Programmierens siehe [Neumann 88, Hill & Gallagher 96].

axioms

then(x,a,b) → then2(applytac(hd(x),a,c),then(tl(x),c,b))
then2(t1,t2) → (t1 , t2)
rel_basic_i(s,sl) → (rel_selfunc$_j$(x,?y),rel_basic_i(s',sl))[50]

für alle rel_basic_i(s,sl) mit selfunc$_j$(x) ist Unterterm in s und $s' = \sigma(s)$ mit $\sigma = \{?y/selfunc_j(x)\}$

Diese Axiome werden durch die drei folgenden Programmklauseln von `reduce` realisiert:

```
reduce(then(X,A,B)):-
       reduce(then2(applytac(hd(A),A,C),then(tl(A),C,B))).

reduce(then2(A,B)):-
       reduce(A),
       reduce(B).

reduce(P):-
       functor(P,R,_),
       member(R,[rel_basic1,...,rel_basicN]),
       change_subterm(selfuncj(X),P,Y,PNew),
       rel_selfuncj(X,Y),
       reduce(PNew).
```

Bei der Realisierung des dritten Axioms wurden bereits einige Abstraktionen eingearbeitet. Das Schema arbeitet für alle Funktionen rel_basic_* und alle darin möglichen Vorkommen der Funktion $selfunc_j$. Hier ist auch sichtbar, wie der *flattening*-Prozeß zur Evaluierung von Funktionen durchgeführt werden kann.
Geht man analog für die übrigen Axiome aus $DTeS_{stat}$ vor, so erhält man ein Metaprogramm, das Punkt 1. der Beschreibung von *keval* (vgl. Seite 3.4.1.1) umsetzen kann.
Die einzelnen Fälle von Punkt 2. in *keval* können ebenfalls auf einfache Art umgesetzt werden (teilweise auch zusammengefaßt werden):

```
reduce(rel_basic1(A,B)):-
       rel_basic1(A,B).

reduce((rel_basic1(A,B),Rest)):-
       rel_basic1(A,B),
       reduce(Rest).
```

Man erkennt hier, daß ein und dasselbe Symbol – `rel_basic1` – unterschiedlichen syntaktischen Kategorien angehören kann. Als Metaprogrammsymbol – Argument von `reduce` – ist es eine Funktion und als Objektsymbol repräsentiert es ein Prädikat. Damit läßt sich die auf Seite 72 erwähnte Umsetzung boolscher Funktionen in PROLOG-Prädikate auf natürliche Weise durchführen. Für alle rel_*-Funktionen und analog für die Funktionen $true$, $fail$, und *append* müssen `reduce`-Klauseln, wie oben beschrieben, spezifiziert werden.

[50] Realisiert wird hierdurch ein *flattening*-Prinzip, ein Standardvorgehen bei der Integration von Funktionalen Sprachen in logische Programmiersprachen (vgl. [Hanus 94]).

```
reduce((if(A,M1,M2),Rest)):-
       (reduce(A) -> reduce(M1)
                     ;
                     reduce(M2)),
       reduce(Rest).
```

Diese Klausel umfaßt sowohl den Fall, daß `M1` bzw. `M2` Merkmale sind als auch den Fall, daß sie bereits Taktiken darstellen.

```
reduce(feature_tac(M)):-
       ist_merkmal(M),
       kond_real_moeglich(M),
       generiere_reduce_programm(M,mitKER,P),
       delete((reduce(M):- T),P,P'),
       aktualisiere_reduce_programm(P'),
       reduce(T).

reduce(feature_tac(M)):-
       ist_merkmal(M),
       generiere_reduce_programm(M,ohneKER,P),
       aktualisiere_reduce_programm(P),
       reduce(M).
```

Die beiden `reduce`-Klauseln enthalten nun Metaprädikate der zweiten Metaebene (vgl. Abbildung 3.9). `ist_merkmal` entscheidet, ob sein aktueller Parameter ein Merkmal ist oder nicht. `kond_real_moeglich` überprüft, ob zu dem Merkmal eine konditionale Ersetzungsregel existiert.
`generiere_reduce_programm` übernimmt die Funktion von $genKAX(APM(i))$, d.h. Axiome bzw. `reduce`-Klauseln zur Realisierung des Merkmals `M` zu bilden. Die Aufgabe von *axiom_update* wird von dem Metaprädikat `aktualisiere_reduce_programm` übernommen. `delete` bezeichnet ein PROLOG-Prädikat zur Entfernung eines Termes aus einer Struktur. Eine `reduce`-Klausel, die die Umsetzung eines Merkmals m unmittelbar beschreibt, hat die Form `reduce(`m`):-`*tac*. Wenn sie im `reduce`-Programm enthalten sein soll, so muß dafür gesorgt sein, daß sie vor den beiden oben zuletzt beschriebenen `reduce(M)`-Klauseln im Programmtext spezifiziert sein muß.[51] Das `reduce`-Metaprogramm ist dann so strukturiert, daß nach den Klauseln für statische Axiome zur Umsetzung von *keval*1 die Klauseln zur *choicetac*- bzw. *feature_tac*-Umsetzung und die Klauseln zur Realisierung von *keval*2 folgen:

3.4.3 Die Planerverfeinerung

Um nun durch die Spezifikation eines konfigurierbaren Taktikmoduls einen Planer zu definieren, kann z.B. das in Abbildung 3.4 eingeführte allgemeine Planerschema verwendet werden. Im Sinne eines *Top-Down*-Entwurfs legt man zunächst die generische Struktur

[51] Dies liegt an der prozeduralen Interpretation von PROLOG; genaugenommen lautet die Klausel `reduce(`m`):-` !, *tac*.

der Planungsstrategie fest. Für die eingeführten Adaptionspunkte müssen dannach die möglichen Konkretisierungen spezifiziert werden. Einen Ausschnitt der dabei entstehenden rel_*-Implementierungen gemäß der Spezifikation *Tacadapt* ist im folgenden zu sehen:

```
rel_tacdef(planer,then(cons(calltac(generiere),
                        cons(choicetac(plantrans),empty))))

rel_tacdef(generiere,orelse(cons(gentac(nochoffeneProbleme(S),SL),
                             cons(then(empty),empty))))

rel_nochoffeneProbleme(S,cons(choicetac(foc),empty)):-
         <Body(S)>

rel_feature_tac(foc_standard,
                selectapply_X(orelse(cons(choicetac(split),
                                          cons(choicetac(probtrans),
                                           cons(choicetac(imp),empty)))),
                              calltac(generiere),
                              S,SL))

rel_feature_tac(imp_standard,
                then(cons(calltac(vorverarbeitung),
                      cons(choicetac(act),empty))))
```

Die Entscheidungsfunktion `nochoffeneProbleme` aktiviert (indirekt) die mit dem Adaptionspunkt AP_{foc} assoziierte Taktik, wenn bzgl. des Argumentes `S` evaluiert ist, daß noch offene Probleme bestehen. Die Operationalisierung dieser Entscheidung ist symbolisiert durch `<Body(S)>`, d.h. den durch `S` parametrisierten Rumpf der Prozedur `rel_nochoffeneProbleme`. `foc_standard` sei ein Merkmal, das den Adaptionspunkt AP_{foc} realisiert. Das zugehörige Taktikschema enthält wiederum andere Adaptionspunkte. Die assoziierte Funktion $selfunc_X$ sorge für die Selektion des ersten offenen Zieles in der Liste `S`. `imp_standard` sei ein Merkmal zur Realisierung von AP_{imp}.
Die hier verwendeten Adaptionspunkte werden von einem allgemeinen (nicht konfigurierbaren) Planer auf die ein oder andere Art konkretisiert.[52] Es liegt aber keine Beschränkung vor, nicht noch weitere Adaptionspunkte definieren zu dürfen, um damit z.B. die hier verwendeten noch weiter zu strukturieren.

Wie zu Beginn dieses Kapitels erwähnt, besteht im Kontext von Assistenzsystemen die Forderung, in Abhängigkeit der aktuellen Plankonsumentenanforderung, Lösungen für Planungsaufgaben flexibel zu erstellen. Um der damit einhergehenden Notwendigkeit nach flexiblen Kontrollstrategien gerecht zu werden, wurde das allgemeine Konzept konfigurierbarer Kontrollstrategien eingeführt. Es erlaubt, adaptierbare Planungstaktiken in Form ausführbarer abstrakter Spezifikationen exakt zu definieren und in Form logischer Programme zu implementieren. Dieses Modell ist soweit noch weitgehend unabhängig von

[52] Einige Adaptionspunkte werden vielleicht im Sinne einer Identitätsabbildung als irrelevant konkretisiert.

einem konkreten Planrepräsentations- und Planungsmechanismus. Da Kontrollstrategien unterschiedlichste Merkmalsrealisierungen vielfältig kombinieren können, muß der zugrundeliegende Planrepräsentationsmechanismus jedoch dieser Flexibilität auch folgen können. Von dem Planungsmechanismus wird verlangt, daß er Strukturen im Sinne von Planungszuständen (vgl. Definition 3.2.1) manipuliert.
Im folgenden Kapitel wird nun ein Planrepräsentations- und Generierungsformalismus vorgestellt, der die nötige Flexibilität in der Darstellung und größtmögliche Freiheit bei der Planerstellung gewährleistet.

Kapitel 4

Ein deduktiver Planungsansatz

In diesem Kapitel wird ein deduktiver Planungsansatz – Planen geschieht durch formallogisches Beweisen – vorgestellt, dessen Flexibilität es erlaubt, Pläne mit unterschiedlichen Abstraktionsgraden zu beschreiben und automatisch zu erzeugen. Die zugrundeliegenden *Planspezifikationen* formalisieren die Anfangsbedingungen und zu erreichenden Ziele eines Planes. Die Spezifikation der Ziele beschränkt sich nicht nur auf die Charakterisierung eines einzigen Endzustandes, sondern auch komplexe zeitliche Beziehungen zwischen Teilzielen können beschrieben werden. Ebenso können neben den zustandsbasierten Vorbedingungen auch zeitlich ausgedehnte *Sicherheitsbedingungen* als Teil einer Planspezifikation definiert werden.
Der Planungsansatz zeigt seine Variabilität aber auch seine Differenzierbarkeit sowohl hinsichtlich der Repräsentation von Plänen als auch den unterschiedlichen Prozessen, die sie erzeugen. So wird gezeigt, daß alle klassischen Planverfeinerungsmethoden, d.h. zustandsbasierte, planbasierte und Aufgabenreduktions[1]-Verfeinerung, in beliebiger Kombination in einer Planungsstrategie vereinigt werden können, zusätzlich aber auch neue Arten abstrakterer Planverfeinerung spezifizierbar sind (vgl. zusammenfassend auch [Dengler 96a]).

Die Basis eines deduktiven Ansatzes bildet eine formale Logik. *Berechnungen* – oder besser Beweise – werden mit Hilfe eines *Kalküls* für diese Logik durchgeführt. Der erste Abschnitt widmet sich diesen Grundlagen.
Der zweite Abschnitt zeigt, wie man Aktionen und Pläne repräsentiert und unter welchen Voraussetzungen man den Kalkül zum Planen verwenden kann. Planen bedeutet dann, Planspezifikationen mit Hilfe des Kalküls zu beweisen. Anhand zahlreicher Beweisstrategien wird belegt, wie Pläne, gekennzeichnet durch differenzierte Abstraktion, erzeugt werden können. Es werden hier die Basismechanismen beschrieben, mit denen die im letzten Kapitel beschriebenen Adaptionspunktrealisierungen erreicht werden können. Die Beweisstrategien, die notwendig sind, um diese Mechanismen zielgerichtet abzubilden, können als Instanzen der dort eingeführten Kontrollstrategien realisiert werden. In [Dengler 94b, Dengler 94a] sind wesentliche Aspekte des Zusammenwirkens adaptierbarer Kontrollstrategien und des im folgenden diskutierten deduktiven Planungsansatzes beschrieben.

[1] *Engl. HTN(hierarchical task network)-refinement.*

4.1 LLP – Eine Logische Planungssprache

Die logische Planungssprache, die als Repräsentationsformalismus Verwendung findet ist LLP[2] [Biundo et al. 92a], eine intervallbasierte modale Temporallogik. Die Grundbausteine der Logik werden gebildet durch eine Kombination von wesentlichen Merkmalen einer linearen diskreten Temporallogik (vgl. [Manna & Pnueli 81]), einer kompositionalen Prozeßlogik (vgl. [Rosner & Pnueli 86]) und denen einer diskreten temporalen Programmlogik (siehe [Kröger 87]). Intervalle werden dabei betrachtet als zeitliche Folgen von Zuständen, d.h. es besteht eine totale Ordnung zwischen den Zuständen.
LLP ist eine sortierte Logik erster Ordnung mit Gleichheit und enthält neben den normalen *logischen Variablen*, auch *globale Variablen* genannt, eine Menge von *lokalen Variablen*. Diese entsprechen vom Konzept her den Programmvariablen in Programmlogiken und haben die wesentliche Eigenschaft, daß der ihnen zugewiesene Wert auch beliebig verändert werden kann; eine Quantifizierung über lokale Variablen ist nicht definiert.
Als Modallogik enthält LLP eine Reihe von temporalen Modaloperatoren zum Ausdruck von Aussagen über bestimmte Zeitintervalle. Im einzelnen sind dies die Operatoren:

- $\bigcirc$ (*next*): ausgehend vom aktuellen Zustand macht man eine Aussage über das Intervall, das mit dem nächsten Zustand beginnt;
- $\Diamond$ (*sometimes*): man macht aktuell eine Aussage über ein Intervall, das irgendwann beginnt;
- $\Box$ (*always*): es wird eine Aussage über die Intervalle gemacht, die in jeweils allen zukünftigen Zuständen beginnen;
- **;** (*chop*): die Zukunft wird unterteilt in eine Folge von zwei Zeitabschnitten; es erfolgt eine Aussage darüber, was in dem aktuell beginnenden Abschnitt gilt und was in dem darauffolgenden gilt.

Verschiedene Kontrollstrukturen lassen sich in LLP definieren,[3] z.B.:

- Das Konditional $\underline{if}\ \epsilon\ \underline{then}\ \alpha\ \underline{else}\ \beta$
  ist definiert durch die Formel $[\epsilon \rightarrow \alpha] \wedge [\neg\epsilon \rightarrow \beta]$.
- Eine *while*-Schleife kann definiert werden durch ein Axiom der Art:
  $\underline{while}\ \epsilon\ \underline{do}\ \alpha\ \underline{od}\ ;\beta \leftrightarrow [\underline{if}\ \epsilon\ \underline{then}\ [\alpha\ ;\underline{while}\ \epsilon\ \underline{do}\ \alpha\ \underline{od}\ ;\beta]\ \underline{else}\ \beta]$.

4.1.1 Die Syntax

Die Sprache LLP wird definiert beginnend mit einer S-Signatur $\Sigma = \Sigma^F \cup \Sigma^P$, wobei S eine nichtleere Menge von Sortensymbolen benennt:

$$\Sigma^F = (\Sigma^F_{ws})_{w\in S^*,\, s\in S} \text{ und } \Sigma^P = (\Sigma^P_v)_{v\in S^*},$$
$$\text{wobei } \{ex\} \subseteq \Sigma^P_{action} \text{ und } \{T\} \subseteq \Sigma^P_\epsilon.$$

[2] Ein Akronym für **L***ogical* **L***anguage for* **P***lanning*.
[3] Das durch sie induzierte zeitliche Verhalten wird abgebildet.

Für $f \in \Sigma^F_{ws}$ ist ws der Rang, w die Stelligkeit und s die Sorte von f. Für $p \in \Sigma^P_v$ ist v die Stelligkeit von p.
VG_s und VL_s bezeichnen die Mengen der globalen bzw. lokalen Variablen der Sorte s, $V_s = VG_s \cup VL_s$ ist die Menge aller Variablen der Sorte s; V ist entsprechend die Menge aller Variablen.
Die Menge der Σ-Terme der Sorte s erhält man standardmäßig:

- für jede Variable $v \in V_s$ gilt $v \in \mathcal{T}(\Sigma)_s)$;
- wenn $f \in \Sigma^F_{ws}$ eine n-argumentige Funktion ist und $t_1, \ldots, t_n$ Terme von entsprechender Sorte aus $\mathcal{T}(\Sigma)_s)$ sind, dann ist $f(t_1, \ldots, t_n)$ auch in $\mathcal{T}(\Sigma)_s$.

$\mathcal{T}_\Sigma = (\mathcal{T}(\Sigma)_s)_{s \in S}$ sei die Menge aller Σ-Terme.
Die Menge $\mathcal{F}_\Sigma$ der Σ-Formeln wird standardmäßig mit Hilfe der Operatoren $\{\neg, \wedge, \forall, \bigcirc, \Box, \ ;\}$ gebildet. Es gelten die Abkürzungen:

$$\begin{aligned} \Diamond\phi &\leftrightarrow \neg\Box\neg\phi \\ [\phi \rightarrow \psi] &\leftrightarrow [\neg[\phi \wedge \neg\psi]] \\ \neg T &\leftrightarrow F \end{aligned}$$

Ein als $\odot$ (*strong next*) bekannter Operator ist definiert durch $\odot\phi \leftrightarrow \neg\bigcirc\neg\phi$.
LLP enthält ein ausgezeichnetes einstelliges Prädikat ex mit Stelligkeit *action*, das als Argument einen Aktionsterm aufnehmen kann.[4]

Die Signatur von LLP kann erweitert werden um metalogische Variablen- und Propositionssymbole, die Platzhalter für LLP-Terme bzw. LLP-Formeln eines bestimmten Types darstellen.[5] Gehandhabt werden diese Metasymbole zunächst wie gewöhnliche Symbole in LLP. Die Logik, die sich durch die Signaturerweiterung ergibt, nennen wir LLP*. Die Propositionssymbole übernehmen auch die Aufgaben von *Metavariablen* (vgl. z.B. [Paulson 90, Heisel et al. 90, Felty & Howe 94]). In konstruktiven Beweisen verwendet man diese Formelmetavariablen z.B. für Formeln, die zu Beginn eines Beweises noch nicht bekannt sind; nur ihre Beziehung zu anderen Formeln ist spezifiziert. Die Metavariablen beschreiben in diesem Sinne eine nichtdeterministische Auswahl einer Formel, die erst zu einem späteren Zeitpunkt determiniert werden kann. Term- oder Objektmetavariablen stehen für einen bestimmten Term, dessen konkrete Instanz erst zu einem späteren Zeitpunkt während des Beweisfortschrittes sinnvollerweise festgelegt wird. Mit Abschluß eines erfolgreichen Beweises findet man dann eine Instantiierung für alle verwendeten Metavariablen. Man erhält damit einen reinen Beweis in der Objektlogik und handelt während des Beweisvorganges nur hilfsweise in LLP*, um einen erfolgreichen Beweis in LLP herbeizuführen.

4.1.2 Die Semantik

Wir definieren zunächst, was eine Σ-Struktur für die sortierte Sprache LLP ist.

[4] Dadurch werden die Basisaktionen einer Anwendungsdomäne benannt.
[5] Wir unterscheiden später zwischen den Typen *Plan-* und *Nichtplan-Formel*.

Definition 4.1.1 *Gegeben eine Signatur Σ, dann ist eine Σ-Struktur ein Paar (D, I), wobei $D = (D_s)_{s \in S}$* Grundbereich *genannt wird und $I = (I(g))_{g \in \Sigma}$ eine Familie von Abbildungen, definiert über Σ, darstellt. Funktions- und Prädikatssymbolen aus Σ werden dadurch Funktionen und Prädikate über D zugewiesen.*

Gegeben eine Σ-Struktur $\mathcal{S} = (D, I)$, $f \in \Sigma$, dann schreiben wir für die Abbildung $I(f)$ oft auch $f^{\mathcal{S}}$.
Globale Variablen werden mit Hilfe der sortenerhaltenden Belegung $\beta : VG \to D$ abgebildet auf Elemente des Grundbereiches. Ist β eine Belegung, $a \in D_s$ und $x \in VG_s$ dann ist $\beta\frac{a}{x}$ diejenige Belegung, die x auf a abbildet und für alle von x verschiedenen Variablen mit β übereinstimmt.
In Zukunft wird, wenn es nicht zu Mißverständnissen führen kann, zur Vereinfachung der Notation nicht zwischen verschiedenen Sorten unterschieden.

Um den Begriff *Intervall* zu definieren, betrachten wir zunächst eine nichtleere unendliche Menge von Zuständen $\{\sigma_0, \ldots, \sigma_n, \ldots\}$. Jeder Zustand σ_i ist ein Paar $\sigma_i = (\sigma_i^1, \sigma_i^2)$.
$\sigma_i^1 : VL \to D$ ist eine Belegung, die jeder lokalen Variable ein entsprechendes Element aus D zuweist. $\sigma_i^2 \in D_{action}$ ist die sogenannte *Kontrollkomponente*, die genau eine Aktion bezeichnet, die in Zustand σ_i ausgeführt wird.
Ein *Intervall* σ ist nun eine nichtleere Sequenz von Zuständen: $\langle \sigma_0 \sigma_1 \ldots \rangle$. Die Länge eines endlichen Intervalles $\sigma = \langle \sigma_0 \sigma_1 \ldots \sigma_n \rangle$ ist n ($|\sigma| = n$). Die Länge eines unendlichen Intervalles wird mit ∞ benannt. W bezeichne fortan eine nichtleere unendliche Menge von Intervallen.
Die *unmittelbare Erreichbarkeitsrelation* über Intervallen wird definiert als die Teilintervallbeziehung R mit

$$\sigma R \sigma' \quad \text{gdw } \sigma = \langle \sigma_0 \sigma_1 \ldots \rangle \text{ und } \sigma' = \langle \sigma_1 \ldots \rangle$$

das heißt, daß σ' das terminale Teilintervall von σ ist, das sich durch Abspalten von σ_0 ergibt. R' und R^* bezeichnen den transitiven bzw. reflexiv-transitiven Abschluß von R.
Die *Komposition* von Intervallen wird als partielle Funktion über der Menge von Intervallen definiert:

$$\sigma \circ \sigma' = \begin{cases} \sigma, \text{ wenn } \sigma \text{ unendlich ist} \\ \langle \sigma_0 \ldots \sigma_n \ldots \rangle, \text{ wenn } \sigma = \langle \sigma_0 \ldots \sigma_n \rangle \text{ und } \sigma' = \langle \sigma_n \ldots \rangle \end{cases}$$

Um den Begriff einer Interpretation für die Temporallogik LLP präzise fassen zu können, definieren wir zunächst den Begriff einer *Kripke*-Struktur.

Definition 4.1.2 *Eine Kripke-Struktur $\mathcal{K}$ ist ein Tupel $\mathcal{K} = (\mathcal{S}, W, \sigma, R, \circ)$ mit $\mathcal{S}$ eine Σ-Struktur, W eine Menge von Intervallen, σ ein aktuelles Intervall aus W, R die Erreichbarkeitsrelation über W und $\circ$ die Kompositionsfunktion über W.*[6] *Wir schreiben im folgenden die Kripke-Struktur $\mathcal{K}$ abkürzend stets als Paar $(\mathcal{S}, \sigma)$.*

Definition 4.1.3 *Eine $\mathcal{K}$-Interpretation ist ein Paar $(\mathcal{K}, \beta)$ mit $\mathcal{K} = (\mathcal{S}, \sigma)$ eine Kripke-Struktur und β eine Belegung globaler Variablen in $\mathcal{S}$.*

[6] Allgemein spricht man von W als der Menge der *Welten* und von σ als einer ausgezeichneten oder aktuellen Welt.

Gegeben eine $\mathcal{K}$-*Interpretation* $\mathcal{I} = ((\mathcal{S}, \sigma), \beta)$, dann ist der Wert eines Termes $t \in \mathcal{T}_\Sigma$ in einem Intervall $\sigma \in W$ definiert gemäß:

- $\mathcal{I}(x) := \beta(x)$ für jedes $x \in VG$;
- $\mathcal{I}(a) := \sigma_0^1(a)$ für jedes $a \in VL$;

Die Interpretation wird standardmäßig auf funktionale Ausdrücke erweitert:

- für $f \in \Sigma_{ws}^F$ eine n-argumentige Funktion, $t_1, \ldots, t_n$ Terme von entsprechender Sorte aus $\mathcal{T}(\Sigma)_s$ sei $\mathcal{I}(f(t_1, \ldots, t_n)) := f^{\mathcal{S}}(\mathcal{I}(t_1), \ldots, \mathcal{I}(t_n))$.

Man kann nun die *Modell*beziehung präzisieren, die beschreibt, wann ein Ausdruck bei einer Interpretation in eine wahre Aussage übergeht. Wir definieren die Beziehung $\mathcal{I}$ *ist Modell einer Formel* ϕ und sagen dafür auch $\mathcal{I}$ *erfüllt* ϕ bzw. schreiben $\mathcal{I} \models \phi$.

Eine $\mathcal{K}$-Interpretation $\mathcal{I} = ((\mathcal{S}, \sigma), \beta)$ erfüllt eine Formel aus $\mathcal{F}_\Sigma$ gemäß:

- $\mathcal{I} \models \mathrm{T}$
- $\mathcal{I} \models ex(t)$ gdw $\mathcal{I}(t) = \sigma_0^2$
- $\mathcal{I} \models p(t_1, \ldots, t_n)$ gdw $p^{\mathcal{S}}(\mathcal{I}(t_1), \ldots, \mathcal{I}(t_n)) = T$
- $\mathcal{I} \models \neg\phi$ gdw nicht $\mathcal{I} \models \phi$
- $\mathcal{I} \models [\phi \wedge \psi]$ gdw $\mathcal{I} \models \phi$ und $\mathcal{I} \models \psi$
- $\mathcal{I} \models \forall x\ \phi$ gdw für alle $a \in D_s$ gilt $\mathcal{I}' \models \phi$ mit $\mathcal{I}' = ((\mathcal{S}, \sigma), \beta\frac{a}{x})$
- $\mathcal{I} \models \bigcirc\phi$ gdw $\mathcal{I}' \models \phi$ mit $\mathcal{I}' = ((\mathcal{S}, \sigma'), \beta)$ und es gilt $\sigma R \sigma'$ oder $|\sigma| = 0$
- $\mathcal{I} \models \Box\phi$ gdw für alle σ' mit $\sigma R^* \sigma'$ gilt $\mathcal{I}' \models \phi$ mit $\mathcal{I}' = ((\mathcal{S}, \sigma'), \beta)$
- $\mathcal{I} \models \phi\,;\psi$ gdw es gibt $\sigma', \sigma'' \in W$, wobei $\sigma = \sigma' \circ \sigma''$, σ' endlich und $\mathcal{I}' \models \phi$ mit $\mathcal{I}' = ((\mathcal{S}, \sigma'), \beta)$ und $\mathcal{I}'' \models \psi$ mit $\mathcal{I}'' = ((\mathcal{S}, \sigma''), \beta)$

Mit Hilfe der Modellbeziehung kann nun die *Folgerungs*beziehung definiert werden:

Definition 4.1.4 *Eine Formel $\phi \in \mathcal{F}_\Sigma$* folgt *aus einer Formel $\psi \in \mathcal{F}_\Sigma$ (wir schreiben auch $\psi \models \phi$) genau dann, wenn jede Interpretation, die Modell von ψ ist, auch Modell von ϕ ist.*

Bemerkungen: Die Verwendung einer Kontrollkomponente als Teil einer Zustandsbeschreibung hat das Ziel, die zu betrachtenden Modelle einzuschränken. Sie entspricht einer *Markierung* der Erreichbarkeitsrelation und koppelt die direkte Erreichbarkeit $\sigma R \sigma'$ mit der Ausführung einer einzigen Aktion in σ_0. Man betrachtet jetzt also nur noch Modelle mit speziell markierten Erreichbarkeitsrelationen über den relevanten Intervallen.

Der Unterschied zwischen den Operatoren $\circ$ (next) und $\odot$ (strong next) besteht nun darin, daß eine Formel $\odot\phi$ nur erfüllt werden kann wenn das Intervall, bzgl. dessen die Interpretation stattfindet, mindestens die Länge 1 hat. $\circ\phi$ hingegen kann auch von Intervallen der Länge 0 erfüllt werden.

4.1.3 Der Kalkül

Der Kalkül, der für LLP benutzt wird, basiert auf einem vollständigen Sequenzenkalkül für die Modallogik S4 (vgl. [Wallen 89]). Ergänzungen bzgl. temporallogischer Inferenzen entsprechen den Axiomatisierungen in [Manna & Pnueli 81] und [Rosner & Pnueli 86].
In einem kurzen Einschub folgt nun eine Erläuterung der beweistechnischen Vorgehensweise und der Begriffsfestlegung bei der Verwendung eines Sequenzenkalküls.

4.1.3.1 Sequenzenkalküle – Das Prinzip

Sequenzenkalküle sind Kalküle des natürlichen Schließens, die als organisatorische Variante des *Gentzen-Kalküls* (vgl. [Gentzen 35]) gelten. Eine *Sequenz* besteht aus zwei Listen von Formeln, dem *Antezedens* $A_1, \ldots, A_n$ und dem *Sukzedens* $S_1, \ldots, S_m$. Diese beiden Listen werden mit Hilfe des Zeichens $\Rightarrow$ verbunden und man erhält als Darstellung einer Sequenz die Form:

$$A_1, \ldots, A_n \Rightarrow S_1, \ldots, S_m$$

Die Bedeutung einer Sequenz ist die folgende:

> Unter der Annahme der Konjunktion der Antezedensformeln kann die Disjunktion der Sukzedensformeln abgeleitet werden.

Eine Sequenz enthält die vollständige Information über den aktuellen Zustand einer Ableitung, sie gibt eine *Beweissituation* wieder. Ein Beweis besteht aus einer Schrittfolge von Sequenzen; die Schritte führen dabei von bereits erzielten Beweissituationen zu einer neuen Beweissituation. Die Regeln des Schließens stellt man durch Regeln eines Kalküls (*Inferenzregeln*) dar, der mit Sequenzen operiert.
Der Beweis einer Formel G beginnt mit *logischen Axiomen*, d.h. Sequenzen der Form

$$\Theta, A, \Lambda \Rightarrow \Gamma, A, \Delta$$

(Antezedens und Sukzedens enthalten eine gemeinsame Formel A)
und endet mit einer Sequenz

$$\Rightarrow G$$

wobei G die ursprünglich zu beweisende Formel bezeichnet.

Als Beispiel einer Regel betrachten wir die $\wedge : rechts$-Regel:

$$\frac{\Theta \Rightarrow \Gamma, A, \Delta \qquad \Theta \Rightarrow \Gamma, B, \Delta}{\Theta \Rightarrow \Gamma, A \wedge B, \Delta} \; (\wedge : rechts)$$

Die Regel beschreibt, daß, um die Ableitbarkeit der Konjunktion zweier Formeln zu zeigen, gezeigt werden muß, daß jedes einzelne Konjunkt ableitbar ist, oder, um der Schlußweise der Regel zu folgen, aus der Ableitbarkeit der beiden oberen Sequenzen schließt man auf die Ableitbarkeit der unteren Sequenz.

Jede Regel besteht aus einer oder zwei oberen Sequenzen, den *Prämissen*, und aus einer unteren Sequenz, der *Konklusion*. Sie besagt, daß aus der Ableitbarkeit der Prämissen auf die Ableitbarkeit der Konklusion geschlossen werden kann.

Zusammenhängend angewandte Inferenzregeln können als Baum dargestellt werden. Ein *Beweisbaum* ist dann ein Baum, dessen Blätter logische Axiome sind und dessen Wurzel die zu beweisende Formel G in der Sequenzendarstellung $\Rightarrow G$ enthält. Die Anwendung von Inferenzregeln beschreibt dann den Übergang von den Blättern zur Wurzel, d.h. bestimmt die Struktur des Baumes. Typischerweise führt man einen Sequenzenbeweis aber rückwärts, d.h. man wendet die Regeln rückwärts an beginnend mit der Sequenz $\Rightarrow G$.

Der Kalkül von Wallen wurde für den Umgang mit LLP um eine Reihe von Regeln erweitert. Dies sind etwa:

- Die allgemeine $(\bigcirc - rechts)$-Regel:

  $$\frac{\Gamma^* \Rightarrow \phi, \Delta^*}{\Gamma \Rightarrow \bigcirc\phi, \Delta}(\bigcirc{-}rechts) \quad \text{mit} \quad \begin{array}{l}\Gamma^* = \{\psi \mid \bigcirc\psi \in \Gamma\} \cup \{\Box\psi \mid \Box\psi \in \Gamma\}, \text{ und} \\ \Delta^* = \{\psi \mid \bigcirc\psi \in \Delta\} \cup \{\Diamond\psi \mid \Diamond\psi \in \Delta\}\end{array}$$

  Die Intention dieser Regel ist, daß wenn man in der Gegenwart die Gültigkeit einer Aussage ϕ über den nächsten Zustand zeigen möchte, dann versetze man sich in diesen Zustand und zeige dort unter den dort geltenden (reduzierten) Voraussetzungen die Aussage ϕ.

- Mit der $(;-Komposition)$-Regel kann man zwei unabhängig voneinander gefundene gültige Aussagen zu einer gültigen Aussage zusammensetzen:

  $$\frac{\phi_1 \Rightarrow \psi_1 \qquad \phi_2 \Rightarrow \psi_2}{\phi_1;\phi_2 \Rightarrow \psi_1;\psi_2}\,(;-Komposition)$$

Zum Verständnis des Vorgehens werden unmittelbar wichtige Regeln in diesem Kapitel bedarfsweise an der Stelle eingeführt, an der sie für zu führende Beweise gebraucht werden. Anhang B enthält dann die restlichen verwendeten Regeln.
Für das Führen von Beweisen in der Logik LLP* muß es auch Regeln geben, die mit Metavariablen umgehen können. Es gilt dabei die folgende offensichtlich notwendige Einschränkung:

Festlegung: Zu jeder Metavariablen $meta_k \in \text{LLP}^* \setminus \text{LLP}$, die in einem Beweis verwendet wird, gibt es *höchstens zwei* assoziierte Einführungsregeln, nämlich eine *links*-Einführungs- und eine *rechts*-Einführungsregel. Beide Regeln abstrahieren dabei die selbe Formel aus LLP* auf die Metavariable $meta_k$ (analog für Termmetavariablen, vgl. auch Seite 131).

Das Schema für eine *links*-Einführung sieht wie folgt aus:

$$\frac{\Gamma,\ \phi,\ \Delta \Rightarrow \Theta}{\Gamma,\ meta_k,\ \Delta \Rightarrow \Theta}$$

Es gilt: $meta_k \in \text{LLP}^* \setminus \text{LLP}$ und $meta_k$ kommt nicht in $\phi \in \text{LLP}^*$ vor.

Die *rechts*-Einführung einer Metavariable ist dazu analog. Zur Vereinfachung der Beweisführung ist in den Regeln auch eine Einführung innerhalb einer Formel erlaubt, d.h. in der Regelkonklusion taucht die Metavariable als Teilformel einer anderen Formel auf.

Logische Axiome erhalten für den Umgang mit der Logik LLP* eine zusätzliche Einschränkung gemäß:

$\Theta, A, \Lambda \Rightarrow \Gamma, A, \Delta$ Es gilt: $A \in$ LLP und Θ, Λ enthalten nur Ausdrücke aus LLP.

Für die Führung eines Beweises in LLP* wird gefordert, daß zu allen in dem Beweis verwendeten Metavariablen entsprechende Einführungsregeln gemäß den oben beschriebenen Bedingungen während des Beweises eingesetzt werden. Wenn in Zukunft von einem *LLP*-Beweis* geredet wird, dann impliziert dies, daß der Beweis gemäß dieser Forderung geführt wurde. Folgt man dieser Forderung, so ist leicht einzusehen, daß ein erfolgreicher Beweis einem Beweis rein in LLP entspricht.

4.2 Flexible Planrepräsentation und -generierung

In Abschnitt 3.2 wurde definiert, wie ein Planungsproblem charakterisiert ist und daß Planen die Realisierung eines Planungsalgorithmus im Sinne einer Suche im Raum der Planungszustände kennzeichnet. Diese allgemeine Beschreibung wird nun auf einen deduktiven Planungsansatz projiziert. Als Instantiierung der logischen Planungssprache $\mathcal{L}$ wird im folgenden LLP angenommen. Damit wird ein initialer Planungszustand (vgl. Definition 3.2.2)

$$\langle V, MvP, Z, MvPSubst, MvSubst, Th \rangle$$

auf eine *Planspezifikationsformel* $V \wedge MvP \rightarrow Z$ aus LLP* abgebildet. Übergänge zwischen Planungszuständen werden durch *Regelanwendungen* im Sequenzenkalkül für LLP* wiedergegeben. Spezielle Regeln sind dabei für die Berücksichtigung von Aktionen notwendig. Einem Planungsalgorithmus entspricht eine bestimmte *Beweisstrategie*, die in der Lage ist, einen nichttrivialen Beweis der Planspezifikation zu führen.
Die nun folgenden Abschnitte beschreiben den deduktiven Planungsansatz im Detail.

4.2.1 Charakterisierung von Aktionen

Planen wird in unserer Betrachtung im Sinne von Handlungsplanung verstanden und somit sind die elementaren Vorgänge, die bei der Beschreibung der *Theorie* einer Planungsdomäne zu berücksichtigen sind, Handlungen bzw. *Aktionen*[7] eines *Agenten*. Diese Theorie dient als Grundlage, um einerseits zu definieren, was Pläne sind, und andererseits dem Konzept einer Planspezifikation eine Interpretationsgrundlage zu verschaffen.
Aktionen werden im Sinne von zustandsbasierten Handlungen in der Planungsdomäne verstanden, deren Ausführung bestimmte, exakt identifizierbare Veränderungen eines Zustandes hervorrufen.
Eine Aktion zeigt folgende charakteristische Eigenschaften, wobei damit einhergehende Einschränkungen nicht notwendigerweise von prinzipieller Natur sind:

- sie ruft nur diskrete Veränderungen hervor und bildet einen atomaren Zustandsübergang ab, d.h. es gibt einen *Beginn* und ein *Ende*, aber es gibt kein *Dazwischen*;

[7]Dieser Begriff wird im folgenden präferiert.

  - der logische Formalismus läßt auch zeitlich komplexere Aktionenschemata zu, deren Verwendung dann allerdings auch wesentliche Veränderungen bei der Schemainstantiierung und -verwendung hervorruft;

- sie hat deterministische Effekte, d.h. die Effekte entstehen durch Anwendung einer deterministischen Funktion auf die Aktion und den Zustand bei Aktionsbeginn;

  - nichtdeterministische Effekte können nur auf Basis von prädikatenlogischer Disjunktion beschrieben werden;

- ihre Effekte sind vollständig in ihrer Beschreibung erfaßt, d.h. es gibt keine unabhängig von der Aktion beschriebenen Gesetzmäßigkeiten aus denen zusätzliche Effekte ableitbar sind;

  - die Integration von Nicht-Aktionsaxiomen zur Beschreibung von Domänenrandbedingungen erfordert zusätzlichen Aufwand bei der Berechnung von Anwendbarkeitsbedingungen für Aktionen und von Rahmenaxiomen (vgl. auch Abschnitt 4.2.5).

Wir stellen folgende zusätzliche Forderung an das Modell einer realen Domäne:

- die einzige Möglichkeit, Veränderungen in dem Modell hervorzurufen, besteht darin, Aktionen auszuführen, d.h. es gibt keine exogenen Ereignisse;

  - man kann sich die Repräsentation betreffend jedoch z.B. dadurch behelfen, daß man eine Zweiteilung der Menge von Aktionsaxiomen vornimmt, wobei eine Menge Aktionen eines Agenten beschreibt und die andere Ereignisse axiomatisiert[8]; für die Erstellung eines Planes verwendet man explizit nur die erste Menge, wobei Axiome der zweiten Menge aber als Steuerungsinformation verfügbar sind.

Um besser zu verstehen, wie Zustandsänderungen ausgeführt werden können, ist es wichtig, zwei wesentliche Aspekte der Semantik von LLP (siehe Abschnitt 4.1.2) noch einmal herauszustellen:

- Neben globalen Variablen, d.h. den "normalen" logischen Variablen, gibt es auch sogenannte lokale Variablen, die ihren Wert bei Zustandsübergängen ändern können.

- Ein Zustand ist charakterisiert durch die aktuelle Belegung der lokalen Variablen, d.h. effektive Zustandsänderungen sind durch die Änderung der Belegungen von lokalen Variablen[9] erreichbar.[10]

[8] In einem Plangenerierungs-, Planausführungsszenario kann diese Menge unter Einhaltung bestimmter Randbedingungen dynamisch sein.

[9] Sie sind neben dem Prädikat *ex* die einzigsten Symbole in LLP, deren Interpretation sich verändern kann.

[10] Der Aspekt, daß ein Zustand zusätzlich auch durch die ausgeführte Aktion charakterisiert wird, ist hier jetzt nicht entscheidend.

Die Entitäten einer Anwendungsdomäne, die modelliert werden, damit planbasiertes Schließen in dieser Domäne überhaupt erst möglich wird, werden ganz allgemein als *Objekte* mit spezifischen Eigenschaften angesehen. Dies sind z.B. *Personen* mit Eigenschaften wie *Größe, Alter, Augenfarbe, Angestellt-bei,* usw., oder so abstrakte Dinge wie *Dateien* mit Eigenschaften wie *Name, Typ, Zugriffsrecht.* Aktionen oder Handlungen in einer durch Objekte beschreibbaren Welt verändern nun durch ihre Ausführung Eigenschaften bestimmter Objekte.
Wenn nun also veränderbare Objekte einer Anwendungsdomäne modelliert werden sollen, bleibt nur die Möglichkeit, sie mit lokalen Variablen zu assoziieren.
Da Objekte im allgemeinen strukturiert sind, modelliert man diese Struktur durch die Verwendung von entsprechenden Selektorfunktionen. Abbildung 4.1 verdeutlicht den Zusammenhang.

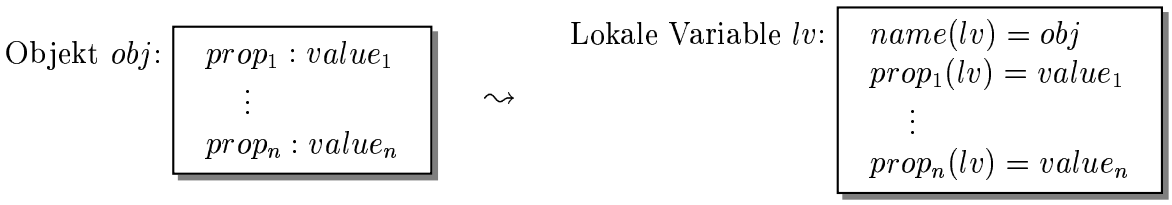


Abbildung 4.1: Objektstruktur vs. lokale Variable

Die Struktur eines Objektes ist durch eine Menge entsprechender Selektorfunktionen bzgl. der lokalen Variablen dargestellt, d.h. einer Eigenschaft $prop_i$ entspricht eine Selektorfunktion $prop_i$ der lokalen Variable. Zur eindeutigen Identifikation eines Objektes dient die Funktion *name*, die auf einen unikalen konstanten Bezeichner abbildet.
Die Strukturierung eines Objektes kann einer beliebig tiefen Hierarchie entsprechen (siehe Abbildung 4.2) und hinter Eigenschaften können sich auch Assoziationslisten von strukturierten Objekten befinden (siehe Abbildung 4.3); die Schlüssel der Assoziationslisten sind nicht weiter strukturierbar[11].

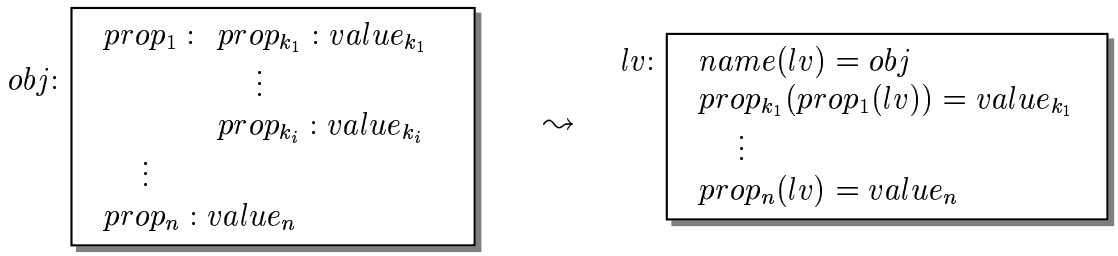


Abbildung 4.2: Hierarchische Objektstruktur

Einen Zugriff $prop_{k_1}(assoc\_position(key_1, prop_1(lv)))$ auf ein Element einer Assoziationsliste schreiben wir in Zukunft abkürzend als $prop_{k_1}(prop_1(key_1, lv))$. Die Bezeichner key_i

[11] Zur Repräsentation einer einfachen Liste wählt man die Schlüssel als mit 1 beginnende aufsteigende natürliche Zahlen; sie geben dann die Position eines Elementes in der Liste an.

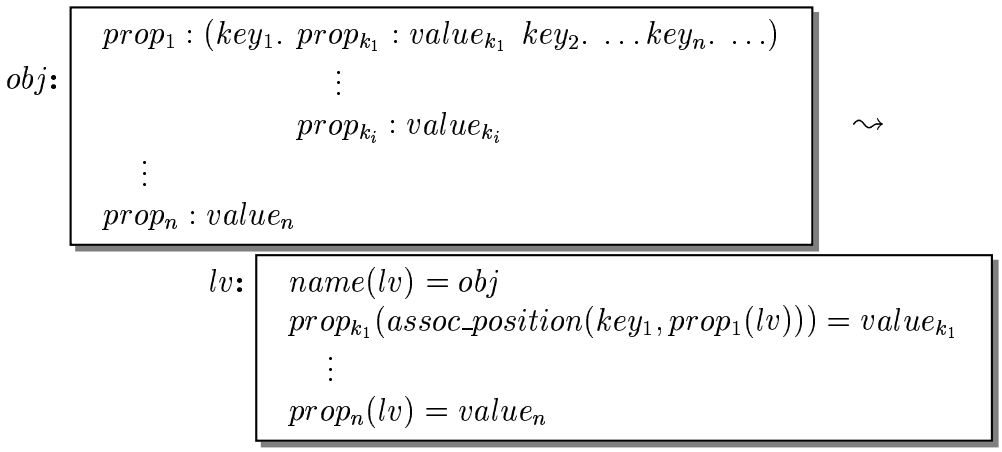


Abbildung 4.3: Assoziationslisten als Objekteigenschaften

und $value_i$ stehen für funktionsfreie[12] Terme mit konstanter Interpretation. Eine Objekteigenschaft beschreiben wir abstrakt als eine Gleichung $t(\vec{x}', lv) = x^0$ bzw. als $p(\vec{x}, lv)$. Der Vektor $\vec{x}$ besteht aus dem Symbol x^0 und den Symbolen des Vektors $\vec{x}'$. Sie stehen für Konstanten bzw. logische Variablen.[13] lv bezeichnet eine lokale Variable.
In seiner Gesamtheit wird ein strukturiertes Objekt nun also modelliert durch eine Menge von LLP-Gleichungen.

Definition 4.2.1 *Die Menge der unterschiedlichen Gleichungen zur Repräsentation von Objekteigenschaften, die durch die Modellierung der unterschiedlichen Objektarten einer Planungsdomäne $\mathcal{D}$ entstehen, werden in der Menge $\underline{Basiseigenschaften_{\mathcal{D}}}$ zusammengefaßt. Diese Menge umfaßt auch alle möglichen Gleichungen zwischen funktionsfreien Termen.*

Das Verhalten einer Aktion kann man nun beschreiben, indem man jeweils in Form von Gleichungen Eigenschaften betroffener Objekte vor Ausführung und nach Ausführung der Aktion angibt. Man beschreibt damit implizit die von der Aktion bewirkten *minimalen Veränderungen* von Eigenschaften. Beschreibt man Aktionen in einem rein logischen Formalismus, so heißt das, ihr Verhalten in Form von Axiomen zu definieren. Die folgende Definition beschreibt die verwendete Form von Aktionsaxiomen.

Definition 4.2.2 *Aktionsaxiome in LLP haben folgendes Aussehen:*

$$\forall \vec{x}\ pre(\vec{x}, \vec{v}) \wedge ex(a(\vec{x})) \rightarrow \bigcirc eff(\vec{x}, \vec{v}) \tag{4.1}$$

oder

$$\forall \vec{x}\ ex(a(\vec{x})) \rightarrow [\bigwedge_{i=1}^{n} [pre_i(\vec{x}, \vec{v}) \rightarrow \bigcirc eff_i(\vec{x}, \vec{v})]] \tag{4.2}$$

[12] Zu interpretieren im Sinne von: Funktionen mit Stelligkeit größer als 0 sind nicht erlaubt.

[13] Zur Vereinfachung werden wir später oft eine Quantifizierung über eine mit einem Vektor $\vec{x}$ parametrisierte Formel schreiben als $\forall \vec{x}\ \phi(\vec{x})$, auch wenn in $\vec{x}$ Konstanten enthalten sind; die Quantifizierung einer Konstante sei dann entsprechend eliminiert.

$\vec{v}$ steht für einen Vektor von lokalen Variablen; die Dimension von $\vec{v}$ ist kleiner oder gleich der Dimension von $\vec{x}$. $pre(\vec{x},\vec{v})$ ($pre_i(\vec{x},\vec{v})$) steht für eine modalfreie LLP-Formel, wobei alle auftretenden Literale (negierten) atomaren Formeln aus der Menge Basiseigenschaften$_{\mathcal{D}}$ *entsprechen; $eff(\vec{x},\vec{v})$ ($eff_i(\vec{x},\vec{v})$) steht für eine Konjunktion von atomaren Formeln aus* Basiseigenschaften$_{\mathcal{D}}$*; ex ist ein ausgezeichnetes Prädikat zur Aufnahme eines Aktionstermes (vgl. Abschnitt 4.1).*

Das Axiom 4.1 besagt, daß das Gelten von Vorbedingungen $pre(\vec{x},\vec{v})$ und das Gelten des Prädikates $ex(a(\vec{x}))$ – die Ausführung der Aktion $a(\vec{x})$ – implizieren, daß im unmittelbar nächsten Zustand die Effekte $eff(\vec{x},\vec{v})$ gelten.
Das Axiom 4.2 beschreibt eine Variante der Aktion $a(\vec{x})$ mit konditionalen Effekten, d.h. zusätzlich zu dem "normalen" Effekt entsteht unter der Bedingung, daß $pre_i(\vec{x},\vec{v})$ gilt, noch der Effekt $eff_i(\vec{x},\vec{v})$. Dieses Axiom ist wie man leicht sieht äquivalent zu der folgenden Menge von Axiomen:

$$\begin{aligned}
&\forall \vec{x}\ pre_1(\vec{x},\vec{v}) \wedge ex(a(\vec{x})) \rightarrow \bigcirc eff_1(\vec{x},\vec{v})\\
&\vdots\\
&\forall \vec{x}\ pre_n(\vec{x},\vec{v}) \wedge ex(a(\vec{x})) \rightarrow \bigcirc eff_n(\vec{x},\vec{v})
\end{aligned}$$

Für eine Formel $pre_i(\vec{x},\vec{v})$ gilt folgende notwendige Voraussetzung:

- zu jeder Variablen $v_i \in \vec{v}$ gibt es eine Gleichung $name(v_i) = x_i$ in $pre(\vec{x},\vec{v})$ mit $x_i \in \vec{x}$.

Damit sichert man, daß man mit allen lokalen Variablen eindeutig ein bestimmtes Objekt identifiziert. In den Argumenten der Aktion verwendet man eine globale Variable, um die relevanten Objekte anzusprechen.
An die Formeln $pre_i(\vec{x},\vec{v})$ bzw. $eff_i(\vec{x},\vec{v})$ können verschiedene Voraussetzungen gestellt werden:

1. für $i \neq j,\ 1 \leq i,j \leq n$ gilt: $pre_i(\vec{x},\vec{v}) \wedge pre_j(\vec{x},\vec{v})$ ist inkonsistent;
2. zwei Formeln $eff_i(\vec{x},\vec{v})$ und $eff_j(\vec{x},\vec{v}),\ i \neq j,\ 1 \leq i,j \leq n$ sind jeweils nicht äquivalent;
3. die Konjunktion $\bigwedge_{i=1}^{n} eff_i(\vec{x},\vec{v})$ ist
   (a) konsistent
   (b) nicht konsistent
4. es gibt ein $eff(\vec{x},\vec{v})$ mit $eff_i(\vec{x},\vec{v}) \rightarrow eff(\vec{x},\vec{v})$ für $1 \leq i \leq n$;
5. es gibt ein $pre(\vec{x},\vec{v})$ mit $pre_i(\vec{x},\vec{v}) \rightarrow pre(\vec{x},\vec{v})$ für $1 \leq i \leq n$.

Die Auswirkungen der jeweiligen Voraussetzungen auf den Umgang mit den resultierenden Aktionsbeschreibungen werden später noch näher erläutert.

4.2.2 Zulässige Planspezifikationen

Ausgangspunkt für die Plangenerierung sind formale Planspezifikationen. Diese beschreiben die zu lösende Planungsaufgabe in Form von Formeln einer Subsprache der Logik LLP* (siehe Abschnitt 4.1), wobei eine Spezifikationsformel das zeitliche und zielgerichtete Verhalten eines Planes, der die Spezifikation erfüllt, beschreiben.
Das Grundprinzip einer Spezifikation ist als Formelschema wie folgt beschrieben:

$$\forall \vec{x}\ \mathit{Vorbedingungen}(\vec{x}) \wedge \mathit{Planplatzhalter}(\vec{x}) \rightarrow \mathit{Ziele}(\vec{x})\ ,$$

das heißt, die Voraussetzungen (*Vorbedingungen*), unter denen ein noch nicht bekannter Plan (*Planplatzhalter*) einen bestimmten Zweck (*Ziele*) erfüllt, werden festgelegt. Zum Zwecke der Verallgemeinerung beschreibt man dies als eine Aussage über variablen Objekten, dargestellt durch einen Vektor $\vec{x}$ von globalen Variablen, der die Elemente des Schemas parametrisiert.[14] *Vorbedingungen* und insbesondere *Ziele* bezeichnen im allgemeinen temporallogische Formeln.
Es folgen nun einige Beispiele für Planspezifikationen, die die mögliche Bandbreite spezifizierbarer Planungsprobleme in der Logik LLP* charakterisieren.

- Die *klassische* Problemspezifikation

  $$\forall \vec{x}\ \mathit{Vorbedingungen}(\vec{x}) \wedge \mathit{Planplatzhalter}(\vec{x}) \rightarrow \Diamond \mathit{Ziele}(\vec{x})$$

  Hier sind *Vorbedingungen* und *Ziele* jeweils modalfreie Formeln, d.h. es ist die klassische (in den meisten Planungsansätzen ausschließlich betrachtete) Situation gegeben, daß ein initialer Zustand und ein einziger in der Zukunft liegender Zielzustand beschrieben sind.

- Multiple Zielzustände

  $$\forall \vec{x}\ \mathit{Vorbedingungen}(\vec{x}) \wedge \mathit{Planplatzhalter}(\vec{x}) \rightarrow \Diamond[\mathit{ZieleA}(\vec{x}) \wedge \Diamond \mathit{ZieleB}(\vec{x})]$$

  $$\forall \vec{x}\ \mathit{Vorbedingungen}(\vec{x}) \wedge \mathit{Planplatzhalter}(\vec{x}) \rightarrow \Diamond \mathit{ZieleA}(\vec{x}) \wedge \Diamond \mathit{ZieleB}(\vec{x})$$

  Vorbedingungen, *ZieleA* und *ZieleB* sind jeweils modalfreie Formeln, d.h. im ersten Fall ist eine zeitlich totale Ordnung zwischen Zielzuständen und im zweiten Fall nur eine partielle Ordnung beschrieben.

  Neben zeitlich nicht determinierten Zielzustandsaussagen sind auch eingeschränktere Spezifikationen möglich, beispielsweise beschreibt

  $$\forall \vec{x}\ \mathit{Vorbedingungen}(\vec{x}) \wedge \mathit{Planplatzhalter}(\vec{x}) \rightarrow \Diamond[\mathit{ZieleA}(\vec{x}) \wedge \bigcirc\bigcirc\bigcirc \mathit{ZieleB}(\vec{x})]$$

  daß, nachdem ein Zustand erreicht ist, in dem *ZieleA* gilt, *drei* Zustände später *ZieleB* durch den Plan erreicht werden muß.

- Verhaltensspezifikation *aller Planzustände*

  $$\forall \vec{x}\ \mathit{BedingungenA}(\vec{x}) \wedge \Box \mathit{BedingungenB} \wedge \mathit{Planplatzhalter}(\vec{x}) \rightarrow \mathit{Ziele}(\vec{x})$$

[14] Der Vektor von Variablen darf auch leer sein.

BedingungenA und *BedingungenB* bezeichnen hier modalfreie Formeln, d.h. die Vorbedingungen des Planes sind aufgeteilt in eine initiale Zustandsbeschreibung und in eine Beschreibung, die alle vom zu findenden Plan zu erreichenden Zustände charakterisiert. Dies bedeutet, daß der Plan niemals eine Aktion verwenden kann,[15] deren Voraussetzungen oder Ziele mit *BedingungenB* unverträglich sind.

Neben zustandsbasiertem Verhalten kann auch intervallbasiertes Verhalten als Allaussage über allen Planzuständen formuliert werden:

$$\forall \vec{x}\ \mathit{BedingungenA}(\vec{x}) \wedge \Box[\mathit{BedingungB} \rightarrow \bigcirc \mathit{BedingungC}] \wedge \mathit{Planplatzhalter}(\vec{x}) \rightarrow \mathit{Ziele}(\vec{x})$$

Hier ist z.B. spezifiziert, daß, wenn immer *BedingungB* erreicht wird, im folgenden Zustand *BedingungC* gelten soll.

Mit der zusätzlichen Spezifikation von Bedingungen über das gewünschte Verhalten aller Bestandteile eines Planes erreicht man die Selektion einer bestimmten Planklasse, deren Vertreter die beschriebene Planspezifikation erfüllen, d.h. man spezifiziert auf diese Weise Kontrollmechanismen für den nachfolgenden Prozeß der Plangenerierung. Die zusätzlichen Bedingungen können als *Sicherheitsbedingungen* aufgefaßt werden.

4.2.3 Pläne in der Logik LLP

Die in Abschnitt 3.1.3 eingeführte Sichtweise auf Pläne als Beschreibungen von Äquivalenzklassen von Aktionsfolgen wird nun auf die Logik LLP projiziert. In Abschnitt 4.1 wurde bereits erwähnt, daß Pläne eine spezielle Klasse von Formeln der Logik LLP darstellen, die nur bzgl. einer Planspezifikationsformel und einer Menge von Aktionsaxiomen eine aussagekräftige Bedeutung haben.
Die vorgeschlagene Charakterisierung von Aktionsplänen stellt eine Verallgemeinerung traditionell verwendeter Plankonzepte[16] dar, die aber als Spezialfälle der vorgeschlagenen Repräsentation darstellbar sind. Wir betrachten nun zum einen die Beziehung von Plänen zu korrespondierenden *Aktionsintervallen* in der Logik LLP und zum anderen das Verhältnis von Plänen zu Planspezifikationen aufbauend auf dieser Beziehung.
In der Definition eines Modelles einer Formel (vgl. die Semantik von LLP in Abschnitt 4.1.2) bildet ein Intervall bzw. die Teilintervallbeziehung eine wesentliche Komponente. Insbesondere gibt es in jedem Zustand eines Intervalles eine Kontrollkomponente, die aus einer Abbildung auf einen Aktionsterm besteht. Im vorherigen Abschnitt wurden Aktionsaxiome eingeführt, die notwendig sind, um eine spezifische Planungsdomäne zu beschreiben. Ihre Formulierung hat eine modelleinschränkende Wirkung, d.h., legt man die Axiome zugrunde, so gibt es nun für eine Formel nur noch Modelle, in denen auch die Axiome gelten. Für eine Formel ϕ, die unter der Berücksichtigung der Aktionsaxiome ein Modell hat, kann man von dem assoziierten Intervall σ abstrahieren zu einer Folge von Aktionen, die gerade den Kontrollkomponenten der Zustände von σ entsprechen. Für $\sigma = \langle \sigma_0 \sigma_1 \ldots \rangle$

[15] Um nicht in einer trivialen Aussage zu enden.

[16] Pläne
sind lineare Folgen von Aktionen (vgl. z.B. PRODIGY [Carbonell & the PRODIGY Group 92]) oder partiell geordnete Mengen von Aktionen (vgl. z.B. SNLP [McAllester & Rosenblitt 91]).

sei $\sigma^A = \langle \sigma_0^2 \sigma_1^2 \ldots \rangle$ dann das zugehörige *Aktionsintervall*; entsprechend bezeichnen wir mit $\sigma^D = \langle \sigma_0^1 \sigma_1^1 \ldots \rangle$ das zu σ gehörende *Beschreibungsintervall*. Zusammenfassend schreiben wir σ auch in der Form

$$\sigma = \left\langle \begin{bmatrix} \sigma_0^1 \\ \sigma_0^2 \end{bmatrix}_0 \begin{bmatrix} \sigma_1^1 \\ \sigma_1^2 \end{bmatrix}_1 \ldots \right\rangle .$$

Das Aktionsintervall charakterisiert eine Klasse von Modellen, deren Unterschiede bzgl. der Erfüllbarkeit von ϕ unwesentlich sind. Beschreibt σ ein endliches Intervall $\langle \sigma_0 \sigma_1 \ldots \sigma_n \rangle$ so ergibt sich für σ^A das Aktionsintervall $\langle \sigma_0^2 \sigma_1^2 \ldots \sigma_{n-1}^2 \rangle$, d.h. die Kontrollkomponente des letzten Zustandes von σ ist irrelevant.[17]
Da ein Modell $\mathcal{I}$ im allgemeinen unendlich groß ist, kann dennoch folgendes zutreffen: $\mathcal{I}$ ist ein Modell von ϕ und zu einem Intervall $\sigma = \langle \sigma_0 \sigma_1 \ldots \rangle$ assoziiert mit $\mathcal{I}$ gibt es ein minimales initiales Teilintervall $\sigma' = \langle \sigma_0 \sigma_1 \ldots \sigma_n \rangle$, das gerade ausreicht, um ϕ zu erfüllen. Die Zustände in σ' sind dann für ϕ *relevante Zustände*. Das Aktionsintervall σ'^A heißt dann *relevantes Aktionsintervall*.
Im allgemeinen gibt es für die Formel ϕ viele Modelle, insbesondere solche deren assoziierten Aktionsintervalle bzw. relevanten Aktionsintervalle sich signifikant unterscheiden. Zwei Aktionsintervalle γ^A und δ^A unterscheiden sich *signifikant*, wenn es ein i gibt, so daß für die relevanten Zustände γ_i und δ_i gilt, daß γ_i^2 und δ_i^2 unterschiedliche Funktionen enthalten.
Die bisherige Charakterisierung von Aktionsintervallen, die auf der Modellexistenz begründet ist, reicht noch nicht aus, um von Plänen im operationalen Sinne reden zu können; der *Ausführbarkeit* eines Aktionsintervalles wurde noch keine Rechnung getragen. Ein Aktionsintervall σ^A wird als *ausführbar* bezeichnet, wenn für alle σ_i^2 aus σ^A gilt, daß die Vorbedingungen der mit σ_i^2 assoziierten Aktionsaxiomeninstanz in der Zustandskomponente σ_i^1 aus σ gelten. Nur dieser Zusatz garantiert, daß die Effekte der in σ_i^2 befindlichen Aktion tatsächlich im Modell etabliert sind und damit die Ausführung der Aktion bestätigen. Mit Hilfe des Aktionsintervallbegriffes kann man nun den Begriff eines Planes charakterisieren.

Definition 4.2.3 *Gegeben sei eine Planspezifikation PS. Ein* Plan *wird nun angesehen als eine LLP-Formel, die eine Äquivalenzklasse von bzgl. PS relevanten ausführbaren Aktionsintervallen charakterisiert. Die mit den Aktionsintervallen assoziierten Modelle erfüllen alle die Planspezifikation PS. Der Plan kann sich sowohl der Information aus den Aktionsintervallen als auch aus den assoziierten Beschreibungsintervallen bedienen.*

Mithilfe des vorgestellten Konzeptes von Plänen ist es nun möglich, eine große Bandbreite zwischen einem extrem unspezifischen Plan – die Klasse der Aktionsintervalle ist maximal groß bzgl. der vorausgesetzten Aktionsaxiome – und einem extrem spezifischen Plan – die Klasse der Aktionsintervalle ist singulär – zu beschreiben. Da ein Plan syntaktisch gesehen eine LLP-Formel ist, steht prinzipiell die gesamte Ausdrucksfähigkeit der logischen Sprache zur Beschreibung eines Planes zur Verfügung. Die Notwendigkeit der Ausführbarkeit eines Aktionsintervalles steht in unmittelbarem Zusammenhang mit der Operationalisierung des Planbegriffes. Der Prozeß des Planens hat im allgemeinen die Aufgabe, diese notwendige Eigenschaft zu beweisen. Ein Plan wirkt in dem hier verfolgten

[17] Die Aktion, die den im letzten Zustand geltenden Effekt hervorruft, wird im vorletzten Zustand ausgeführt.

Ansatz dann als *Informationsstruktur*, die charakteristische Aspekte umfaßt, die während des Planens, d.h. dem Prozeß, der die Existenz von Aktionsintervallen absichert, entstehen. Je nach Verwendungsweise des Planes kann dies sehr unterschiedliche Information sein und es muß nicht notwendigerweise Information aus den Aktionsintervallen sein, sondern die ausschließliche Verwendung von Beschreibungsintervallauszügen kann genügen, d.h. der Plan kann durchaus eine vollständig intensionale Spezifikation der intendierten Aktionsfolge(n) sein.
Die Uniformität in der Beschreibung von Planspezifikationen und Plänen kommt einer solchen Flexibilität der Darstellung geradezu entgegen; weiterhin löst sich damit das Problem der Garantie einer wohldefinierten Semantik eines Planes von selbst.

Relevante Modelle enthalten aus Planverwendungssicht auf verschiedene Weise interessante Information:

- das Vorkommen von Aktionen inklusive einer (partiellen) Ordnung, die ihr Auftreten einschränken;
- temporale Strukturen, die die Intervalldichte bzgl. der explizit genannten Aktionen garantieren bzw. diese nicht fordern;
- Charakterisierung einer Teilklasse von Aktionsintervallen durch die implizite Spezifikation auftretender Aktionen:
  - indem Bedingungen spezifiziert werden, die in bestimmten Zuständen aller Intervalle gelten müssen;[18]
  - durch den expliziten Ausschluß bestimmter Aktionen;[19]
- Fokussierung von Modellen hinsichtlich der induzierten Intervallänge: Angabe von exakter, minimaler und/oder maximaler Länge;
- Einschränkungen von Aktionsparameterwerten.

Der Plan als objektsprachliche Formel betrachtet ist, um dies noch einmal zu betonen, nur eine je nach Verwendung des Planes mehr oder weniger detaillierte Charakterisierung der die Planspezifikation erfüllenden Aktionsintervalle.

Traditionelle wohldefinierte Planungsansätze [Penberthy & Weld 92, Fink & Veloso 95] hingegen verfolgen eine strikte Trennung zwischen der Planstruktur und der zugrundeliegenden Semantik des Planungsprozesses. Dies ist durch ihr algorithmisches Vorgehen begründet. Ihre Planungsalgorithmen arbeiten nicht direkt auf der zugrundeliegenden semantischen Ebene einer Planungslogik, sondern die Korrektheit ihres Vorgehens muß explizit unter Zuhilfenahme spezieller Abbildungen bewiesen werden; dies gilt auch für die Interpretation der erstellten Pläne. Eine Veränderung der Planstruktur hätte somit zur Folge, eine veränderte Interpretation aufzustellen und zu beweisen; ebenso müßte

[18] Indem sie bereits gegolten haben und nicht zerstört werden dürfen, oder indem sie noch erreicht werden müssen.

[19] Und damit dem erlaubten Auftreten anderer Aktionen.

der Planungsalgorithmus verändert werden und seine Korrektheit von neuem bewiesen werden.

Betrachten wir nun Beispiele für Pläne und beginnen mit der einfachsten Art eines Planes, einem Plan, der nur aus einer Aktion besteht. Die Formel $ex(aktion) \wedge \odot\bigcirc F$ stellt abstrakt solch einen Plan dar. Sie beschreibt als Formel ein Intervall, das genau einen Zustandsübergang lang ist und dessen erster Zustand die Aktion *aktion* verlangt. Der Effekt der Aktion tritt dann im zweiten (und letzten) Zustand des Intervalls ein (vgl. die Aktionsaxiomenstruktur in Abschnitt 4.2.1). Als zusätzliche Randbedingungen sind hier lediglich zeitliche Eigenschaften des Intervalls festgelegt, nämlich, daß das Intervall genau die Länge 1 haben muß, dargestellt durch die modallogische Formel $\odot\bigcirc F$[20]. Durch die sequentielle Komposition[21] solcher Basispläne erhält man einfache *lineare* Pläne. Ein total geordneter Plan dieser Art beschreibt dann ein Intervall, das genau so viele Zustandsübergänge enthält wie Aktionen in dem Plan benannt sind, d.h. das Intervall ist dicht bzgl. der explizit spezifizierten Aktionen. Der Plan beschreibt genau ein relevantes Aktionsintervall.

Zum besseren Verständnis der Beziehung von Plänen zu ihren Planspezifikationen betrachten wir exemplarisch ein Aktionsintervall für einen linearen Plan gemäß der LLP-Semantik (vgl. Abschnitt 4.1.2) im Detail.
Sei folgende abstrakte Planspezifikation gegeben:

$$pre \wedge \mathsf{P} \rightarrow \Diamond[goal_1 \wedge \Diamond goal_2]$$

pre, $goal_1$ und $goal_2$ seien modalfreie Formeln. Eine zugehörige Instantiierung für P sei:

$$\mathsf{P}_{inst} \equiv ex(a_1) \wedge \odot\bigcirc F; ex(a_2) \wedge \odot\bigcirc F; \ldots; ex(a_n) \wedge \odot\bigcirc F \ ,$$

wobei die a_i für Grundterme stehen sollen. P_{inst} beschreibt dann ein singuläres Aktionsintervall. Ein Modell für die instantiierte Planspezifikationsformel ist wie folgt charakterisierbar:

$$\left\langle \underset{a_1}{\boxed{pre \wedge pre_{a_1}}}_0 \ \underset{a_2}{\boxed{pre_{a_2} \wedge eff_{a_1}}}_1 \ \ldots \ \underset{a_i}{\boxed{pre_{a_i} \wedge goal_1}}_{i-1} \ \ldots \ \underset{a_n}{\boxed{pre_{a_n} \wedge eff_{a_{n-1}}}}_{n-1} \ \boxed{goal_2}_n \right\rangle .$$

Hier sind für den Beschreibungsintervallanteil signifikante Formeln hervorgehoben, die im jeweiligen Zustand gelten; der Aktionsintervallanteil ist direkt durch die Aktionsterme beschrieben anstelle der Funktionen, die auf sie abbilden. In Zustand 1 z.B. gelten demnach die Vorbedingungen für Aktion a_2 und die Effekte von Aktion a_1; Aktion a_n sorgt dafür, daß das spezifizierte Ziel $goal_2$ in Zustand n gilt.
Das Intervall besteht aus den Zuständen $\sigma_0, \ldots, \sigma_n$, hat also die Länge n. Die Planspezifikation beschreibt Vorbedingungen pre für den Plan, die in Zustand σ_0 gelten müssen. Weiterhin sind zwei Ziele spezifiziert, wobei das zweite Ziel, $goal_2$, nicht vor dem ersten Ziel erreicht werden darf. Der Plan wird gerade so gewählt, daß das zweite Ziel mit der

[20] Der Modaloperator $\odot$ ist eine Abkürzung für die Operatorfolge $\neg\bigcirc\neg$.
[21] Mit Hilfe des Modaloperators " ; ".

letzten Aktion im Plan gerade vollständig erreicht ist, es handelt sich also um einen notwendigen Plan.[22]
In der Modellcharakterisierung ist ebenfalls gut erkennbar, wie die Vorbedingungen und Effekte aufeinanderfolgender Aktionen in Beziehung stehen. Im letzten Zustand, σ_n, ist der Wert der Zustandskontrollkomponente, d.h. das entsprechende ex-Prädikat, nicht von Bedeutung. Die Formel $\odot\bigcirc F$, verbunden mit jeder Basisaktion im Plan, bewirkt die Dichtheit des Intervalles bzgl. der Planformel. Die Intervallstruktur läßt dann nicht zu, daß zwischen zwei Aktionen a_i und a_{i+1} eine dritte Aktion auftreten kann.

Neben den sehr einfachen konkreten Plänen, wie sie gerade vorgestellt wurden, gibt es auch komplexere Pläne, die durch bestimmte Arten von Abstraktionen charakterisieren werden können, d.h. insbesondere, daß damit auch Eigenschaften der mit den Plänen assoziierten Äquivalenzklassen von Aktionsintervallen beschrieben werden.

Objektabstraktion: Pläne können in ihren Basisaktionen Variablen für Objekte der Anwendungsdomäne enthalten.

Konditionale: Es wird in einem Plan von dem Gelten bestimmter Vorbedingungen abstrahiert, in dem man eine vollständige Fallunterscheidung einführt.

Nichtdeterminismus: Ein Plan enthält explizit oder implizit eine nichtdeterministische Aktionsauswahl. Explizit, indem eine Disjunktion von Teilplänen enthalten ist; implizit, indem eine abstrakte (künstliche) Aktion als Teilplan erscheint, die nichts anderes darstellt als die abkürzende Schreibweise eines expliziten Nichtdeterminismus – die Beziehung wird mittels eines Abstraktionsaxioms sichergestellt.

Nichtlinearität: Hierbei wird von der konkreten Ausführungsreihenfolge von Aktionen oder Teilplänen abstrahiert; zwischen ihnen besteht nur noch eine partielle Ordnung.

Unterbrechbarkeit: Man abstrahiert von dem konkreten Ausführungszeitpunkt eines Teilplanes, konkrete zeitliche Beziehungen innerhalb eines Planes werden in relative zeitliche Beziehungen überführt, so daß eine Unterbrechung des Planes durch den Einschub nicht explizit genannter Aktionen möglich ist. Die "Bruchstelle" des Planes wird durch die Einführung von bestimmten Randbedingungen spezifiziert.

Iteration: Dies entspricht der Zusammenfassung einer iterativen Folge von Teilplänen durch ein Iterationsschema (ein Schleifenkonstrukt).

Implizite Aktionsdetermination: Aktionen werden nicht explizit benannt, sondern es wird eine Bedingung spezifiziert, die durch erlaubte Aktionen erreicht bzw. erhalten werden soll.

Anhang A enthält eine vollständige syntaktische Charakterisierung von Planformeln, wie sie im weiteren Verlauf dieses Kapitels als Konkretisierungen der obigen Abstraktionen mit den entsprechenden Strategien ihrer Erzeugung eingeführt werden.

[22]Die oben beschriebene Planspezifikation beschreibt keinen eindeutigen Endzustand, sondern nennt nur zwei zeitlich in Bezug stehende Zustände, die auf jeden Fall erreicht werden müssen; dannach dürfen allerdings auch noch andere Zustände erreicht werden.

4.2.4 Planen als *Modellverfeinerung*

Wie im vorherigen Abschnitt ausgeführt, dient ein Plan in Verbindung mit einer Planspezifikation und den verfügbaren Aktionsaxiomen dazu, eine Klasse von Modellen, insbesondere Aktionsintervalle, zu charakterisieren bzw. mehr oder weniger detailliert einzugrenzen. Eine Planspezifikation als Ausgangspunkt vollzieht bereits die erste Einschränkung an die relevanten Modelle. Typischerweise beschreibt sie einen Ausgangszustand und einen oder mehrere Zielzustände. Ein Beispiel einer solchen Einschränkung ist in einer Modellcharakterisierung gegeben als

$$\sigma = \left\langle \begin{array}{|c|}\hline pre \\ ? \\ \hline \end{array}_0 \ldots \begin{array}{|c|}\hline goal_1 \\ ? \\ \hline \end{array}_i \ldots \begin{array}{|c|}\hline goal_2 \\ \\ \hline \end{array}_n \right\rangle \quad \text{für } 0 \leq i \leq n.$$

Hierin sind zwei Zielzustände spezifiziert; über die notwendige Intervallänge ist noch keine absolute Aussage gemacht. Die Planspezifikation legt nun also eine erste notwendige *Verfeinerung* relevanter Modelle (Intervalle) fest. Ziel ist es, wohlfundierte weitere Verfeinerungen vorzunehmen, bis die Existenz eines ausführbaren Aktionsintervalles gesichert ist. Unter Verwendung von Aktionsaxiomen (siehe Abschnitt 4.2.1) kann diese Existenz gegebenenfalls garantiert werden und führt dann z.B. zu folgender Verfeinerung:

$$\rho\sigma = \left\langle \begin{array}{|c|}\hline pre \\ a_1 \\ \hline \end{array}_0 \sigma_1 \right\rangle \circ \langle \sigma_1\sigma_2 \rangle \circ \langle \sigma_2\sigma_3 \rangle \circ \ldots \left\langle \sigma_{i-1} \begin{array}{|c|}\hline goal_1 \\ a_{i+1} \\ \hline \end{array}_i \right\rangle \circ \ldots \circ \left\langle \sigma_{n-1} \begin{array}{|c|}\hline goal_2 \\ \\ \hline \end{array}_n \right\rangle .$$

Das Intervall σ ist dabei in n Teilintervalle $\theta_1, \ldots, \theta_n$ der Länge 1 unterteilt worden. Jedes Intervall θ_i ist jeweils durch ein Aktionsaxiom eindeutig charakterisierbar, d.h. σ^2_{i-1} bildet auf eine Aktion a_i ab, deren Vorbedingungen in σ_{i-1} und deren Effekte in σ_i gelten. Daß zwei benachbarte Intervalle jeweils einen Zustand gemeinsam haben, stellt eine lückenlose Verbindung des ersten Zustandes σ_0 zum letzten Zustand σ_n her. Ausgehend von σ gibt es im allgemeinen eine Vielzahl möglicher Verfeinerungsschritte, um letztendlich zu einer Aufteilung $\rho\sigma$ zu gelangen.
Die einzelnen Schritte gehören dabei unterschiedlichen Kategorien an:[23]

Aufsplittung: Ein Intervall σ wird dabei aufgeteilt in σ' und σ'' wobei gilt $\sigma = \sigma' \circ \sigma''$, d.h. es wird eine Dekomposition mit eindeutigem Verschmelzungspunkt vorgenommen. Durch die Aufsplittung entstehen jeweils Charakterisierungen für die Teilintervalle, die in einer bestimmten Beziehung zur Charakterisierung von σ stehen.

Längeneinführung: Ein Intervall σ wird dadurch näher eingeschränkt, indem seine Länge charakterisiert wird. Dies erfolgt durch die Festlegung eines konstanten, minimalen und/oder maximalen Wertes.

Aktionsetablierung: Ein Intervall σ wird konform zu einer Aktion a eingeschränkt, d.h. σ_0 wird nun zusätzlich durch die Vorbedingungen von a charakterisiert, entsprechend σ_1, falls existent, durch die Effekte von a.

[23] Zur Erläuterung beziehen wir uns auf den Intervallbegriff.

Sicherheitsbedingungseinführung: Ein Intervall σ wird eingeschränkt, indem in allen seinen Zuständen eine bestimmte Bedingung ϕ gelten muß – eine *starke* Sicherheitsbedingung –, oder indem ϕ lediglich im letzten Zustand von σ gelten muß – eine *schwache* Sicherheitsbedingung.

Aufgabenreduktion: Ein Intervall σ wird dabei in möglicherweise mehreren Zuständen durch zusätzliche Bedingungen spezialisiert.[24]

Nebenläufigkeit: Ein Intervall σ wird dadurch eingeschränkt, daß es gleichzeitig verschiedene individuelle Verfeinerungen erfüllen muß.

Korrelation: Ein Intervall σ wird verfeinert, indem zwei nebenläufige Verfeinerungen ihre Anforderungen teilweise (oder ganz) zur Übereinstimmung bringen.[25]

Disjunktion: Ein Intervall σ wird verfeinert, indem eine Auswahlmöglichkeit zwischen alternativen spezialisierten Verfeinerungen eingeräumt wird.[26]

Determinierung: Die Anzahl der bestehenden alternativen Verfeinerungen für ein Intervall σ wird reduziert.

Eine Verfeinerung $\rho\sigma$, wie oben beschrieben, kann durch die Abfolge von Verfeinerungsschritten, wie sie Abbildung 4.4 zeigt, entstehen.
In Abschnitt 4.2.7 werden Beweisstrategien zur Plangenerierung in LLP* vorgestellt, die sich der obigen Verfeinerungsschritte bedienen.

4.2.5 Aktionsaxiomenschemata

In diesem Abschnitt werden wir auf die Verwendungsmöglichkeiten von Aktionsaxiomen während des deduktiven Planens eingehen. Es wird sich zeigen, daß potentiell unendlich viele Aktionsaxiome notwendig wären, um Beweise beliebiger Planspezifikationen führen zu können. Umgehen kann man dieses Problem, indem man ein Aktionsaxiomenschema angibt, aus dem nach Bedarf die Aktionsaxiome gebildet werden, die gerade benötigt werden.
Es gibt nun zwei wesentliche Gründe, Aktionsaxiome während der Ausführung des Beweises einer Planspezifikationsformel einzusetzen:

1. um mit der entsprechenden Aktion ein primäres Ziel zu erreichen, d.h. der unmittelbare Effekt der Aktion ist gefragt;

2. um zu gewährleisten, daß bestimmte Eigenschaften bzgl. der Aktion invariant sind, d.h. das Rahmenproblem zu lösen.

[24] Eine Aktionsetablierung ist ein Spezialfall dieses Verfeinerungsschrittes.

[25] Dies bedeutet z.B., daß eine allgemeinere Verfeinerung die Spezialisierung einer nebenläufigen Verfeinerung übernimmt.

[26] Bei alternativen Verfeinerungen ist es erlaubt, daß der Schnitt zwischen den durch sie charakterisierten Modellen auch leer sein kann (im allgemeinen leer sein wird).

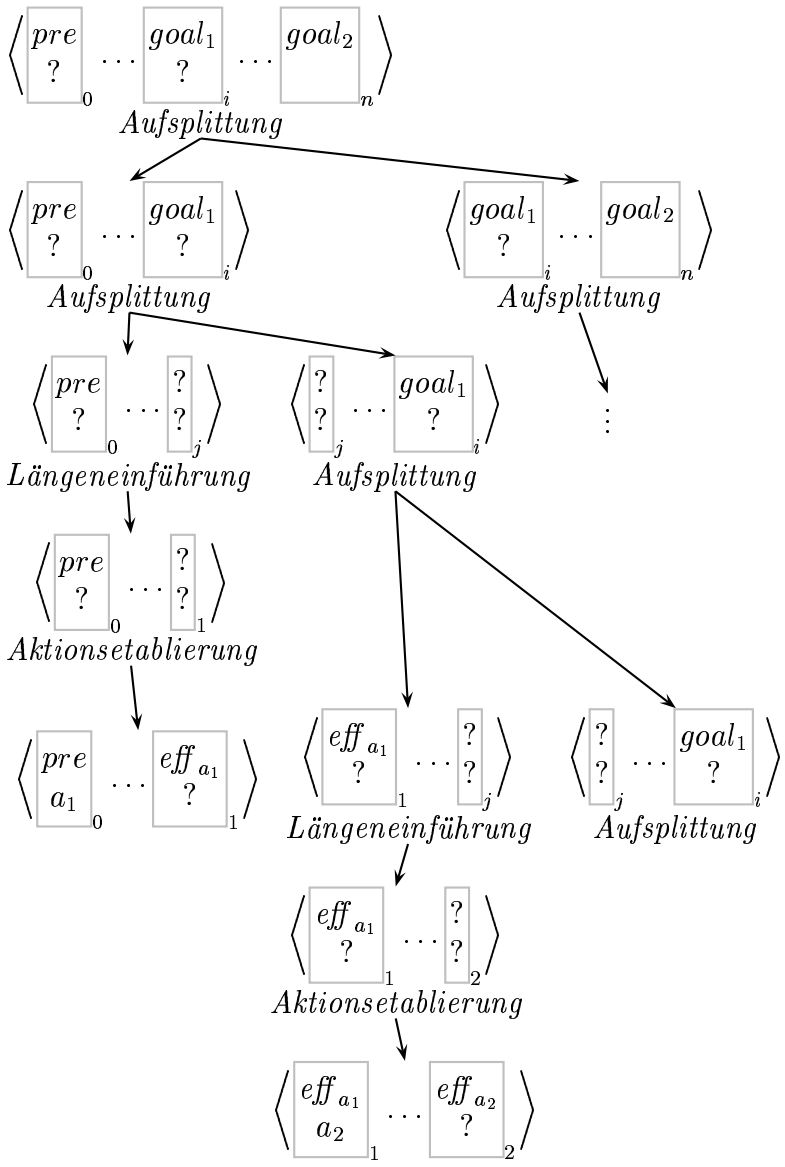


Abbildung 4.4: Beispielabfolge von Verfeinerungsschritten

Zunächst gehen wir auf den ersten Punkt ein. Wie in Abschnitt 4.1 beschrieben, sind LLP bzw. LLP* getypte Sprachen, wobei gemäß den Definitionen von speziellen Teilklassen dieser Sprachen, wie etwa was Planformeln sind, Metatypen existieren, die Formeln charakterisieren, d.h. bestimmte Eigenschaften mit Formeln assoziieren. Die Metatypen dienen in der Beweisansteuerung dazu, die Instantiierung von Metavariablen gezielt zu steuern. Die Metavariable P aus einer Planspezifikationsformel sollte natürlich nur durch Planformeln instantiiert werden, da eine andere Instantiierung nicht zu einem erfolgreichen Beweis führen kann.[27]
Um einen erfolgreichen Beweis führen zu können, muß man durch die Anwendung von Regeln irgendwann logische Axiome erreichen, insbesondere muß man Instantiierungen für vorhandene Metavariablen finden. Der einzig sinnvolle Weg, dies zu tun, ist die Verwendung von Aktionsaxiomen. Um sie verwenden zu können, muß man während der Beweisführung eine Sequenz S der Form

$$\Gamma,\ \mathsf{P} \Rightarrow \bigcirc\psi \qquad \text{mit } \psi \text{ eine modalfreie Formel}$$

erreichen. Sie besagt, daß man ausgehend von Vorbedingungen Γ und einer noch nicht bekannten Planformel P im nächsten Zustand die Formel ψ erreicht. Jetzt kann man die zu einer entsprechenden Aktionsaxiomeninstanz korrespondierende Regel anwenden und logische Axiome erreichen. Eine Aktionsaxiomeninstanz $pre \wedge ex(a) \rightarrow \bigcirc\psi$[28] entspricht einer Regel R:

$$\frac{\Delta \Rightarrow pre \wedge ex(a)}{\Delta \Rightarrow \bigcirc\psi}$$

Wendet man R auf S an, erreicht man eine Sequenz S':

$$\Gamma,\ \mathsf{P} \Rightarrow pre \wedge ex(a)$$

Durch die Anwendung einer *links*-Einführungsregel kann die Planmetavariable P nun mit der Planformel $ex(a)$ instantiiert werden. Durch eine iterative Anwendung der Standardregel $\wedge - rechts$ (bis die Konjunktion pre vollständig zerlegt ist) wird S' zerlegt in $S'_1,\ldots,S'_n$. Auf die Sequenzen S'_i wird iterativ die Standardregel $\wedge - links$ angewendet, bis man ein logisches Axiom erreicht oder die Regel nicht mehr anwendbar ist. Abbildung 4.5 skizziert noch einmal den Zusammenhang.
Die oben beschriebene Verwendung einer neuen Sequenzenregel R kann wie folgt durch eine Standardableitung begründet werden. Betrachten wir dazu die Sequenz

$$\Gamma,\ \mathsf{P} \Rightarrow \bigcirc \mathit{eff}(x),$$

die während eines Beweises entstanden ist. Eine entsprechende Ableitung ist in Abbildung 4.6 skizziert.
Wie man leicht sieht, erhält man mit Sequenz (6) die gleiche noch offene Sequenz, wie dies auch beim Vorgehen unter Verwendung der Regel R (vgl. Abbildung 4.5) auftritt –

[27] Wenn wir einmal davon absehen, daß man P mit $false$ instantiieren könnte und dann natürlich immer einen erfolgreichen Beweis geführt hätte.

[28] Wir nehmen an, daß pre für eine Konjunktion atomarer Formeln $pre_1, \ldots, pre_n$ steht.

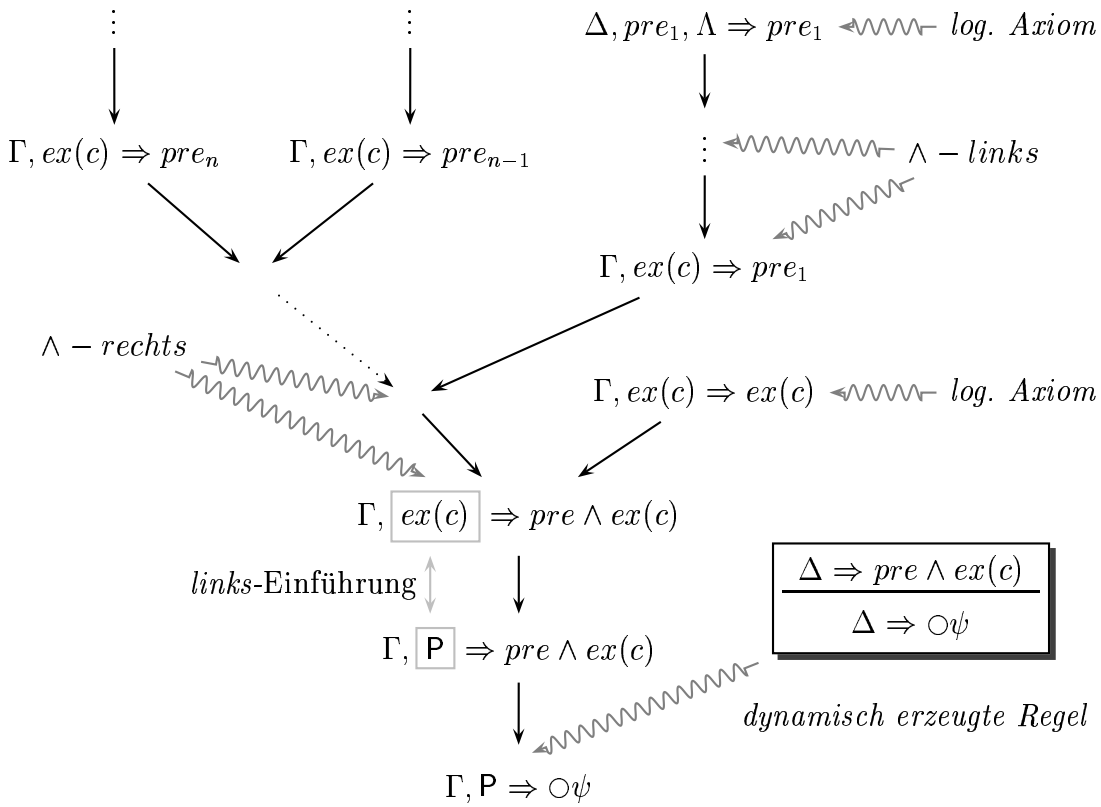


Abbildung 4.5: Einführung einer Basisaktion

unter Vernachlässigung der Parametrisierung x. Eine differenziertere Betrachtung dieses Vorgehens erfolgt zu Ende dieses Abschnittes.

Wir wenden uns nun dem zweiten Motiv zu, Aktionsaxiome zu verwenden, der Gewährleistung von Invarianten. Möchte man Pläne erstellen, die aus einer Hintereinanderausführung mehrerer Aktionen bestehen, so muß man auf irgendeine Weise sogenannte Rahmenaxiome (vgl. Abschnitt 2.1) verwenden. Es wird während des Planens notwendig sein zu wissen, ob bestimmte Bedingungen oder welche Bedingungen zu einem bestimmten Zeitpunkt gelten. Das heißt, man muß z.B. bestimmen können, welche initialen Bedingungen (Vorbedingungen) nach Ausführung eines Planes noch gelten. Ein Rahmenaxiom beschreibt gerade die Bedingungen, die unverändert bleiben.
Um zu erkennen, an welcher Stelle der Beweisführung Rahmenaxiome notwendig sind und was sie konkret beschreiben, betrachten wir eine abstrakte Planungssituation etwas genauer. Ein Planungsproblem sei folgendermaßen beschrieben:

$$pre \wedge \mathsf{P} \Rightarrow \Diamond[g_1 \wedge g_2]$$

Dabei seien pre initiale Bedingungen, also gegebene Vorbedingungen, g_1 und g_2 zwei Ziele, die irgendwann erreicht sein sollen und P eine Metavariable für einen noch zu bestimmenden Plan, der die beiden Ziele erreicht. Nehmen wir weiterhin an, daß die beiden Ziele durch einen sequentiellen Plan, bestehend aus zwei Teilplänen, erreicht werden können,

			Angewandte Regeln
(1)	Γ, P	$\Rightarrow \bigcirc \mathit{eff}(x)$	cut-Regel
(2)		$\Rightarrow \forall y\ \mathit{pre}(y) \wedge \mathit{ex}(a(y)) \rightarrow \bigcirc \mathit{eff}(y)$	Als nichtlogisches Axiom angenommen
(3)	$\forall y\ \mathit{pre}(y) \wedge \mathit{ex}(a(y)) \rightarrow \bigcirc \mathit{eff}(y), \Gamma, \mathsf{P}$	$\Rightarrow \bigcirc \mathit{eff}(x)$	$\forall$-links$\{y/x\}$
(4)	$\mathit{pre}(x) \wedge \mathit{ex}(a(x)) \rightarrow \bigcirc \mathit{eff}(x), \Gamma, \mathsf{P}$	$\Rightarrow \bigcirc \mathit{eff}(x)$	$\rightarrow$ -links
(5)	$\bigcirc \mathit{eff}(x)$	$\Rightarrow \bigcirc \mathit{eff}(x)$	
(6)	Γ, P	$\Rightarrow \mathit{pre}(x) \wedge \mathit{ex}(a(x))$	$\wedge$-rechts
(7)	Γ, P	$\Rightarrow \mathit{pre}(x)$	
(8)	Γ, P	$\Rightarrow \mathit{ex}(a(x))$	*links*-Einführung

Abbildung 4.6: Rechtfertigung einer Aktionsregel

d.h. für P wird eine Ersetzung $\mathsf{P}_1; \mathsf{P}_2$ eingeführt, wobei P_1 und P_2 wiederum Metavariablen sind. Während der Beweisführung werden dann folgende Teilprobleme zu lösen sein:[29]

1. $\mathit{pre} \wedge \mathsf{P}_1 \Rightarrow \bigcirc g_1$
2. $\mathit{pre} \wedge \mathsf{P}_1 \Rightarrow \bigcirc \mathsf{pre}_2$
3. $\mathsf{pre}_2 \wedge \mathsf{P}_2 \Rightarrow \bigcirc g_2$
4. $\mathsf{pre}_2 \wedge g_1 \wedge \mathsf{P}_2 \Rightarrow \bigcirc g_1$

Mit der Lösung von Teilproblem 1 erzielt man einen Plan (eine metavariablenfreie Instantiierung von P_1), der das Teilziel g_1 ausgehend vom initialen Zustand erreicht; dies kann z.B. wie in Abbildung 4.5 skizziert geschehen. Nehmen wir an, es ergebe sich eine Instantiierung $\mathit{ex}(a_1) \wedge \odot\bigcirc F$.
Mit der Lösung von Teilproblem 3 erzielt man einen Plan, der das Teilziel g_2 ausgehend von Bedingungen, die nach Ausführung von Plan P_1 gelten, erreicht. Nehmen wir hier an, daß sich als Planformel $\mathit{ex}(a_2) \wedge \odot\bigcirc F$ ergibt. Das Teilproblem 2 und die Verwendung der gleichen Metavariablen pre_2 stellt die Verbindung sicher. Teilproblem 4 schließlich sichert, daß das erste Ziel auch noch nach Ausführung des zweiten Teilplanes gilt.
Zur Lösung des zweiten Teilproblems benötigt man nun ein Rahmenaxiom bzgl. der Aktion a_1. In der das Teilproblem beschreibenden Sequenz ist zunächst noch eine Metavariable, pre_2, für eine Bedingung enthalten.
Die Aufgabe besteht nun darin, für diese Metavariable eine geeignete Instantiierung zu finden, die beschreibt, welche Bedingungen nach Ausführung von a_1 neben dem Effekt g_1 noch gelten, z.B. weil sie vor der Ausführung schon gegolten haben und invariant bzgl. a_1

[29] Wie man zu diesen Teilproblemen kommt, wird in einem späteren Abschnitt genau beschrieben.

sind. Beschreiben läßt sich solch eine Invarianz in Form eines zusätzlichen Axioms für die Aktion a_1, z.B.:

$$pre_{a_1} \wedge \phi \wedge ex(a_1) \rightarrow \bigcirc[eff_{a_1} \wedge \phi]$$

In Ergänzung der Definition 4.2.2 beschreibt man eine solche Invarianz allgemein als *Rahmenaxiom* gemäß der folgenden Definition:

Definition 4.2.4 *Ein Rahmenaxiom für eine Aktion a hat die folgende Form:*

$$\forall \vec{x}\ pre(\vec{x}, \vec{v}) \wedge \phi(\vec{x}, \vec{y}, \vec{v}, \vec{w}) \wedge ex(a(\vec{x})) \rightarrow \bigcirc[\mathit{eff}(\vec{x}, \vec{v}) \wedge \phi(\vec{x}, \vec{y}, \vec{v}, \vec{w})]$$

$\phi(\vec{x}, \vec{y}, \vec{v}, \vec{w})$ ist eine modalfreie Formel; $\vec{y}$ bezeichnet einen Vektor von $\vec{x}$ verschiedener globaler Variablen, $\vec{w}$ einen Vektor von $\vec{v}$ verschiedener lokaler Variablen.

Im allgemeinen kann man bzgl. einer Aktion a für beliebig viele Formeln $\phi()$ ein solches Axiom formulieren.
Verwenden kann man ein Rahmenaxiom in der gleichen Weise, wie man ein gewöhnliches Aktionsaxiom verwendet, also analog zu der in Abbildung 4.5 beschriebenen Weise. Man verwendet dabei jedoch zusätzlich eine Einführungsregel für die Metavariable pre_2.
Zur Lösung des vierten Teilproblems muß man ebenfalls ein Rahmenaxiom verwenden, in diesem Fall eines der Aktion a_2. Genaugenommen muß man hier überprüfen, ob g_1 denn eine Invariante der Aktion a_2 darstellt; hingegen bei Problem 2 muß man ein *geeignetes* Rahmenaxiom bzgl. a_1 auswählen.

Wie bereits oben bemerkt, benötigt man zur Beschreibung der Invarianten von Aktionen in einer Planungsdomäne im allgemeinen unendliche viele Rahmenaxiome. Aus praktischer Sicht ist selbst die notwendige Beschränkung auf eine endliche Zahl zur Abdeckung *typischer* Probleme der aktuellen Planungsdomäne nicht effektiv.[30] Der einzig sinnvolle Ausweg ist, einen Mechanismus zu haben, der Rahmenaxiome angepaßt an die aktuellen Anforderungen während des Beweisvorganges generiert. Aus der Axiomatisierungssicht ist dies eine Zugriffsfunktion auf die theoretisch vorhandene unendliche Menge von Rahmenaxiomen; man verändert also nicht dynamisch die Axiomatisierung einer Domäne, sondern fokussiert nur eine bestimmte Menge von Axiomen.
Um einen Mechanismus zur Axiomengenerierung angeben zu können, ändern wir zunächst die Repräsentation von Aktionsaxiomen. Wir nutzen dazu aus, daß die Formulierung von einer Gleichungsmenge zur Beschreibung von Eigenschaften, die nach Ausführung einer Aktion gelten, reduziert werden kann auf die Spezifikation minimaler, simultan durchzuführender Veränderungen bestimmter Eigenschaften; diese Veränderungen können als eine Form von Zuweisungen angesehen werden. Die in Definition 4.2.2 eingeführte Axiomenform verallgemeinern wir deshalb zu dem folgenden Axiomenschema:

Definition 4.2.5 *Aktionsaxiome und zugehörige Rahmenaxiome werden durch ein Axiomenschema*

$$\forall \vec{x}\ ex(a(\vec{x})) \rightarrow [\bigwedge_{i=1}^{n} [pre_i(\vec{x}, \vec{v}) \wedge \mathcal{FT}(A^i, \mathsf{E}) \rightarrow \bigcirc \mathsf{E}]]$$

[30] Zum einen stellt sich die Frage, was denn typische Probleme sind, zum anderen wäre es mehr als mühsam, alle resultierenden Axiome aufzuschreiben.

beschrieben. Hierin ist E *eine Metavariable für eine LLP-Formel.* A^i *sind jeweils Mengen gerichteter Gleichungen der Art* t/t' *mit* $t' = t$ *ist in der Menge* Basiseigenschaften *(siehe Definition 4.2.1) der aktuellen Planungsdomäne enthalten;* t *ist ein funktionsfreier Term.* $\mathcal{FT}$ *ist eine formeltransformierende Funktion, die LLP-Formeln auf LLP-Formeln abbildet; sie berechnet den dynamischen Teil der schwächsten Vorbedingungen*[31] *der Formel* E *bzgl. der Aktion* a *unter der Bedingung der Ersetzungen gegeben durch* A^i*. Den statischen und zweiten Teil bildet* $pre_i(\vec{x}, \vec{v})$*.*

Die Mengen A^i beschreiben gerade die Auswirkungen der Aktion a unter den jeweils verschiedenen Ausgangsbedingungen. Zwei äquivalente Darstellungen eines Axiomenschemas der gerade beschriebenen Art sind die Aufteilung in einzelne Axiome

$$\begin{aligned} &\forall \vec{x}\ ex(a(\vec{x})) \wedge pre_1(\vec{x}, \vec{v}) \wedge \mathcal{FT}(A^1, \mathsf{E}) \rightarrow \bigcirc \mathsf{E} \\ &\quad \vdots \\ &\forall \vec{x}\ ex(a(\vec{x})) \wedge pre_n(\vec{x}, \vec{v}) \wedge \mathcal{FT}(A^n, \mathsf{E}) \rightarrow \bigcirc \mathsf{E} \end{aligned}$$

beziehungsweise die disjunktive Verknüpfung der unterschiedlichen Vorbedingungen gemäß

$$\forall \vec{x}\ [ex(a(\vec{x})) \wedge \bigvee_{i=1}^{n} [pre_i(\vec{x}, \vec{v}) \wedge \mathcal{FT}(A^i, \mathsf{E})]] \rightarrow \bigcirc \mathsf{E}\ .$$

Betrachten wir der Einfachheit halber zunächst den Fall, daß der Parameter n den Wert 1 trägt, also daß ein eindeutiger Effekt spezifiziert ist. Gegeben eine Instantiierung ϕ der Metavariablen E, dann erhält das Axiomenschema die folgende Interpretation:
$\mathcal{FT}(A^1, \phi)$ transformiert die Formel ϕ zu ϕ', so daß ein Axiom

$$\forall \vec{x}\ ex(a(\vec{x})) \wedge pre_1(\vec{x}, \vec{v}) \wedge \phi' \rightarrow \bigcirc \phi$$

entsteht. Es legt hinreichende Bedingungen fest, die sicherstellen, daß nach Ausführung der Aktion a die Formel ϕ gilt. Die Menge A^1 von Ersetzungen wird bei der Bildung von ϕ' so berücksichtigt, daß in ϕ' ausreichende Bedingungen spezifiziert sind, so daß die Anwendung von A^1 auf ϕ in einer konsistenten d.h. widerspruchsfreien Formel resultiert, falls $\phi' \wedge pre_1(\vec{x}, \vec{v})$ nicht schon selbst inkonsistent d.h. widerspruchsvoll ist. Wenn $pre_1(\vec{x}, \vec{v}) \wedge \phi'$ die *schwächste Vorbedingung* von ϕ bzgl. der Aktion a ist, dann bedeutet dies, daß für alle Formeln ψ mit

$$\forall \vec{x}\ ex(a(\vec{x})) \wedge pre_1(\vec{x}, \vec{v}) \wedge \psi \rightarrow \bigcirc \phi$$

die Beziehung $\psi \rightarrow \phi'$ gilt.
Die syntaktische Form der Axiomenschemata erlaubt nun neben der Spezifikation singulärer Bedingungen und Wertveränderungen sowohl die Spezifikation allquantifizierter Vorbedingungen als auch (eingeschränkter) allquantifizierter unmittelbarer Effekte. Eine Axiomeninstanz kann z.B. durch die Form

$$\forall \vec{x}\vec{y}\ ex(a(\vec{x})) \wedge pre_1(\vec{x}, \vec{v}) \wedge T \rightarrow \bigcirc \phi(\vec{y}, \vec{v})$$

[31] Dijkstra's Semantik für Programmiersprachen (vgl. [Dijkstra 76]) prägt diesen Begriff; ebenso wird dort auch der Begriff *stärkste Nachbedingung* eingeführt.

gegeben sein, was aufgrund gültiger Umformungen der Formel

$$\forall \vec{x}\ ex(a(\vec{x})) \wedge pre_1(\vec{x},\vec{v}) \wedge T \rightarrow \forall \vec{y}\ \bigcirc\phi(\vec{y},\vec{v})$$

entspricht. Da Vorbedingungen und Effekte sich hier keine quantifizierten Variablen teilen, sind sie als unabhängig allquantifiziert gegeben. Aufgrund der in Definition 4.2.5 definierten Beschränkung auf eine spezifizierbare Menge von einzelnen Wertveränderungen sind nur Konjunktionen positiver Literale als Effekte einer Aktion spezifizierbar. Die Beschränkung auf positive Literale bedeutet keine Einschränkung, da in dem verfolgten Ansatz, eine Domäne charakterisierende Prädikate oder Relationen als boolesche Funktionen repräsentiert sind; ein verneintes Prädikat wird durch einen Wert "$\ldots = 0$" dargestellt.

Ein Axiomenschema kann man auch noch auf eine andere Art verwenden. Gegeben sei der Wert der Funktion $\mathcal{FT}(A^1, \mathsf{E}) \equiv \phi'$ bei bekanntem A^1 und noch unbekanntem E. Dies beschreibt die Situation, daß man wissen möchte, welche Bedingungen ϕ notwendigerweise nach Ausführung der Aktion a gelten, wenn vorher ϕ' gegolten hat. Damit beschreibt man unmittelbar *notwendige Nachbedingungen* von ϕ' bzgl. der Aktion a. ϕ erhält man also, wenn man die Umkehrfunktion $\mathcal{FT}^{-1}(A^1, \phi')$ berechnet. Da die Funktion $\mathcal{FT}$ nicht injektiv ist, hat man mit $\mathcal{FT}^{-1}$ die *stärksten* Nachbedingungen nicht eindeutig erfaßt. Sie sind beschrieben als

$$\mathcal{FT}^{-1}(A^1, \phi' \wedge pre_1(\vec{x},\vec{v})) \wedge \mathsf{konjunktion}(A^1)\ .$$

Die Funktion konjunktion wandelt dabei die Menge gerichteter Gleichungen in eine Konjunktion von (ungerichteten) Gleichungen um.
Wir betrachten nun an einem konkreten Beispiel mögliche Axiomenschemainstanzen. Das Schema einer Aktion *akt* sei wie folgt gegeben:

$$\forall x\ ex(akt(x)) \wedge name(v) = x \wedge prop(v) = 0 \wedge \mathcal{FT}(\{1/prop(v)\}, \mathsf{E}) \rightarrow \bigcirc\mathsf{E}$$

0 und 1 seien Kodierungen boolscher Werte. Die folgende Tabelle enthält für eine gegebene Instantiierung von E die entsprechenden schwächsten, dynamischen Vorbedingungen.

Instantiierung für E	$\mathcal{FT}(\{1/prop(v)\}, \mathsf{E})$
$prop(v) = 1$	$1 = 1 \equiv T$
$prop(w) = 1$	$[v = w \wedge T] \vee [v \neq w \wedge prop(w) = 1]$
$prop(w) = 0$	$[v = w \wedge F] \vee [v \neq w \wedge prop(w) = 0]$
$prop(v) = 0$	$1 = 0 \equiv F$
$prop\_x(w) = 0$	$prop\_x(w) = 0$

Die folgende Tabelle enthält Beispiele, die notwendige Nachbedingungen einer Formel ϕ' bzgl. der Aktion *akt* beschreiben.

ϕ'	$\mathcal{FT}^{-1}(\{1/prop(v)\}, \phi')$
$prop(v) = 0$	T
$prop(w) = 1$	$v = w \vee [v \neq w \wedge prop(w) = 1]$
$prop(w) = 0$	$v = w \vee [v \neq w \wedge prop(w) = 0]$
$prop\_x(w) = 0$	$prop\_x(w) = 0$

α steht in der Tabelle für eine beliebige Formel; dies gilt, weil ϕ' unter der entsprechenden Bedingung eine inkonsistente Zustandsbeschreibung entstehen läßt.

Für den Fall $1 < n$ beschreibt das Axiomenschema in Definition 4.2.5 eine Aktion mit konditionalen Effekten. Am Ende des Abschnittes 4.2.1 wurden verschiedene Voraussetzungen für konditionale Effekte genannt. Es gelten von nun an die Voraussetzungen 1., 2. und 3.(a).
Hat man eine konkrete Aktion mit n unterschiedlichen Ausgangsbedingungen definiert, so führt die Bildung einer Aktioneninstanz bzgl. einer Instantiierung ϕ für die Metavariable E zu n individuellen Ausprägungen schwächster Vorbedingungen, d.h. im Sinne der Aufteilung der gesamten Aktion führt dies zu n einzelnen Aktionsaxiomen. Es gibt unter diesen Axiomen im allgemeinen solche, die nur noch eine triviale Aussagekraft haben, da die in ihnen spezifizierten Vorbedingungen einen inkonsistenten Zustand beschreiben. Das folgende Beispiel verdeutlicht dies. Ausgangspunkt sei das Aktionsaxiomenschema

$$\begin{aligned}\forall x\ ex(akt(x)) \rightarrow \\ [[name(v) = x \wedge prop(v) = 0 \wedge propy(v) = 0 \wedge \mathcal{FT}(\{1/prop(v)\}, \mathsf{E}) \rightarrow \bigcirc\mathsf{E}] \wedge \\ [name(v) = x \wedge prop(v) = 0 \wedge propy(v) = 1 \wedge \mathcal{FT}(\{0/propy(v)\}, \mathsf{E}) \rightarrow \bigcirc\mathsf{E}]]\end{aligned}$$

das für eine Aktion *akt* zwei konditionale Effekte spezifiziert. Die Instantiierung des Schemas geschieht bzgl. der Formel $prop(v) = 1$. Man erhält dann die zwei folgenden Axiome:

$$\forall x\ ex(akt(x)) \wedge name(v) = x \wedge prop(v) = 0 \wedge propx(v) = 0 \rightarrow \bigcirc prop(v) = 1$$

$$\begin{aligned}\forall x\ ex(akt(x)) \wedge name(v) = x \wedge \boxed{prop(v) = 0} \wedge propy(v) = 1 \wedge \boxed{prop(v) = 1} \\ \rightarrow \bigcirc prop(v) = 1\end{aligned}$$

Das letztere dieser Axiome stellt eine triviale Aussage dar, da die spezifizierten Vorbedingungen inkonsistent sind. Daß solche Axiome entstehen können, hat natürlich Konsequenzen für ihre Verwendung beim Beweis von Planspezifikationen, wie zu Beginn dieses Abschnittes beschrieben. Da sie nutzlos sind, also ein Beweis wie in Abbildung 4.5 mit ihnen scheitert, sollten sie von einer Verwendung vorweg schon ausgeschlossen werden. Zur Bildung von Axiomeninstanzen gehört deshalb auch die Sicherung ihrer Nichttrivialität mit hinzu.[32]
Wie oben bereits für den Fall einer konditionalfreien Aktion erwähnt, besteht die zweite Möglichkeit, ein Axiomenschema zu instantiieren, darin, die stärksten Nachbedingungen einer Aktion zu erhalten. Unter der angenommenen Voraussetzung, daß alle statischen Vorbedingungen sich gegenseitig widersprechen, bedeutet dies, daß für ein Schema gemäß Definition 4.2.5 jeweils höchstens eine der Funktionen $\mathcal{FT}^{-1}(A^i, \phi)$ für gegebenes ϕ relevant ist.

Die folgende Definition faßt nun zusammen, was effektive schwächste Vorbedingungen bzw. effektive stärkste Nachbedingungen bzgl. einer allgemeinen Aktion sind.

Definition 4.2.6 *Gegeben sei ein Axiomenschema für eine Aktion a mit konditionalen Effekten:*

$$\forall \vec{x}\ ex(a(\vec{x})) \rightarrow [\bigwedge_{i=1}^{n} [pre_i(\vec{x}, \vec{v}) \wedge \mathcal{FT}(A^i, \mathsf{E}) \rightarrow \bigcirc\mathsf{E}]]$$

[32] Es ist offensichtlich, daß man mit dem Ausschluß trivialer Axiome nichts an Vollständigkeit bzgl. der generierbaren Pläne verliert.

Für eine gegebene Instantiierung ϕ für die Metavariable E *sind die effektiven schwächsten Vorbedingungen von ϕ bzgl. der Aktion a gegeben als*

$$\bigvee_{j=1}^{m} [pre_j(\vec{x}, \vec{v}) \wedge \mathcal{FT}(A^j, \phi)]$$

mit $0 \leq m \leq n$ und $pre_j(\vec{x}, \vec{v}) \wedge \mathcal{FT}(A^j, \phi)$ ist konsistent.
Für eine gegebene Formel ψ sind die effektiven stärksten Nachbedingungen von ψ bzgl. der Aktion a gegeben als

$$\bigvee_{j=1}^{m} [\mathcal{FT}^{-1}(A^j, \psi \wedge pre_j(\vec{x}, \vec{v})) \wedge \mathsf{konjunktion}(A^j)]$$

mit $0 \leq m \leq n$ und $pre_j(\vec{x}, \vec{v}) \wedge \psi$ ist konsistent. Im konkreten Fall der Ausführung der Aktion a gilt meist $m \leq 1$, da höchstens für ein j die Beziehung $\psi \rightarrow pre_j(\vec{x}, \vec{v})$ gilt.[33]

4.2.5.1 Die Grundlagen von Regression und Progression

Die Berechnung der schwächsten Vorbedingungen bzgl. einer Aktion wird in der Literatur zum Planen als *Regression* bezeichnet; der inverse Vorgang, die stärksten Nachbedingungen zu berechnen, wird als *Progression* bezeichnet (vgl. dazu [Waldinger 77, Rosenschein 81, Kautz 82, Pednault 86]). Angepaßt an die Logik LLP geben wir nun ein Verfahren an, die Funktionen $\mathcal{FT}$ bzw. $\mathcal{FT}^{-1}$ zu berechnen und darauf aufbauend dann die effektiven schwächsten Vorbedingungen bzw. stärksten Nachbedingungen zu bestimmen.

Die Berechnung von $\mathcal{FT}$ Auf der Grundlage von Definition 4.2.6 geben wir nun an, wie $\mathcal{FT}(A^j, \phi)$ für eine gegeben Formel ϕ berechnet wird. A^j sei gegeben als die Menge $\{x_1^0/t_1(\vec{x}', v_1), \ldots, x_n^0/t_n(\vec{x}', v_n)\}$ von Ersetzungen. In Abhängigkeit von der syntaktischen Struktur von ϕ wird $\mathcal{FT}(A^j, \phi)$ berechnet.

- $\mathcal{FT}(A^j, s(\vec{y}', w) = y^0) \equiv s(\vec{y}', w) = y^0$

  Es gilt: $s(\vec{y}', w) = y^0 \in \text{Basiseigenschaften}_{\mathcal{D}}$, s symbolisiert einen beliebig tief strukturierten Term; ferner gilt: $\nexists i$ mit $1 \leq i \leq n$, so daß $unification(s(\vec{y}', w), t_i(\vec{x}', v_i)) =: \varsigma$[34] , d.h. eine Objekteigenschaft $s(\vec{y}', w)$ wird von der betrachteten Aktion nicht manipuliert.

- $\mathcal{FT}(A^j, s(\vec{y}', w) = y^0) \equiv \bigvee_{i=1}^{m} \mathcal{FT}(\{x_{k_i}^0/t_{k_i}(\vec{x}', v_{k_i})\}, s(\vec{y}', w) = y^0)$

  Es gilt:

$$\mathcal{FT}(\{x_{k_i}^0/t_{k_i}(\vec{x}', v_{k_i})\}, s(\vec{y}', w) = y^0) \equiv \begin{array}{l} \mathsf{konjunktion}(\varsigma_{1i} \cup \varsigma_{2i}) \vee \\ [\neg \mathsf{konjunktion}(\varsigma_{1i}) \wedge s(\vec{y}', w) = y^0] \end{array}$$

[33] ψ beschreibt dann im allgemeinen einen aktuellen deterministischen Weltzustand.

[34] *unification* bezeichnet eine Prozedur, die die Standardtermunifikation nach [Robinson 65] durchführt; für den Fall, daß sie erfolgreich endet, liefert sie entsprechende Substitutionen zurück.

falls ς_{1i} durch die Unifikation $\mathit{unification}(s(\vec{y}', w), t_{k_i}(\vec{x}', v_{k_i})) =: \varsigma_{1i}$ für ein k_i mit $1 \leq k_i \leq n$ entsteht und ς_{2i} sich ergibt als $\varsigma_{2i} := \mathit{unification}(y^0, x^0_{k_i})$; für $i \neq j$ gilt $k_i \neq k_j$; m ist maximal. Wenn ς_{2i} nicht existiert, dann gilt

$$\mathsf{konjunktion}(\varsigma_{1i} \cup \varsigma_{2i}) := F$$

da dann $s(\vec{y}', w) = y^0$ der Eigenschaft $t_{k_i}(\vec{x}', v_{k_i}) = x^0_{k_i}$ widerspricht.

Beschrieben wird zum einen die Situation, daß die Eigenschaft $s(\vec{y}', w) = y^0$ von der mit A^j assoziierten Aktion a schon erzeugt wird, und zwar unter einer Bedingung $\mathsf{konjunktion}(\varsigma_{1i} \cup \varsigma_{2i})$; zum anderen wird beschrieben, unter welchen Bedingungen, $\neg\mathsf{konjunktion}(\varsigma_{1i})$, sie schon vorher gegolten hat und dann nach Ausführung der Aktion a noch immer gilt.

Für eine Substitutionsmenge $\varsigma = \emptyset$ ergibt sich $\mathsf{konjunktion}(\varsigma) := T$.

- $\mathcal{FT}(A^j, \phi \wedge \psi) \equiv \mathcal{FT}(A^j, \phi) \wedge \mathcal{FT}(A^j, \psi)$
- $\mathcal{FT}(A^j, \phi \vee \psi) \equiv \mathcal{FT}(A^j, \phi) \vee \mathcal{FT}(A^j, \psi)$
- $\mathcal{FT}(A^j, \neg\phi) \equiv \neg\mathcal{FT}(A^j, \phi)$
- $\mathcal{FT}(A^j, \forall y\ \phi(y)) \equiv \forall y\ \mathcal{FT}(A^j, \phi(y))$

Die Berechnung von $\mathcal{FT}^{-1}$ Wiederum auf der Grundlage von Definition 4.2.6 wird angegeben, wie $\mathcal{FT}^{-1}(A^j, \phi)$ für eine gegebene Formel ϕ berechnet wird. A^j sei gegeben als die Menge $\{x^0_1/t_1(\vec{x}', v_1), \ldots, x^0_n/t_n(\vec{x}', v_n)\}$ von Ersetzungen. In Abhängigkeit von der syntaktischen Struktur von ϕ wird $\mathcal{FT}^{-1}(A^j, \phi)$ berechnet.

- $\mathcal{FT}^{-1}(A^j, s(\vec{y}', w) = y^0) \equiv s(\vec{y}', w) = y^0$

  Es gilt: $s(\vec{y}', w) = y^0 \in \text{Basiseigenschaften}_{\mathcal{D}}$, s symbolisiert einen beliebig tief strukturierten Term; ferner gilt: $\nexists i$ mit $1 \leq i \leq n$, so daß $\mathit{unification}(s(\vec{y}', w), t_i(\vec{x}', v_i)) =: \varsigma$, d.h. eine Objekteigenschaft $s(\vec{y}', w)$ wird von der betrachteten Aktion nicht manipuliert.

- $\mathcal{FT}^{-1}(A^j, s(\vec{y}', w) = y^0) \equiv \bigvee_{i=1}^{m}[\mathsf{konjunktion}(\varsigma_{1i}) \vee [\neg\mathsf{konjunktion}(\varsigma_{1i}) \wedge s(\vec{y}', w) = y^0]]$

  falls ς_{1i} durch die Unifikation $\mathit{unification}(s(\vec{y}', w), t_{k_i}(\vec{x}', v_{k_i})) =: \varsigma_{1i}$ für ein k_i mit $1 \leq k_i \leq n$ entsteht; für $i \neq j$ gilt $k_i \neq k_j$; m ist maximal.

  Beschrieben wird einerseits, daß die Eigenschaft $s(\vec{y}', w) = y^0$ von der mit A^j assoziierten Aktion a manipuliert wird – unter der Bedingung $\mathsf{konjunktion}(\varsigma_{1i})$ – und andererseits, wann sie unverändert bleibt – unter der Bedingung $\neg\mathsf{konjunktion}(\varsigma_{1i})$

- $\mathcal{FT}^{-1}(A^j, \phi \wedge \psi) \equiv \mathcal{FT}^{-1}(A^j, \phi) \wedge \mathcal{FT}^{-1}(A^j, \psi)$
- $\mathcal{FT}^{-1}(A^j, \phi \vee \psi) \equiv \mathcal{FT}^{-1}(A^j, \phi) \vee \mathcal{FT}^{-1}(A^j, \psi)$
- $\mathcal{FT}^{-1}(A^j, \neg\phi) \equiv \neg\mathcal{FT}^{-1}(A^j, \phi)$
- $\mathcal{FT}^{-1}(A^j, \forall y\ \phi(y)) \equiv \forall y\ \mathcal{FT}^{-1}(A^j, \phi(y))$

In Definition 4.2.6 wird, um die Effektivität der erzeugten Axiomeninstanzen zu gewährleisten, die Konsistenz oder Widerspruchsfreiheit einer modalfreien Formel gefordert. Eine geeignete Methode diese Widerspruchsfreiheit zu sichern, ist das *Resolutionsverfahren* anzuwenden (vgl. z.B. [Gallier 86] zum Einsatz des Verfahrens im Kontext von Sequenzenkalkülen). Will man die Widerspruchsfreiheit einer Formel Δ zeigen, so heißt dies, daß aus der Klauselform von Δ nicht durch Anwendung des Resolutionsverfahrens die leere Klausel ableitbar sein darf.

Zwei Strategien zum Einsatz von Rahmenaxiomen Im Zusammenhang mit der Definition 4.2.4 wurde veranschaulicht, wie Rahmenaxiome beim Planen eingesetzt werden können; beschrieben wurde dort eine progressive Strategie. Wir skizzieren noch einmal die prinzipielle Vorgehensweise und stellen ihr eine regressive Strategie gegenüber.
Betrachten wir zunächst in Abbildung 4.7 die schematische Darstellung der Planungszustände, wie sie bei einem progressiven Vorgehen nach dem Finden eines Planes bestehen.[35]

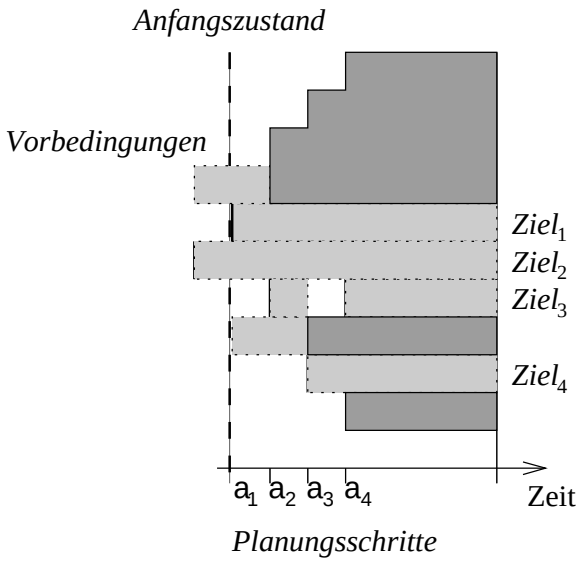


Abbildung 4.7: Progressive Strategie

Die progressive Strategie berechnet nach jedem einzelnen Planungsschritt, genauer, immer dann, wenn eine Aktion eingeführt wurde, um ein bestimmtes Ziel zu erreichen, die stärksten Nachbedingungen. Sie schreibt deshalb den Anfangszustand durch die Ausführung von Aktionen solange in die Zukunft fort, bis die spezifizierten Ziele erreicht sind. Man kennt somit in jedem Zwischenzustand alle Bedingungen, die bezogen auf die initialen Vorbedingungen gelten. Darunter sind dann neben den eigentlichen Zielen $Ziel_1, \ldots, Ziel_4$, die man erreichen will, auch noch andere Bedingungen, die sich aus den Vorbedingungen oder zwischenzeitlich erreichten Teilzielen ableiten. In der Grafik sind diese Bedingungen dunkelgrau markiert. Sie können während der Suche nach dem restlichen Plan genutzt werden, u.U. sind sie aber auch *überflüssig*. Eine progressive Strategie benötigt zur Erreichung eines effektiven Verhaltens eine zielgerichtete Steuerung der Aktionsanwendung (siehe auch Seite 140 ff.).

[35]Ein waagerechter Balken in der Graphik bedeutet, daß eine Eigenschaft (Ziel) über eine gewisse Zeit hinweg gilt.

Abbildung 4.8 zeigt hingegen die Planungszustände, die bei der Regression entstehen.

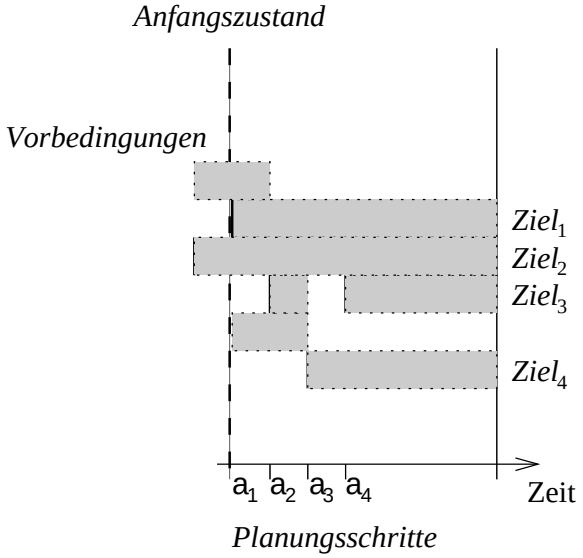


Abbildung 4.8: Regressive Strategie

Die regressive Strategie berechnet bei jedem Planungsschritt die schwächsten Vorbedingungen, d.h. man will nur die Vorbedingungen bestimmen, die minimal notwendig sind, um das vorgegebene Ziel mit einem bestimmten Plan zu erreichen. Die wesentliche Aufgabe besteht darin, einen Erzeuger für die benötigten Vorbedingungen zu suchen. Man rechnet also den partiell charakterisierten Zielzustand solange bzgl. ausgeführter Aktionen zurück, bis man zu einem Zustand kommt, der sich unmittelbar aus dem spezifizierten Ausgangszustand ergibt. Die regressive Strategie konzentriert sich nur auf notwendige Zustandsbeschreibungen. Im Unterschied zum Schema der Progression in Abbildung 4.7 fehlen hier die dunkelgrauen Bereiche, da sie gerade Bedingungen enthalten, die zur Erreichung der Ziele nicht notwendig sind.
Aus dem prinzipiellen Vorgehen der beiden genannten Strategien heraus ergeben sich charakteristische Eigenschaften, die ihre Verwendung während des deduktiven Planens beeinflussen. Tabelle 4.1 stellt verschiedene Eigenschaften gegenüber.
Abschnitt 4.2.7 geht auf die Unterschiede der Ansätze im Hinblick auf die jeweilige Realisierung entsprechender Beweisstrategien noch einmal detaillierter ein.

4.2.5.2 Transformation in Sequenzenregeln

In Abbildung 4.6 wurde das prinzipielle Vorgehen bei der Umsetzung eines Aktionsaxioms in eine Sequenzenregel, die während des Beweises einer Planspezifikationsformel eingesetzt werden kann, beschrieben. Dort wird davon ausgegangen, daß alle Vorbedingungen und Effekte der verwendeten Aktion mit der selben Variable parametrisiert sind, d.h. die mit Anwendung der Regel $\forall$-*links* einhergehende Substitution der quantifizierten Variable ist hier vollständig durch die vorhandene Problemspezifikation getriggert.
Wir betrachten dieses Vorgehen nun etwas differenzierter, insbesondere hinsichtlich der sich entsprechend Definition 4.2.5 ergebenden Varianten eines Aktionsaxioms für den progressiven bzw. regressiven Einsatz. Wir können zwischen vier prinzipiellen Möglichkeiten

Eigenschaften	**Progression**	**Regression**
Ausnutzung von Nebeneffekten von Aktionen	bis zum Planende	lokal
Direkte Berücksichtigung von	Anfangszustand	Zielzustand
Vermeidung redundanter Aktionen	möglich	erfordert Planreparatur
Aufdeckung von Zielkonflikten	oft zu spät	sobald sie entstehen
Größe eines Planungszustandes abhängig von	Anfangszustand Aktionseffekte	Zielzustand Aktionsvorbedingungen
Sicherheit eines partiellen Planes, den Zielzustand zu erreichen	nicht gegeben	gegeben
Sicherheit eines partiellen Planes, im Anfangszustand ausführbar zu sein	gegeben	nicht gegeben

Tabelle 4.1: Eigenschaften von Progression vs. Regression

zur Formulierung eines Aktionsaxioms hinsichtlich der Parametrisierung seiner spezifizierten Vorbedingungen und Effekte unterscheiden:[36]

1. Alle Vorbedingungen und Effekte der Aktion a sind mit der selben Variable parametrisiert:
$$\forall x\ pre(x) \wedge ex(a(x)) \rightarrow \bigcirc goal(x)$$
2. Es gibt zwei unterschiedliche Parameter, mit denen jeweils individuelle Effekte assoziiert sind:[37]
$$\forall xy\ pre(x,y) \wedge ex(a(x,y)) \rightarrow \bigcirc[goal(x) \wedge goal'(y)]$$
3. Es gibt durch einen zweiten Parameter individuell spezifizierte (existenzquantifizierte) Vorbedingungen:
$$\forall xy\ pre(x) \wedge pre'(y) \wedge ex(a(x)) \rightarrow \bigcirc goal(x)$$
4. Es sind durch einen zweiten Parameter individuell spezifizierte (allquantifizierte) Effekte gegeben:
$$\forall xy\ pre(x) \wedge ex(a(x)) \rightarrow \bigcirc[goal(x) \wedge goal'(y)]$$

Die aktuelle Problemspezifikation für die regressive Verwendung eines Aktionsaxioms zur Problemreduktion sei gegeben als $\mathsf{pre} \wedge \mathsf{P} \Rightarrow \bigcirc goal(z)$. Bezüglich des ersten Falles kann ebenso wie in Abbildung 4.6 verfahren werden. Es entsteht eine Regel

$$\frac{\Gamma \Rightarrow pre(z) \wedge ex(a(z))}{\Gamma \Rightarrow \bigcirc goal(z)}$$

[36] Wir fokussieren hier nur die quantifizierbaren Variablen.

[37] Ob auch separierte Vorbedingungen gegeben sind, ist für unsere Betrachtung irrelevant.

Betrachtet man den zweiten Fall, so ergibt sich für die Anwendung der $\forall$-*links*-Regel bzgl. der Variable y die Schwierigkeit, eine geeignete Substitution zu finden, die den Abschluß des Gesamtbeweises möglichst unterstützt. Der Parameter y bezeichnet wie der Parameter x ein individuelles, an der Aktion a beteiligtes Objekt. Die mit diesem Objekt assoziierte spezifische Eigenschaft, ist notwendige Voraussetzung, daß die Aktion a tatsächlich angewandt werden kann. Nimmt man etwa eine Substitution {s/y} an, so müssen alle irgendwann im fortschreitenden Beweis entstehenden Beweisprobleme mit Sukzedens $pre(z, s)$ durch ein logisches Axiom der Art

$$\Delta, pre(z, s) \Rightarrow pre(z, s)$$

abgeschlossen werden, d.h. man bestimmt mit Festlegung der Substitution die Voraussetzung, die abgesichert aus den global spezifizierten Annahmen folgern muß, damit die Aktion a so konkret wie beschrieben verwendet werden kann.
Aus der aktuell vorhandenen Problemspezifikation kann jedoch kein geeigneter Substitutionskandidat für y bestimmt werden. Es stehen nun zwei prinzipielle Methoden zur Verfügung, mit diesem Problem umzugehen:

- Man betrachtet den globalen Zustand des aktuellen Beweises und versucht, einen geeigneten Kandidaten zu bestimmen. Bei mehreren resultierenden Alternativen entscheidet man sich für eine Konkretisierung. Scheitert der weitere Beweis irgendwann aufgrund dieser Entscheidung, muß man bis zum Entscheidungspunkt zwischen den Alternativen zurücksetzen und kann sich für eine der übrigen Konkretisierungen entscheiden. Geeignete Kandidaten kann man z.B. bestimmen, indem man die Voraussetzungen der initialen Problemspezifikation auf mit $pre(.,.)$ kompatible Formeln untersucht und die dort enthaltenen Parameter übernimmt.[38]

- Im Gegensatz zur vollständigen Entscheidung über die Verwendung einer Aktion kann man die Entscheidung über die konkrete Substitution von y auch bis zur Erreichung eines geeigneten Planungszustandes aufschieben. Man setzt als Substitution also zunächst eine Metavariable ein, die dann irgendwann später konkretisiert werden kann. Zum Zeitpunkt dieser Konkretisierung können dann alle Vorkommen der Metavariable durch den bestimmten Wert ersetzt werden.[39] Mit diesem Vorgehen hat man nun auch die Möglichkeit, Konkretisierungen verwenden zu können, die man bei der oben beschriebenen ersten Möglichkeit nicht ohne großen Aufwand zu treiben als geeignet erkennen konnte, z.B. durch eine zwischenzeitlich eingefügte Aktion neu eingeführte Objekte. Die entstehende Regel ist dann von der Form

$$\frac{\Gamma \Rightarrow pre(z, \mathsf{G}^{\mathsf{i}}) \wedge ex(a(z, \mathsf{G}^{\mathsf{i}}))}{\Gamma \Rightarrow \bigcirc goal(z)}$$

  G^{i} bezeichne die i-te neu eingeführte Objektmetavariable. Analog zu den Regeln zur Behandlung von Formelmetavariablen (vgl. Seite 104) gibt es für Objektmetavariablen Regeln zu ihrer kontrollierten Instantiierung.

[38] Das macht jedoch nur für freie Variablen oder Konstanten Sinn.

[39] Es handelt sich dabei um eine im taktischen Theorembeweisen häufig verwendete Technik (vgl. [Felty & Howe 94]).

Im dritten der oben beschriebenen Fälle ist eine existenzquantifizierte Vorbedingung spezifiziert. Bevor nun eine $\forall$-*links*-Regel angewandt wird, wird das Axiom zunächst umgeformt in die äquivalente Form:

$$\forall x\ pre(x) \wedge [\exists y\ pre'(y)] \wedge ex(a(x)) \rightarrow \bigcirc goal(x)$$

Nun kann die $\forall$-*links*-Regel mit der Substitution $\{z/x\}$ angewendet werden und es resultiert eine Regel

$$\frac{\Gamma \Rightarrow pre(z) \wedge \exists y\ pre'(y) \wedge ex(a(z))}{\Gamma \Rightarrow \bigcirc goal(z)}$$

Eine Instantiierung von y mithilfe der Regel $\exists$-*rechts* wird erst vorgenommen, wenn $\exists y\ pre'(y)$ einer verläßlichen Voraussetzung gegenübersteht, es erfolgt also wiederum die Aufschiebung einer Entscheidung bis eine entscheidungsrelevante Situation aufgetreten ist. Man hat hier aber auch die Option, eine Objektmetavariable für die Variable y einzuführen, da die $\exists$-*rechts*-Regel die gleiche Wirkung, wie die $\forall$-*links*-Regel hat, die Variable y kann durch einen beliebigen Term substituiert werden.

Der vierte Fall ist für ein regressives Vorgehen nicht von Bedeutung, es resultiert die gleiche Regel wie im ersten Fall.

Zur Erläuterung der progressiven Verwendung eines Aktionsaxioms gehen wir von der Problemspezifikation $\phi(z) \wedge \mathsf{P} \Rightarrow \bigcirc[goal(z) \wedge \mathsf{pre}']$ aus, d.h. es stehen konkrete Anfangsbedingungen ϕ zur Verfügung und die zu verwendende Aktion muß den Effekt $goal(z)$ erreichen.
Die erste Form der Aktionsaxiome stellt wiederum einen unkritischen Fall dar. Die Bildung der entsprechenden Regel erfolgt wie oben beschrieben.
Bezüglich des zweiten möglichen Falles eines Aktionsaxioms ergibt sich u.U. wiederum die Schwierigkeit, eine geeignete Substitution für die Variable y zu bestimmen. Die Ersetzung von x ist getriggert durch den von der Problemspezifikation geforderten unmittelbaren Effekt. Zu den stärksten Nachbedingungen der Aktion a gehören jedoch alle Effekte der Aktion, auch die mit y parametrisierten. Außerdem fordert der Test auf Anwendbarkeit der Aktion unter den Voraussetzungen $\phi(z)$, daß mindestens eine Substitution für y gefunden werden kann, so daß die Folgerung von $pre(z, .)$ aus $\phi(z)$ bewiesen werden kann. Tragen zum Beweis dieser Beziehung verschiedene Substitutionen für y bei, so befindet man sich wiederum in dem Dilemma, eine Entscheidung für das Fortschreiten des Beweises treffen zu müssen, ohne ihre Auswirkung lokal abschätzen zu können – es entstehen jeweils unterschiedliche sekundäre Effekte der Aktion. Es bietet sich auch wiederum die Verwendung einer Objektmetavariable als Instantiierung für die Variable y an. Die Regel, die dann entsteht, ist z.B. von der Form:

$$\frac{\Gamma \Rightarrow pre(z, \mathsf{G}^{\mathsf{i}}) \wedge ex(a(z, \mathsf{G}^{\mathsf{i}}))}{\Gamma \Rightarrow \bigcirc[goal(z) \wedge goal'(\mathsf{G}^{\mathsf{i}})]}$$

Bezogen auf die aktuelle Problemspezifikation bleibt, wenn man vor Anwendung der Aktionsregel eine geeignete Instantiierungsregel bzgl. pre' anwendet, ein (offenes) Problem

$$\phi(z) \Rightarrow pre(z, \mathsf{G}^{\mathsf{i}}),$$

von dem man weiß, daß es für bestimmte Fälle bewiesen werden kann. Der Abschluß dieses Problems kann nun verschoben werden, bis durch die Verwendung von $goal'(\mathsf{G}^i)$ in anderen Teilen des Gesamtbeweises eine Forderung nach einer bestimmten Substitution aufkommt. Man kann dann prüfen, ob dem Substitutionswunsch nachgekommen werden kann und dazu versuchen, das oben genannte offene Problem unter dieser Substitution zum Abschluß zu bringen. Die Instantiierung der Metavariablen ist hier also mit einer Bedingung verknüpft.

Der dritte Fall der Aktionsaxiomformulierung kann analog zur Verfahrensweise beim regressiven Vorgehen behandelt werden. Die entstehenden existenzquantifizierten Vorbedingungen können hier nun jedoch ohne Aufschiebung direkt bewiesen werden.

Im vierten Fall kann nach entsprechender äquivalenter Umformung ein Axiom

$$\forall x\ pre(x) \wedge ex(a(x)) \rightarrow \bigcirc[goal(x) \wedge \forall y\ goal'(y)]$$

erhalten werden, d.h. es ist ein allquantifizierter Effekt beschrieben. Durch eine entsprechende Instantiierung der Metavariable pre' kann er im weiteren Beweis verwendet werden und z.B. effektiv zum Beweis eines allquantifizierten Zieles beitragen[40].

4.2.6 Was spricht für einen Sequenzenkalkül?

Betrachtet man alleine die zwei prinzipiellen Vorgehensweisen zum deduktiven Planen, wie sie im letzten Abschnitt beschrieben wurden, so verlangt die Realisierung beider Prinzipien durch Beweisstrategien zum einen nach einer flexiblen Sprache zur Formulierung solcher Strategien (vgl. Abschnitt 3.3.2), zum anderen, und was noch essentieller ist, nach einem Beweiskalkül, der eine flexible und auf natürliche Weise nachvollziehbare Ansteuerung ermöglicht.

Wie bereits in Abschnitt 4.1.3 beschrieben, verwenden wir zur Beweisführung in LLP* einen Sequenzenkalkül. Zu welchen positiven Auswirkungen dies bei der Durchführung des Planens führt, wird nun erläutert.

Ein Beweissystem, das Sequenzen verwendet, spiegelt mit seinen Beweisregeln sehr klar die Semantik der logischen Konnektoren wider. Dies bedeutet, daß sich die Struktur eines Beweises an der Struktur der zu beweisenden Formel orientiert. Sequenzenkalkülbeweise unterstützen damit unmittelbar die strukturierte Vorgehensweise beim Planen. Die Beweisstruktur korrespondiert dabei im allgemeinen mit der Planstruktur (vgl. Abbildung 4.9). Es können z.B. auf einfache Weise Teilbäume des gesamten Beweisbaumes lokalisiert werden, die der Suche nach einer bestimmten Teilplaninstantiierung entsprechen.

Weiterhin entspricht die Reihenfolge, in der eine Planstruktur entsteht, im wesentlichen auch der Reihenfolge, in der entsprechende Teilbeweise geführt werden.

Die natürliche Korrespondenz zwischen Plänen und Beweisen erleichtert die Ansteuerung des Beweisers und führt dazu, daß der Beweisaufwand kontrollierbar bleibt. Der Sequenzenbeweis läßt sich nach planungsspezifischen Gesichtspunkten steuern, insbesondere bleiben strukturelle Eigenschaften der Planspezifikationsformel[41] bei der Beweisführung

[40] Eine $\forall$-*rechts*-Regel gibt dort z.B. eine Instantiierung vor, so daß dann mithilfe einer entsprechenden $\forall$-*links*-Regel ein passender *Partner* zum Abschluß des Beweises eingesetzt werden kann.

[41] Zum Beispiel zeitliche Ordnungen zwischen Zielen.

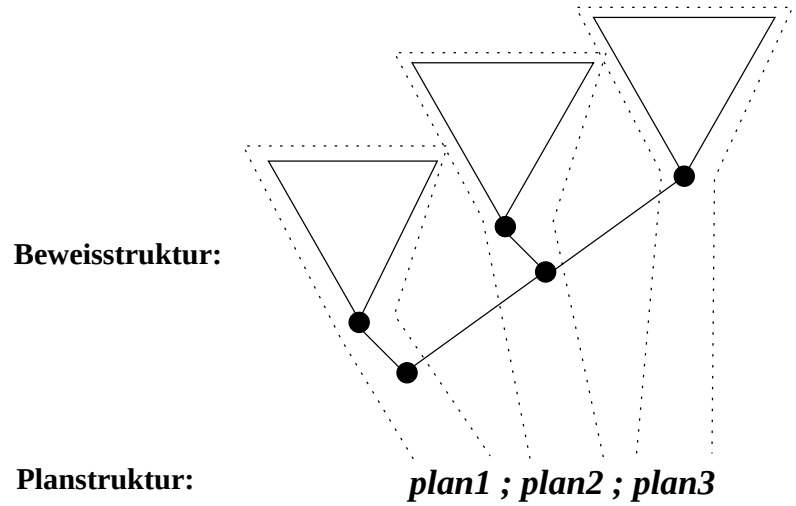


Abbildung 4.9: Korrespondenz Beweis-, Planstruktur

erhalten. Da eine Sequenz die vollständige Information über den momentanen Zustand einer Ableitung enthält, ist auch ein ausgezeichneter Ausgangspunkt für ein inkrementelles und interaktives Vorgehen gegeben. Die Erklärung und Darstellung noch offener Ableitungen gegenüber einem menschlichen Benutzer kann sehr einfach realisiert werden. Der Beweisprozeß zeichnet sich durch Transparenz aus und ist jederzeit vom Benutzer kontrollierbar. Als Kalkül des natürlichen Schließens eignet sich ein Sequenzenkalkül auch für die Integration in ein Beweispräsentationssystem, das z.B. einen Beweis unter Einsatz von Textplanungstechniken in natürliche Sprache ausgibt [Huang & Fiedler 96, Huang & Fiedler 97].

Würde man etwa im Gegensatz zu einem Sequenzenkalkül einen Resolutionskalkül[42] zum Beweisen verwenden, wäre die Korrespondenz zwischen Plan- und Beweisstruktur nicht so offensichtlich gegeben, da Resolutionsbeweise schwer verständlich und wenig intuitiv sind. Hierbei geht bereits bei der Übersetzung der Planspezifikationsformel in Klauselform wichtige strukturelle Information verloren. Ähnliches gilt für den Einsatz des Matrixverfahrens bzw. der Konnektionsmethode; ausgehend von der konjunktiven Normalform einer Formel wird ihre Widersprüchlichkeit direkt aus ihrer inneren Struktur abgeleitet.
Eine weitere Variante von Gentzenkalkülen sind Tableauxkalküle. Als negativer Testkalkül versucht er, die Widersprüchlichkeit der Negation einer als allgemeingültig nachzuweisenden Formel durch die Anwendung von Dekompositionsregeln zu beweisen. Es wird eine kompaktere Darstellung als bei Sequenzenkalkülen gewählt (vgl. z.B. [Bibel 92] für eine Darstellung verschiedener Kalküle zur automatischen Deduktion).

Der während des Planens/Beweisens entstehende Beweisbaum gibt ein Protokoll des Planungsprozesses wieder (in seiner Gesamtheit, wenn Teilbeweisfehlschläge, d.h. falsche Annahmen, ebenfalls mitprotokolliert werden). Er enthält Information, die z.B. die Grundlage für die Erklärung des Vorgehens bei der Generierung eines spezifischen Planes liefert. Ein Plan in der Rolle eines Transmitters – wie in der vorliegenden Arbeit propagiert – kann durch diese Zusatzinformationen angereichert werden. Es handelt sich dabei etwa

[42]Dazu könnte z.B. die Modallogik LLP zunächst in reine Prädikatenlogik übersetzt werden. Ansätze zur Übersetzung von Modallogiken finden sich in [Ohlbach 93].

um die Zuordnung von Teilplänen zu Teilzielen, um Information über die Reihenfolge der Bearbeitung bestimmter Teilziele oder um die Information über fehlende Vorbedingungen, die durch bestimmte Teilpläne zunächst erreicht werden mußten. Solche Information kann für Tutoringzwecke bzw. für das tiefere Verständnis des Planungsvorganges sehr nützlich sein.

Im folgenden skizzieren wir kurz den prinzipiellen Aufbau eines Sequenzenkalkülbeweisers zur Realisierung eines deduktiven Planers (vgl. Abbildung 4.10).

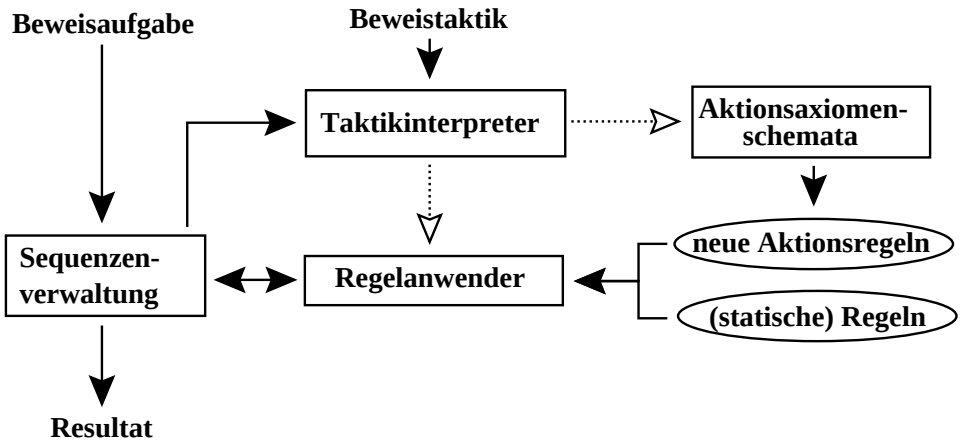


Abbildung 4.10: Prinzipielle Beweiserarchitektur

Als Eingabe stehen zum einen die Beweisaufgabe – in unserem Fall eine Planspezifikation – und zum anderen eine Beweistaktik zur Verfügung. Die Beweisaufgabe bildet die Wurzel(sequenz) des entstehenden Beweisbaumes, der die Abhängigkeiten aller entstehenden Sequenzen protokolliert. Neue Sequenzen entstehen durch die Anwendung von Sequenzenregeln auf eine bereits vorhandene Sequenz. Die Beweistaktik ist nun eine Strategie, die durch entsprechende Interpretation die Anwendung von Sequenzenregeln auf offene Sequenzen steuert. Dazu kann es notwendig sein, daß zunächst aus Aktionsaxiomenschemata bestimmte Aktionsregeln, d.h. in Regelform gepackte Aktionsaxiomeninstanzen, erzeugt werden (vgl. dazu die Erklärungen in Abschnitt 4.2.5). Da ihre Menge theoretisch unendlich groß sein kann, werden diese Aktionsregeln nur auf konkrete Anforderung hin erzeugt. Alle übrigen Sequenzenregeln, ob Basisregeln oder abgeleitete Regeln, sind in diesem Sinne als *statisch* zu bezeichnen.

4.2.7 Generierung Abstrakter Pläne

Die folgenden Teilabschnitte zeigen jeweils, wie die in Abschnitt 4.2.3 eingeführten, unterschiedlichen Abstraktionsstufen von Plänen erzeugt werden können. Unterstützt wird dies durch einen Einblick in die Beweisstruktur der jeweiligen Planungsproblematik. Die Generierung konkreter Pläne, in dem Sinne, daß diese Pläne keine variablen Parameter enthalten, kann als Spezialfall der Generierung abstrakter Pläne angesehen werden.

4.2.7.1 Sequentielle Pläne

Sequentielle Pläne sind die strukturell einfachsten Pläne, die generiert werden können. Sie sind zusammengesetzt aus einer Sequenz von Basisaktionen, deren jeweiliges Ausführungs-

intervall auf die Länge 1 reduziert ist:

$$ex(a_1) \wedge \odot\bigcirc F; ex(a_2) \wedge \odot\bigcirc F; \ldots; ex(a_n) \wedge \odot\bigcirc F$$

Dieser sequentielle Plan enthält eine direkte Abfolge von n Aktionen.

Ausgehend von einem konkreten Planungsproblem, gegeben durch eine Planspezifikation, sind folgende Fragen von einer Beweisstrategie zu beantworten:

1. Wie teilt man das Planungsproblem in geeignete Teilprobleme auf?
2. Wie assoziiert man Teilpläne mit Teilproblemen?
3. Wie erreicht man, daß eine Kette von Teilplänen, eine semantische Verbindung zwischen den spezifizierten Anfangsbedingungen und den Zielen gewährleistet?

Die Lösung der ersten beiden Fragestellungen kann in dem verwendeten Beweiskalkül nur durch die Anwendung spezieller *kompositionaler* Sequenzenregeln erreicht werden.
Für die Erzeugung linearer Pläne spielt Regel 1 dabei die wesentliche Rolle. Ihr unterliegt die folgende Motivation: man geht von einer Planspezifikation aus mit Vorbedingungen ϕ, Zielbeschreibung γ und der Annahme, daß eine Komposition zweier Pläne P_1 und P_2 die Spezifikation erfüllen wird. Dann ergibt sich unmittelbar aus der Semantik des Modaloperators ";", daß die Verbindungsstelle zwischen P_1 und P_2 durch eine Eigenschaft ψ derart beschrieben ist, daß ψ im letzten Zustand des von P_1 dominierten Intervalles gilt[43] und im ersten Zustand des von P_2 dominierten Intervalles.

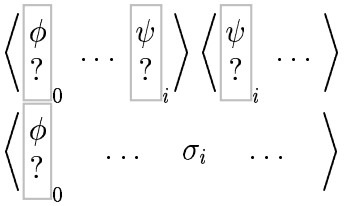
Regel 1 (goal refine)

$$\frac{\phi,\ \mathsf{P}_1 \Rightarrow \Diamond[\bigcirc F \wedge \psi] \qquad \psi,\ \mathsf{P}_2 \Rightarrow \Diamond\gamma}{\phi,\ \mathsf{P}_1;\mathsf{P}_2 \Rightarrow \Diamond\gamma}$$

ϕ und ψ sind modalfreie Formeln, P_1 und P_2 sind Planmetavariablen bzw. Formeln aus LLP, γ ist eine LLP-Formel.*

□

Diese Regel beschreibt einen allgemeinen Verfeinerungsschritt in Form einer Aufsplittung (vgl. Abschnitt 4.2.4). Die kompositionale Modellverfeinerung kann wie folgt charakterisiert werden.



Es erfolgt also eine Aufteilung des gesamten Intervalles in zwei aufeinanderfolgende Teilintervalle mit einer notwendigen Anforderung ψ an den *Verschmelzungspunkt* σ_i, $0 \leq i$. Aufsplittungsregeln wie Regel 1 sind notwendig, um ein Planungsproblem in individuelle, beherrschbare Teilprobleme zu zerlegen.

[43] Wenn ein Intervall eine Formel $\Diamond\bigcirc F$ erfüllt, dann ist es ein endliches Intervall.

Die Korrektheit von Regel 1 ergibt sich durch folgende Ableitung:

			Angewandte Regeln
(1)	$\phi,\ \mathsf{P}_1;\mathsf{P}_2$	$\Rightarrow \Diamond[\gamma]$	*Äquivalenz*
(2)	$[\phi \wedge\ \mathsf{P}_1];\mathsf{P}_2$	$\Rightarrow \Diamond[\gamma]$	*Modus Ponens*
(3)	$[\phi \wedge\ \mathsf{P}_1]$	$\Rightarrow \Diamond[\bigcirc F \wedge \psi]$	*redundant zu (9)*
(4)	$[\phi \wedge\ \mathsf{P}_1 \wedge \Diamond[\bigcirc F \wedge \psi]];\mathsf{P}_2$	$\Rightarrow \Diamond[\gamma]$	*Äquivalenz*
(5)	$[[\phi \wedge\ \mathsf{P}_1];[\bigcirc F \wedge \psi]];\mathsf{P}_2$	$\Rightarrow \Diamond[\gamma]$	*Äquivalenz*
(6)	$[\phi \wedge\ \mathsf{P}_1];[\psi \wedge \mathsf{P}_2]$	$\Rightarrow \Diamond[\gamma]$	*temp. Folgerung*
(7)	$[\phi \wedge\ \mathsf{P}_1];[\psi \wedge \mathsf{P}_2]$	$\Rightarrow \Diamond[\psi \wedge \Diamond\gamma]$	*Äquivalenz*
(8)	$[\phi \wedge\ \mathsf{P}_1];[\psi \wedge \mathsf{P}_2]$	$\Rightarrow [T;\psi \wedge \bigcirc F];[T;\gamma]$	*;-Komposition*
(9)	$\phi \wedge\ \mathsf{P}_1$	$\Rightarrow T;\psi \wedge \bigcirc F$	*Äquivalenz*
(10)	$\psi \wedge \mathsf{P}_2$	$\Rightarrow T;\gamma$	*Äquivalenz*
(11)	$\phi \wedge\ \mathsf{P}_1$	$\Rightarrow \Diamond[\psi \wedge \bigcirc F]$	
(12)	$\psi \wedge \mathsf{P}_2$	$\Rightarrow \Diamond\gamma$	

Es wird vorausgesetzt, daß ϕ und ψ jeweils modalfreie Formeln sind. Sequenz (3) ist hier zunächst ein Lemma, das noch zu beweisen ist.

Regel 1 kann auf verschiedene Weise konkretisiert werden. Für den Umgang mit konjunktiven Zielen kann man die Regel **c_goal split** bilden.

Regel 2 (c_goal split)

$$\frac{\phi,\ \mathsf{P}_1 \Rightarrow \Diamond[\bigcirc F \wedge \boxed{\gamma_1 \wedge \mathsf{pre}}] \qquad \boxed{\gamma_1 \wedge \mathsf{pre}},\ \mathsf{P}_2 \Rightarrow \Diamond[\bigwedge_{i=1}^{n} \gamma_i]}{\phi,\ \mathsf{P}_1;\mathsf{P}_2 \Rightarrow \Diamond[\bigwedge_{i=1}^{n} \gamma_i]}$$

ϕ und γ_i sind modalfreie Formeln, P_1 und P_2 sind Planmetavariablen bzw. Formeln aus LLP,* pre *ist eine Metavariable für eine modalfreie Nichtplanformel.*

□

Diese Regel trägt nun auch der Berücksichtigung von Rahmenaxiomen, deren konkrete Form zur Zeit der Anwendung der Regel noch nicht bekannt ist, Rechnung (vgl. dazu auch die Seiten 120–122). Sie beschreibt zunächst nur, daß die Verbindungsstelle zwischen den beiden angenommenen Teilplänen P_1 und P_2 neben der Formel γ_1 auch noch durch eine bisher nicht bekannte Formel pre charakterisiert ist. Eine Instantiierung dieser Formel wird dann im weiterführenden Beweis durch die Verwendung von Rahmenaxiomen ermöglicht. Die allgemeine Form der Regel, läßt sowohl ihre Verwendung bei einer progressiven als auch bei einer regressiven Planungsstrategie zu, d.h. bei einer progressiven Strategie wird pre stärkste Nachbedingungen von ϕ bzgl. des Planes P_1 enthalten; hingegen impliziert ein regressives Vorgehen die Instantiierung von pre mit den schwächsten Vorbedingungen von $\bigwedge_{i=1}^{n} \gamma_i$ bzgl. des Planes P_2.

Eine unmittelbare Variante von Regel 2 kann durch die zusätzliche Berücksichtigung einer Terminierungsbedingung gebildet werden.

Regel 3 (c_goal split pc)

$$\frac{\phi,\ \mathsf{P}_1 \Rightarrow \Diamond[\bigcirc F \wedge \gamma_1 \wedge \mathsf{pre}] \qquad \gamma_1 \wedge \mathsf{pre},\ \mathsf{P}_2 \Rightarrow \Diamond[\bigwedge_{i=1}^{n} \gamma_i \ \wedge \bigcirc F]}{\phi,\ \mathsf{P}_1;\mathsf{P}_2 \Rightarrow \Diamond[\bigwedge_{i=1}^{n} \gamma_i \ \boxed{\wedge \bigcirc F}\ ]}$$

pre *ist eine Metavariable für eine modalfreie Nichtplanformel.* □

Man formuliert damit eine *partielle Korrektheitsaussage* ([Manna & Pnueli 81]) und verlangt, daß der Plan, der letztendlich als Instantiierung für P_2 gefunden wird, auch terminiert und in seinem Endzustand das spezifizierte Ziel erreicht.
Eine Verallgemeinerung von Regel 2 stellt die folgende Regel dar:

Regel 4 (goal split)

$$\frac{\phi,\ \mathsf{P}_1 \Rightarrow \Diamond[\bigcirc F \wedge \gamma_1 \wedge \mathsf{pre}] \qquad \gamma_1 \wedge \mathsf{pre},\ \mathsf{P}_2 \Rightarrow \gamma_2}{\phi,\ \mathsf{P}_1;\mathsf{P}_2 \Rightarrow \Diamond[\gamma_1 \wedge \gamma_2]}$$

γ_1 *ist eine modalfreie Formel,* γ_2 *eine LLP-Formel,* pre *eine Metavariable für eine modalfreie Nichtplanformel.*

□

Als Spezialisierung von Regel 4 kann man Regel 5 für den Umgang mit Planspezifikationen definieren, die in ihrer Zielspezifikation neben einer Aussage über einen Endzustand auch Aussagen über Zwischenzustände machen[44].

Regel 5 (liveness split)

$$\frac{\phi,\ \mathsf{P}_1 \Rightarrow \Diamond[\bigcirc F \wedge \gamma_1 \wedge \mathsf{pre}] \qquad \gamma_1 \wedge \mathsf{pre},\ \mathsf{P}_2 \Rightarrow \Diamond\gamma_2}{\phi,\ \mathsf{P}_1;\mathsf{P}_2 \Rightarrow \Diamond[\gamma_1 \wedge \Diamond\gamma_2]}$$

γ_1 *ist eine modalfreie Formel und* pre *eine Metavariable für eine modalfreie Nichtplanformel.*

□

Der Unterschied zu Regel 1 liegt im wesentlichen nur darin, daß, nachdem γ_1 einmal erreicht ist, keine weitere Forderung nach seinem Erreichen mehr gestellt wird. Regel 1 verlangt die Erreichung nochmals durch den Teilplan P_2. Eine Ableitung durch Regel 1 ergibt sich gemäß:

[44] Man spricht dabei oft auch von *liveness properties.*

			Angewandte Regeln
(1)	$pre,\ \mathsf{P}_1;\mathsf{P}_2$	$\Rightarrow \Diamond[\gamma_1 \wedge \Diamond\gamma_2]$	*goal refine*
(2)	$pre,\ \mathsf{P}_1$	$\Rightarrow \Diamond[\bigcirc F \wedge \gamma_1 \wedge \mathsf{pre}']$	
(3)	$\gamma_1 \wedge \mathsf{pre}',\ \mathsf{P}_2$	$\Rightarrow \Diamond[\gamma_1 \wedge \Diamond\gamma_2]$	$\Diamond$*-rechts*
(4)	$\gamma_1 \wedge \mathsf{pre}',\ \mathsf{P}_2$	$\Rightarrow \gamma_1 \wedge \Diamond\gamma_2$	$\wedge$*-rechts*
(5)	$\gamma_1 \wedge \mathsf{pre}',\ \mathsf{P}_2$	$\Rightarrow \gamma_1$	$\wedge$*-links*
(6)	$\gamma_1,\ \mathsf{pre}',\ \mathsf{P}_2$	$\Rightarrow \gamma_1$	*Axiom*
(7)	$\gamma_1 \wedge \mathsf{pre}',\ \mathsf{P}_2$	$\Rightarrow \Diamond\gamma_2$	

Ist die Zielbeschreibung einer Planspezifikation in der Form $\Diamond\gamma_1 \wedge \Diamond\gamma_2$ gegeben, also zeitlich ungeordnet, so kann sie aufgrund der in LLP geltenden Theoreme

$$\Diamond[\gamma_1 \wedge \Diamond\gamma_2] \rightarrow \Diamond\gamma_1 \wedge \Diamond\gamma_2$$
$$\Diamond[\gamma_2 \wedge \Diamond\gamma_1] \rightarrow \Diamond\gamma_1 \wedge \Diamond\gamma_2$$
$$\Diamond[\gamma_1 \wedge \gamma_2] \rightarrow \Diamond\gamma_1 \wedge \Diamond\gamma_2$$

z.B. eingeschränkt werden, um dann die Regeln 2 bzw. 5 anwenden zu können. Eine weitere Möglichkeit, mit einer derartigen Zielspezifikation umzugehen, besteht darin, zunächst eines der Teilziele (z.B. γ_1) zu fokussieren, und während der Verfeinerung dieses Zieles, das verbleibende Ziel, γ_2, zu erreichen, sobald sich ein *günstiger* Zeitpunkt ergibt. Die folgende Regel spezifiziert dieses opportunistische Vorgehen:

Regel 6 (o_goal split)

$$\frac{\phi,\ \mathsf{P}_1 \Rightarrow \Diamond[\bigcirc F \wedge \psi] \wedge \Diamond\gamma_2 \qquad \psi,\ \mathsf{P}_2 \Rightarrow \Diamond\gamma_1}{\phi,\ \mathsf{P}_1;\mathsf{P}_2 \Rightarrow \Diamond\gamma_1 \wedge \Diamond\gamma_2}$$

□

Hier wird die Entscheidung über eine zeitliche Plazierung von γ_2 herausgezögert, d.h. aufgrund der Semantik des $\Diamond$-Operators entspricht die Formel $\Diamond[\bigcirc F \wedge \psi] \wedge \Diamond\gamma_2$ der Disjunktion

$$\begin{array}{l}\Diamond[\bigcirc F \wedge \psi] \wedge \gamma_2 \vee \\ \Diamond[\bigcirc F \wedge \psi] \wedge \bigcirc\gamma_2 \vee \\ \Diamond[\bigcirc F \wedge \psi] \wedge \bigcirc\bigcirc\gamma_2 \vee \\ \Diamond[\bigcirc F \wedge \psi] \wedge \bigcirc\bigcirc\bigcirc\gamma_2 \vee \\ \vdots \\ \Diamond[\bigcirc F \wedge \psi \wedge \gamma_2]\end{array}$$

und entspricht aus Sicht des Planes P_1 der nichtdeterministischen Entscheidung in irgendeinem der von ihm hervorgerufenen Zustandsübergänge das Ziel γ_2 zu erreichen. Die

Auswahl, wann dies geschehen soll, obliegt dann einer entsprechenden Heuristik in der jeweiligen Planungsstrategie.

Das vorrangige Ziel beim Beweis einer Planspezifikation zur Konstruktion eines linearen Planes ist nun, durch eine Reihe von Zielaufspaltungen eine Kette von Teilzielen zwischen Anfangs- und Zielzustand aufzubauen – dieser Punkt wird in der dritten der oben aufgestellten Fragen problematisiert. Die Teilziele erhält man durch die Anwendung der oben genannten verschiedenen Zielaufspaltungsregeln. Die entstandenen Teilprobleme können dann idealerweise mit Hilfe von Aktionsaxiominstanzen und assoziierten Rahmenaxiomen gelöst werden. Im folgenden zeigen wir nun, wie eine progressive und eine regressive Planungsstrategie die notwendige Kette von Teilproblemen aufstellt.

Generierung unter Einsatz von Progression Eine progressive Planungsstrategie benötigt zur Erreichung eines effektiven Verhaltens eine zielgerichtete Steuerung. Dies ist notwendig, da das progressive Vorgehen Aktionen im spezifizierten Anfangszustand ausführt und nur solche Aktionen berücksichtigen sollte, die einen Beitrag zur Erreichung des spezifizierten Zieles leisten.[45]
Bevor die beweistechnische Durchführung der Strategie diskutiert wird, folgt zunächst eine prozedurale Beschreibung der Strategie:

```
progression(Init,Goals) <-
   if empty(Goals)
   then return
   else G:= member(first(Goals)) und nicht_erfuellt(Init,G) % Ruecksetzpunkt
        Tmp:= erreichbar(G)
        Aktlist:= ordne(Tmp,Init)
        A:= member(Aktlist) % Ruecksetzpunkt
        if ausfuehrbar(A,Init)
        then Init':= folgezustand(Init,A)
             Goals':= noch_offene_Zustaende(Goals,Init')
             progression(Init',Goals')
        else Goals':= append(Vorbedingungen(A),Goals)
             progression(Init,Goals')
```

Die Prozedur `progression` initialisiert mit dem aktuellen Anfangszustand und einer Liste von Listen mit jeweils offenen Zielen[46] hat, wenn sie korrekt endet, zwischendurch einen sequentiellen Plan generiert. Die Funktion `erreichbar` liefert zu einem gegebenen Ziel Aktionsinstanzen zurück, die das Ziel unmittelbar erreichen können. Die Funktion `ordne` ordnet die Aktionen gemäß: *bevorzuge die Aktion a gegenüber b, wenn a bzgl. dem aktuellen Ausgangszustand weniger nichterfüllte Vorbedingungen enthält als b.* `folgezustand` berechnet die stärksten Nachbedingungen der Aktion `A` bzgl. des Zustandes `Init`. Die Funktion `noch_offene_Zustaende` entfernt den maximalen Präfix von `Goals`, der nur Ziellisten enthält, bei denen alle individuellen Ziele bereits aus `Init'` folgen.
Die folgende graphische Skizzierung des Vorgehens verschanschaulicht die Progressionsstrategie nochmals.

[45] Unterläßt man die Fokussierung, führt dies im allgemeinen zu einem inakzeptablen Suchaufwand.
[46] Jede Liste repräsentiert einen Zustand.

Gehen wir dazu von einer Planspezifikation aus, in der ein Ziel bestehend aus einer Konjunktion von Literalen spezifiziert ist. Die zielgerichtete Progressionsstrategie versucht in einer ersten Phase ausgehend vom Zielzustand, eine Verbindung zum Anfangszustand aufzubauen. Abbildung 4.11 zeigt den ersten Schritt dieses Vorgehens. Literale oder all-

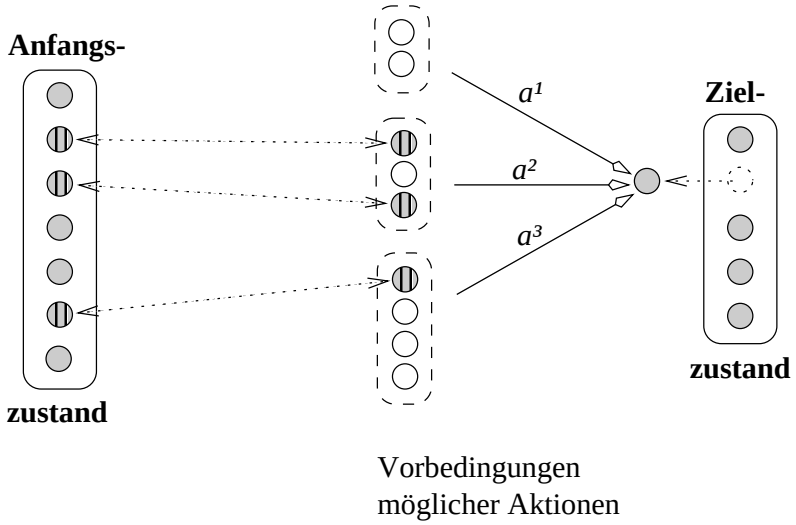


Abbildung 4.11: Skizze einer Progressionsstrategie: Schritt 1

gemein Bedingungen, die in Ziel- und Anfangszustand gelten, sind hier durch Kreise symbolisiert. Die Strategie wählt zunächst ein Literal des Zielzustandes aus und bestimmt alle Aktionen, die dieses Ziel potentiell erreichen könnten. In der Abbildung seien dies die Aktionen a^1, a^2 und a^3. Um diese Aktionen ausführen zu können, sind jeweils Mengen von Vorbedingungen notwendig, dargestellt durch gestrichelte Ovale. Unter den einzelnen Vorbedingungen kann es welche geben, die auch bereits im Anfangszustand gelten. Wenn es nun eine Aktion gibt, deren Vorbedingungen alle schon im Anfangszustand gelten, dann könnte diese Aktion ausgeführt werden, und das selektierte Zielliteral wäre erreicht. Die Abbildung zeigt den Fall, daß zu jeder der verwendbaren Aktionen nichtgeltende Vorbedingungen, d.h. neue, momentan offene Teilziele existieren, dargestellt durch nichtgefüllte Kreise; die schraffierten Kreise stellen Bedingungen dar, die bereits im aktuellen Anfangszustand gelten. Eine Heuristik, realisiert als spezielle Entscheidungsfunktionsinstanz (vgl. Abschnitt 3.1.4) entscheidet nun, daß die Aktion präferiert wird, bei der die wenigsten offenen Teilziele bestehen; im Beispiel ist dies die Aktion a^2. Der geschilderte Prozeß setzt sich nun rekursiv weiter fort, bis eine vollständige Verbindung zum Anfangszustand hergestellt ist; Abbildung 4.12 zeigt diesen Status.
Jetzt kann ein Progressionsschritt bzgl. der Aktion *akt* durchgeführt werden. Man erhält dann eine Fortschreibung des Anfangszustandes, die die stärksten Nachbedingungen bzgl. *akt* enthält (vgl. dazu die Abbildung 4.13). Darin ist jetzt zumindest das vor Ausführung von *akt* noch offene Ziel enthalten, das ursprünglich den Einsatz von *akt* angeregt hat.
Die Situation der verbleibenden Zwischenziele ist vergleichbar mit der Situation zu Beginn der Progressionsstrategie. Es gibt einen Zielzustand, Z', und einen Anfangszustand, A', d.h. die Progressionsstrategie kann nun rekursiv aufgerufen werden. Das Erreichen offener

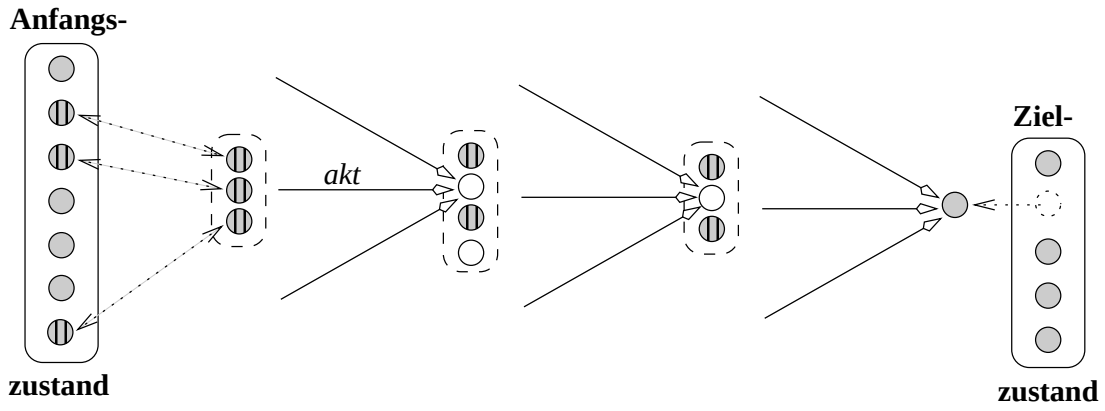


Abbildung 4.12: Skizze einer Progressionsstrategie: Schritt 2

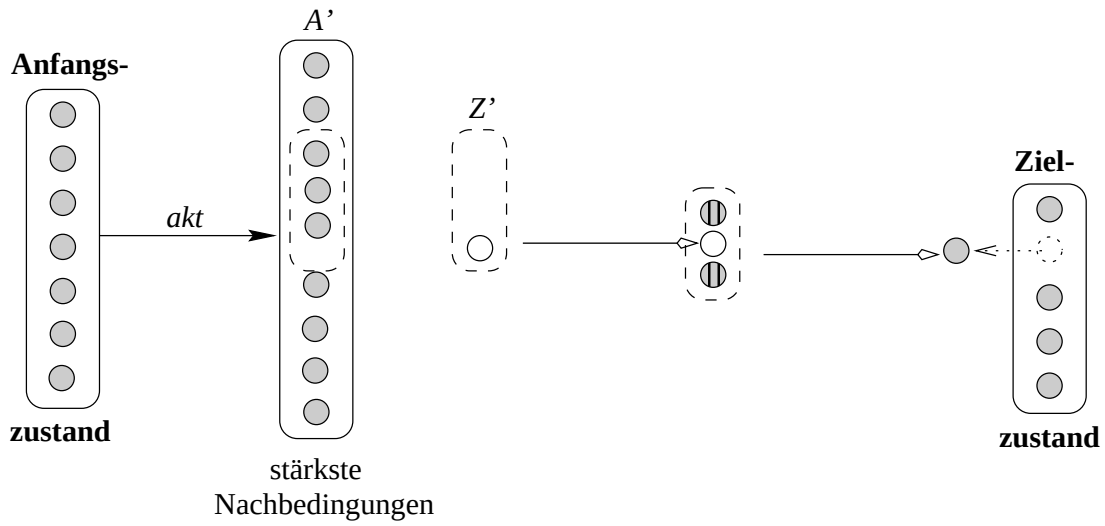


Abbildung 4.13: Skizze einer Progressionsstrategie: Progressionsschritt

Zwischenziele schreitet zunächst solange fort, bis das ursprünglich ausgewählte Teilziel des Zielzustandes (vgl. Abbildung 4.11) in den stärksten Nachbedingungen einer angewandten Aktion gilt. Wenn dann noch offene Teilziele im spezifizierten Zielzustand vorhanden sind, so beginnt der gesamte Progressionsprozeß bzgl. dieser Teilziele von neuem.

Betrachten wir nun die beweistechnische Durchführung der Progressionsstrategie. Die initiale Sequenz, bei der die Beweisführung aufsetzt, sei gegeben als

$$pre,\ \mathsf{P} \Rightarrow \Diamond\ [\bigwedge_{i=1}^{n} \gamma_i \wedge \rho]$$

pre eine modalfreie LLP-Formel, P eine Planmetavariable,
γ_i ein LLP-Literal, ρ eine LLP-Formel.

Abbildung 4.14 charakterisiert die globale Struktur des Beweises. Die Struktur der Zielspezifikation führt zunächst zur Einführung einer sequentiellen Struktur in die Planmetavariable P; es resultiert Sequenz (2). Die Anwendung der Regel *goal split* vollzieht dann die sequentielle Zielstrukturierung. Aufgrund des progressiven Vorgehens wird zunächst nach dem initialen Teilplan gesucht (Fokussierung von Sequenz (4), Aufschub von Sequenz (3)) und erst nach dessen Existenz wird der terminale Teilplan bestimmt.
Die Beweisstrategie selektiert aus komplexen (konjunktiven) Zielen zunächst ein atomares Ziel, das erreicht werden soll (Sequenz (6)). Die restlichen Ziele sind in Sequenz (7) enthalten. Die Verwendung der Regel *c-goal split pc* führt für sich alleine genommen zu keiner Zielreduzierung; es wird jedoch jeweils ein neues Unterziel gebildet, dessen Lösung zu der Lösung des Gesamtproblems beiträgt. Das "Zwischenergebnis" geht über die Metavariable pre_4, die mit den stärksten Nachbedingungen von Plan P_3 instantiiert werden wird, in die Problemlösung ein. Der Beweis schreitet nur dann mit Sequenz (7) fort, wenn Sequenz (6) bewiesen ist.
Die Anwendung von $\wedge$-kommutativ-Regeln auf Sequenz (7) führt zu einer Vertauschung von Konjunktionsgliedern. Damit erreicht man, daß durch eine erneute Anwendung der Regel *c-goal split pc* in Sequenz (10) ein anderes Teilziel abgespalten wird. Zuvor wird jedoch eine weitere sequentielle Planstruktur eingeführt (Sequenz (8)) und die stärksten Nachbedingungen sp_3 des Teilplanes P_3 werden etabliert. Dieses Vorgehen kann nun solange wiederholt werden, bis man mit der Einführung des letzten Teilplanes stärkste Nachbedingungen erreicht, die alle Ziele γ_i schon enthalten. Dies muß dann durch den Beweis der Sequenzen $(k+7)$–$(k+m)$ noch bestätigt werden. Dazu müssen die Metavariablen $\mathsf{pre}_4,\ldots,\mathsf{pre}_x$ als Akkumulatoren der Teilziele gewirkt haben.
Sequenz (6) spezifiziert nun ein Basisproblem, dessen Lösung in den Abbildungen 4.11–4.13 bereits skizziert wurde. Abbildung 4.15 enthält zunächst den zu führenden Beweis für den Fall, daß das Teilziel γ_1 mit Hilfe *einer* Aktion erreicht werden kann.
Beim Übergang von Sequenz (6a) zu Sequenz (6b) wird die mittelbare Erreichbarkeit eines Zieles auf die unmittelbare Erreichbarkeit reduziert, was dann schließlich die Anwendung eines Aktionsaxioms möglich macht. Betrachtet man Sequenz (6g) so ist offensichtlich, daß der Beweis an dieser Stelle, wie in Abbildung 4.5 gezeigt, geschlossen werden kann. Aber Sequenz (6e) ist eine Zusicherung über die Länge von Plan P_3 und verlangt, um bewiesen werden zu können, eine konkrete Annahme über diese Länge.[47] Für Basisaktionen ist die Länge gemäß der Axiomatisierung auf einen Zustandsübergang beschränkt,

[47] Genaugenommen interessiert uns die Länge des Intervalles, in dem Plan P_3 gilt.

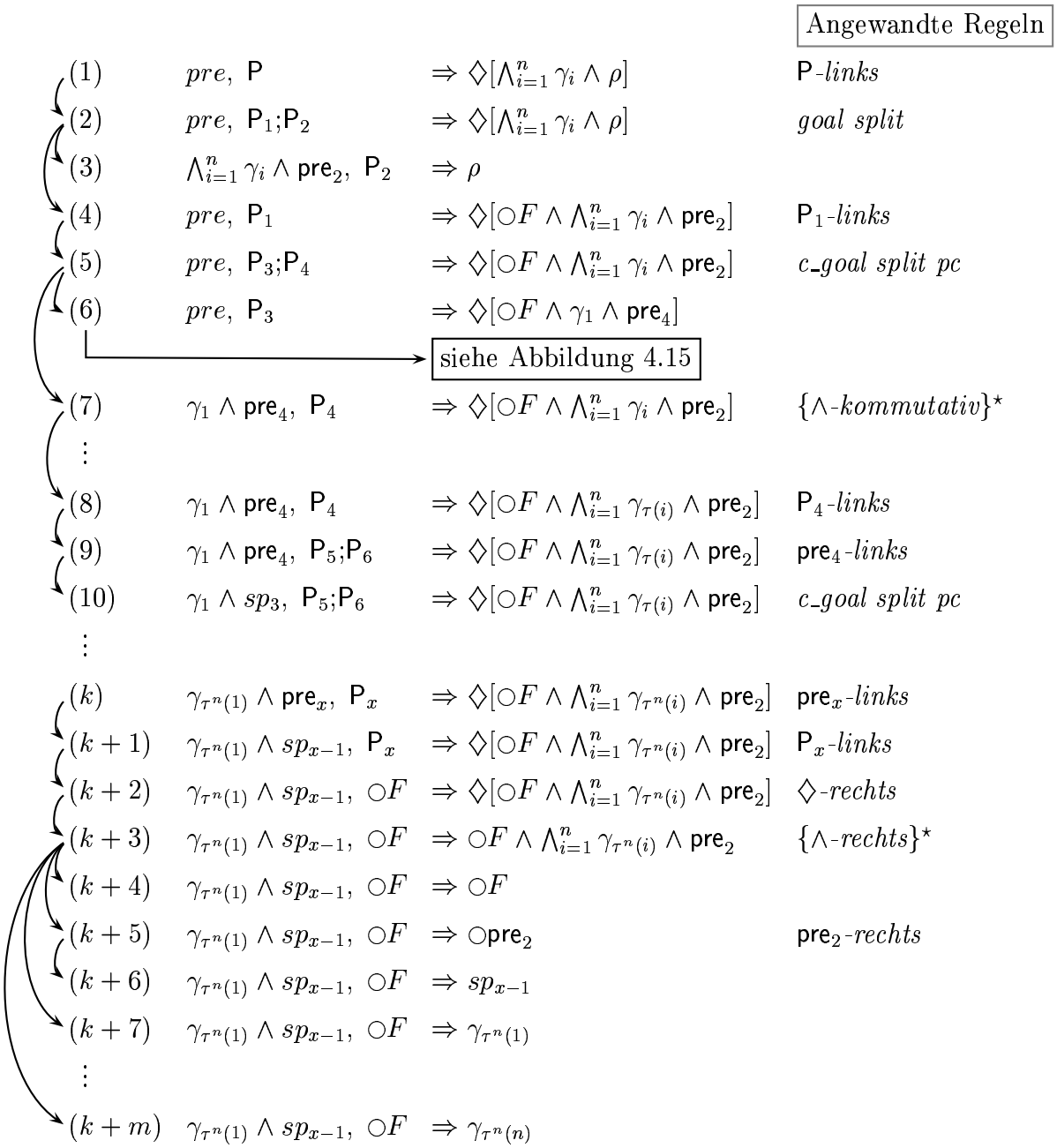


Abbildung 4.14: Durchführung einer Progressionsstrategie: Teil 1

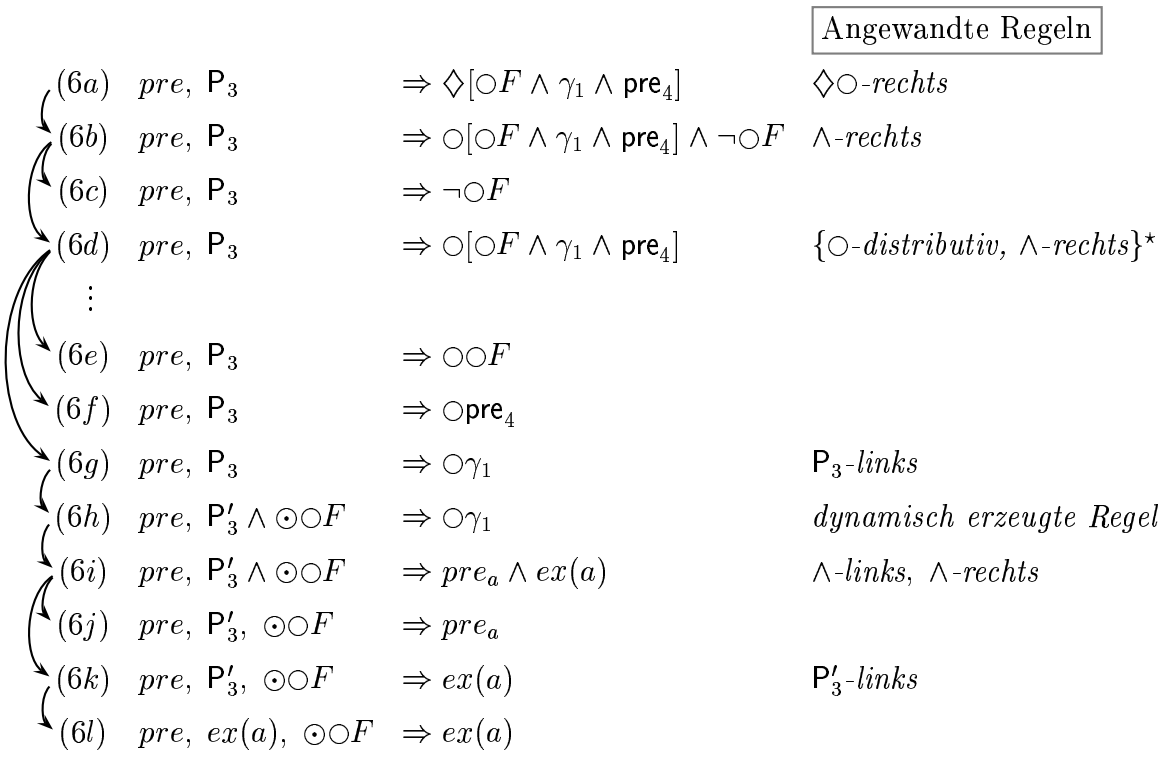


Abbildung 4.15: Durchführung einer Progressionsstrategie: Teil 2a

d.h. eine sichere Aussage über das Eintreten des Effektes einer Aktion kann nur im dem Ausführungszustand unmittelbar folgenden Zustand getroffen werden; die Formel $\odot\bigcirc F$ beschränkt die Länge eines Intervalls auf 1. Mit Sequenz (6h) ist diese Beschränkung eingeführt. Der Beweis dieser Sequenz erfolgt analog zu der Vorgehensweise in Abbildung 4.5. Die Sequenz (6f) spezifiziert, nachdem (6h) bewiesen ist, ein Rahmenproblem; man möchte wissen, welche Eigenschaften nach Ausführung der Aktion a gelten. Gemäß dem in Abschnitt 4.2.5 beschriebenen Vorgehen kann man eine Axiomeninstanz der Aktion a bilden, die die stärksten Nachbedingungen von pre bzgl. a definiert, ein a-Rahmenaxiom bzgl. pre. Die folgenden Sequenzen (6f)–(6f4) zeigen die Etablierung des Rahmenaxioms. Die Bestimmung der Einführungsregel pre_4-*rechts*, die auf (6f2) angewendet wird, geschieht bezüglich der Definition 4.2.6 unter Verwendung der Funktion $\mathcal{FT}^{-1}$. Sie legt die stärksten Nachbedingungen sp_3 fest. Die Rahmenaxiomregel, die auf (6f3) angewendet wird, ist ebenfalls durch Definition 4.2.6 abgesichert, jetzt aber unter Verwendung der Funktion $\mathcal{FT}$. Ein effektiver Vorschlag für ein zu verwendendes Rahmenaxiom erfolgt also durch das progressive Prinzip, die beweistechnische Absicherung basiert aber auf Regression.

			Angewandte Regeln
$(6f)$	$pre,\ \mathsf{P}_3$	$\Rightarrow \bigcirc\mathsf{pre}_4$	P_3-*links*
$(6f1)$	$pre,\ \mathsf{P}'_3 \wedge \odot\bigcirc F$	$\Rightarrow \bigcirc\mathsf{pre}_4$	P'_3-*links*
$(6f2)$	$pre,\ ex(a) \wedge \odot\bigcirc F$	$\Rightarrow \bigcirc\mathsf{pre}_4$	pre_4-*rechts*
$(6f3)$	$pre,\ ex(a) \wedge \odot\bigcirc F$	$\Rightarrow \bigcirc sp_3$	a-Rahmenaxiom bzgl. pre[48]
$(6f4)$	$pre,\ ex(a) \wedge \odot\bigcirc F$	$\Rightarrow pre_a \wedge ex(a)$	

Die Sequenzen (6c) und (6e) aus Abbildung 4.15 sind Zusicherungen über die Länge von Teilplan P_3. Es wird nun eine Ableitung dieser Beweisprobleme gezeigt:

			Angewandte Regeln
$(6c)$	$pre,\ \mathsf{P}_3$	$\Rightarrow \neg\bigcirc F$	P_3-*links,* P'_3-*links*
$(6c1)$	$pre,\ ex(a) \wedge \odot\bigcirc F$	$\Rightarrow \neg\bigcirc F$	$\wedge$-*links*
$(6c2)$	$pre,\ ex(a),\ \odot\bigcirc F$	$\Rightarrow \neg\bigcirc F$	$\odot$-*Definition*
$(6c3)$	$pre,\ ex(a),\ \neg\bigcirc\neg\bigcirc F$	$\Rightarrow \neg\bigcirc F$	$\neg$-*rechts,* $\neg$-*links*
$(6c4)$	$pre,\ ex(a),\ \bigcirc F$	$\Rightarrow \bigcirc\neg\bigcirc F$	$\bigcirc$-*rechts*
$(6c5)$	F	$\Rightarrow \neg\bigcirc F$	*Axiom*

[48] Korrekterweise die dazu korrespondierende Sequenzenregel.

Für die Ableitung von (6e) gilt:

			Angewandte Regeln
(6e)	$pre,\ \mathsf{P}_3$	$\Rightarrow \bigcirc\bigcirc F$	P_3-*links,* P'_3-*links*
(6e1)	$pre,\ ex(a) \wedge \odot\bigcirc F$	$\Rightarrow \bigcirc\bigcirc F$	$\wedge$-*links,* $\odot$-*Definition*
(6e2)	$pre,\ ex(a),\ \neg\bigcirc\neg\bigcirc F$	$\Rightarrow \bigcirc\bigcirc F$	$\neg$-*links*
(6e3)	$pre,\ ex(a)$	$\Rightarrow \bigcirc\bigcirc F,\ \bigcirc\neg\bigcirc F$	$\bigcirc$-*rechts*
(6e4)		$\Rightarrow \bigcirc F,\ \neg\bigcirc F$	$\neg$-*rechts*
(6e5)	$\bigcirc F$	$\Rightarrow \bigcirc F$	*Axiom*

Sequenz (6) aus Abbildung 4.14, d.h. das Beweisproblem zur Erreichung des Teilzieles γ_1 kann im allgemeinen nur unter Zuhilfenahme einer *Kette* von Aktionen abgeleitet werden. Dies bedeutet, daß es nicht möglich war, den Beweis beginnend mit Sequenz (6g) (vgl. Abbildung 4.15) zu schließen, da jede in (6h) verwendbare Aktionsaxiomeninstanz bewirkt, daß ein Beweis von (6j) scheitert, also die notwendigen Vorbedingungen der Aktion können nicht aus den Anfangsbedingungen hergeleitet werden. Wie in Abbildung 4.12 bereits skizziert, kann man nichtgeltende Vorbedingungen auch als Zwischenziel auffassen, die, wenn man sie erreichen würde, die Anwendung einer Aktion zur Erreichung von γ_1 gestatten. Abbildung 4.16 zeigt die dazu notwendige Beweisführung zur Einführung eines Zwischenzieles.

			Angewandte Regeln
(6a)	$pre,\ \mathsf{P}_3$	$\Rightarrow \Diamond[\bigcirc F \wedge \gamma_1 \wedge \mathsf{pre}_4]$	P_3-*links*
(6b)	$pre,\ \mathsf{P}_5;\mathsf{P}_6$	$\Rightarrow \Diamond[\bigcirc F \wedge \gamma_1 \wedge \mathsf{pre}_4]$	*goal refine*
(6c)	$pre,\ \mathsf{P}_5$	$\Rightarrow \Diamond[\bigcirc F \wedge \zeta \wedge \mathsf{pre}_6]$	
(6d)	$\zeta \wedge \mathsf{pre}_6,\ \mathsf{P}_6$	$\Rightarrow \Diamond[\bigcirc F \wedge \gamma_1 \wedge \mathsf{pre}_4]$	

Abbildung 4.16: Durchführung einer Progressionsstrategie: Teil 2b

Das Zwischenziel wird als eine modalfreie Formel ζ angenommen und von der Regel *goal refine* entsprechend eingeführt. Der Beweis schreitet dann zunächst mit der Ableitung von Sequenz (6c) fort und zwar analog zu dem Vorgehen in Abbildung 4.14 beginnend mit Sequenz (4) oder Sequenz (6), je nachdem ob ζ aus einer Konjunktion oder einem Literal besteht. Bei Erfolg wird mit Sequenz (6d) so fortgefahren wie mit Sequenz (6).

Die letzte verbleibende Möglichkeit, das Beweisproblem aus Sequenz (6) zu bearbeiten, besteht darin, den *leeren* Plan für P_3 einzuführen; Abbildung 4.17 zeigt die entsprechende Beweisführung.

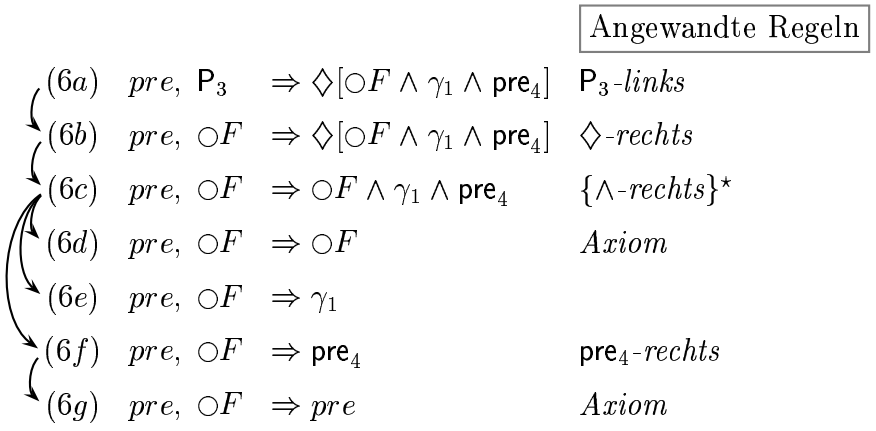


Abbildung 4.17: Durchführung einer Progressionsstrategie: Teil 2c

Das Ziel γ_1 muß in diesem Fall aus den Vorbedingungen *pre* unmittelbar ableitbar sein; d.h. der Beweis von Sequenz (6e) muß abgeschlossen werden können.

Ein sequentieller Plan, der sich nach erfolgreichem Beweis des gesamten Planungsproblems (siehe Sequenz (1) in Abbildung 4.14) ergibt, hat eine Struktur, die durch folgende Grammatik charakterisiert werden kann:

$$
\begin{array}{rcl}
\texttt{Plan} & ::= & \texttt{S\_Plan ; } \bigcirc F \ \texttt{|} \\
 & & \bigcirc F \\
\texttt{S\_Plan} & ::= & \texttt{S\_Plan ; S\_Plan |} \\
 & & ex(Aktionsterm) \wedge \odot\bigcirc F \ \texttt{|} \\
 & & \bigcirc F
\end{array}
$$

Eine Formel $\bigcirc F$ entsteht immer dann, wenn eine Aufspaltung der aktuellen Planmetavariable vorgenommen wurde, die sich im Nachhinein als nicht notwendig erwiesen hat.
Die erfolgreiche Durchführung der progressiven Planungsstrategie führt aus der Sicht der Modellverfeinerung zusammenfassend zu der folgenden Intervallstrukturierung:

$$
\begin{array}{rcccc}
 & & & & \langle\sigma_z\rangle \\
 & & & \langle\sigma_{x-1}\sigma_x\rangle & \langle\sigma_x \ldots \sigma_z\rangle \\
 & & & & \vdots \\
 & \langle\sigma_1\sigma_2\rangle & \langle\sigma_2 & \ldots & \sigma_z\rangle \\
\langle\sigma_0\sigma_1\rangle & \langle\sigma_1 & \ldots & & \sigma_z\rangle \\
\langle\sigma_0 & \ldots & & & \sigma_z\rangle
\end{array}
$$

Das Intervall $\langle\sigma_0 \ldots \sigma_z\rangle$ ist dabei mit der Planmetavariable P assoziiert; $\langle\sigma_0\sigma_1\rangle$ entspricht P_1 und letztendlich korrespondiert $\langle\sigma_x \ldots \sigma_z\rangle$ mit P_x. Die Instantiierung von P_x mit dem leeren Plan, also ein Längeneinführungsverfeinerung, das die Intervallänge auf 0 festlegt, unifiziert x mit z und schließt die Kette von Verfeinerungsschritten ab. Da leere Teilpläne

im Prinzip an beliebigen Stellen im Plan auftreten können, erscheint dies dann durch eine redundante Verfeinerung $\langle\sigma_i\rangle$, falls ein leerer Plan etwa mit dem Zustand σ_i korrespondiert.

Eine effizientere Beweisstrategie Betrachten wir zunächst noch einmal das Vorgehen der oben beschriebenen Strategie für den Fall, daß sich die Notwendigkeit ergibt, eine Kette von Zwischenzielen zu verwenden, um ein ausgewähltes Teilziel zu erreichen. Entsprechend dem Vorgehen in Abbildung 4.16 ergeben sich z.B. für eine Kette von 3 Zwischenzielen die in Abbildung 4.18 gezeigten Beweisprobleme.

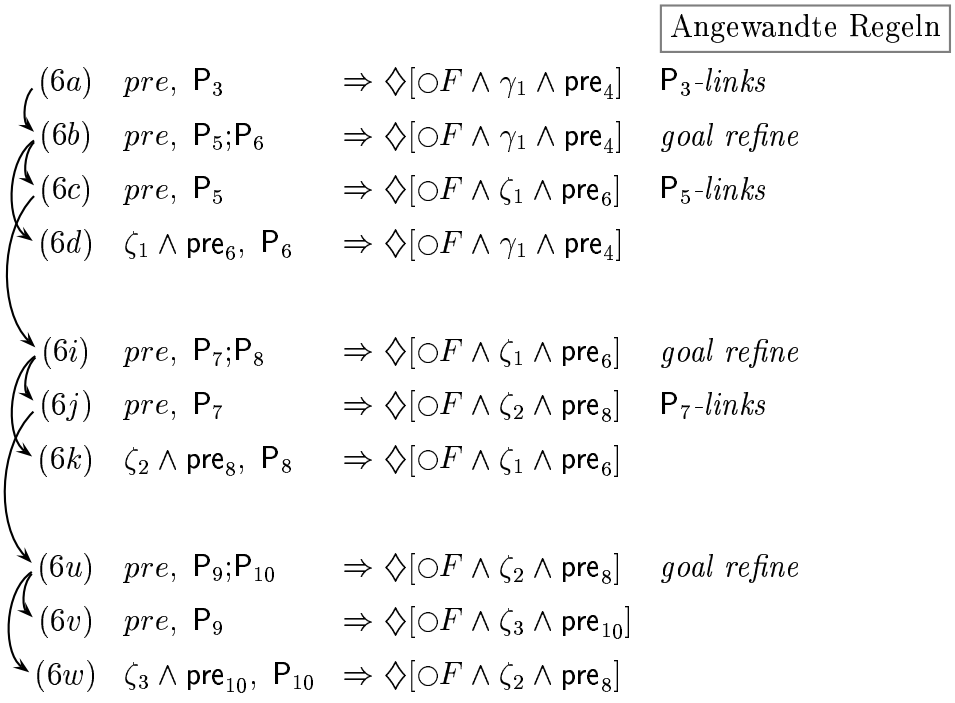


Abbildung 4.18: Entstehung notwendiger Zwischenziele

Das progressive Vorgehen verlangt, daß die Sequenz (6v) als erste zu beweisen ist. Nehmen wir oBdA. an, daß dies gelingt – das Zwischenziel ζ_3 ist also erreicht –, so sind als nächstes die Zwischenziele ζ_2 und ζ_1 und zuletzt das spezifizierte Ziel γ_1 zu erreichen. Zu beachten ist, daß aufgrund der aufgebauten Beweisstruktur die Erreichung der verbleibenden Zwischenziele *notwendig* ist. Aus der Sicht der Problemstellung in Sequenz (6a) ist das Erreichen dieser Ziele aber *nicht* notwendig, sondern lediglich eine Hilfestellung um das Ziel γ_1 letztendlich zu erreichen. Mit der Lösung von (6v) und der einhergehenden Instantiierung von pre_{10} mit den stärksten Nachbedingungen von P_9 kann sich z.B. ergeben, daß γ_1 direkt aus diesen Nachbedingungen ableitbar wäre; das Beweisproblem (6a) wäre dann eigentlich zufriedenstellend gelöst und die Hilfestellung durch die Ziele

ζ_2 und ζ_1 überflüssig geworden. Dies könnte sich so nur dann im Beweis zutragen, wenn ζ_2 und ζ_1 ebenfalls direkt aus den Nachbedingungen folgen. Davon kann im allgemeinen aber nicht ausgegangen werden und somit werden dann unnötigerweise Probleme gelöst, deren Lösung schlimmstenfalls dazu führt, das erreichte Ziel γ_1 wieder zu verlieren. Die folgende Prozedur beschreibt die effizientere Strategie:

```
progression(Init,Goals) <-
   if empty(Goals)
   then return
   else G:= member(first(Goals)) und nicht_erfuellt(Init,G) % Ruecksetzpunkt
        Goals':= append(G,Goals) und notwendig(G)
        Tmp:= erreichbar(G)
        Aktlist:= ordne(Tmp,Init)
        A:= member(Aktlist) % Ruecksetzpunkt
        if ausfuehrbar(A,Init)
        then Init':= folgezustand(Init,A)
             Goals'':= notwendige_Zustaende(Goals',Init')
             progression(Init',Goals'')
        else Goals'':= append(Vorbedingungen(A),Goals')
             progression(Init,Goals'')
```

Notwendige Ziele sind hier als solche markiert. Die Funktion `notwendige_Zustaende` kann aus `Goals'` einen Präfix entfernen, der auch offene nicht notwendige Zielzustände enthält, wenn gleichzeitig alle enthaltenen notwendigen Zielzustände nicht mehr offen sind bzgl. `Init'`.
Um nun die notwendige Flexibilität in die Beweisführung zu integrieren, kann die Regel *goal refine* entsprechend angepaßt werden. Es ergeben sich die beiden folgenden Varianten:

Regel 7 (goal refine ing)

$$\frac{\phi,\ \mathsf{P}_1 \Rightarrow \Diamond[\bigcirc F \wedge \zeta \wedge \mathsf{pre}],\ \bigcirc F \wedge \gamma \wedge \mathsf{pre} \qquad \mathsf{pre},\ \mathsf{P}_2 \Rightarrow \Diamond[\gamma \wedge \rho]}{\phi,\ \mathsf{P}_1;\mathsf{P}_2 \Rightarrow \Diamond[\gamma \wedge \rho]}$$

ϕ, γ und ζ sind modalfreie LLP-Formeln, P_1 *und* P_2 *sind Planmetavariablen,* pre *ist eine Metavariable für eine modalfreie Formel, ρ ist eine LLP*-Formel.*[49]

□

Regel 8 (goal refine png)

$$\frac{\phi,\ \mathsf{P}_1 \Rightarrow \Diamond[\bigcirc F \wedge \zeta \wedge \mathsf{pre}_1],\ \bigcirc F \wedge \gamma \wedge \mathsf{pre}_1 \qquad \mathsf{pre}_1,\ \mathsf{P}_2 \Rightarrow \Diamond\rho,\ \bigcirc F \wedge \gamma \wedge \mathsf{pre}}{\phi,\ \mathsf{P}_1;\mathsf{P}_2 \Rightarrow \Diamond\rho,\ \bigcirc F \wedge \gamma \wedge \mathsf{pre}}$$

pre_1 *ist eine Metavariable für eine modalfreie Formel.*

□

[49] Zur Erinnerung: die Abtrennung von Formeln durch Kommata im Sukzedens einer Sequenz ist gleichzusetzen mit ihrer Disjunktion.

Regel 7 abstrahiert zwei Vorgehensweisen, die Anwendung von Regel 1 und die Folge Regel 4 gefolgt von Regel $\Diamond$-*rechts*. Hierdurch hat man die Möglichkeit, die Entscheidung, welche der beiden Varianten im Beweis zum Zuge kommen soll, in die konkrete Beweissituation zu verlagern, in der ausreichende Information vorliegt. Man erspart sich dadurch ein potentielles Zurücksetzen im Beweisprozeß, wenn man sich zu früh für eine der beiden Varianten entschieden hätte und sich diese Entscheidung später als falsch erweisen würde. Die kombinierte Regel führt somit zu einer Effizienzsteigerung des Beweisprozesses. Regel 8 ist die Variante, die nach der erstmaligen Anwendung von Regel 7 anzuwenden ist.
Abbildung 4.19 zeigt, wie unter Verwendung der beiden Regeln die Flexibilität in der Beweisführung bereitsteht, die es erlaubt, die Zwischenziele ζ_1 und ζ_2 nicht erreichen zu müssen.

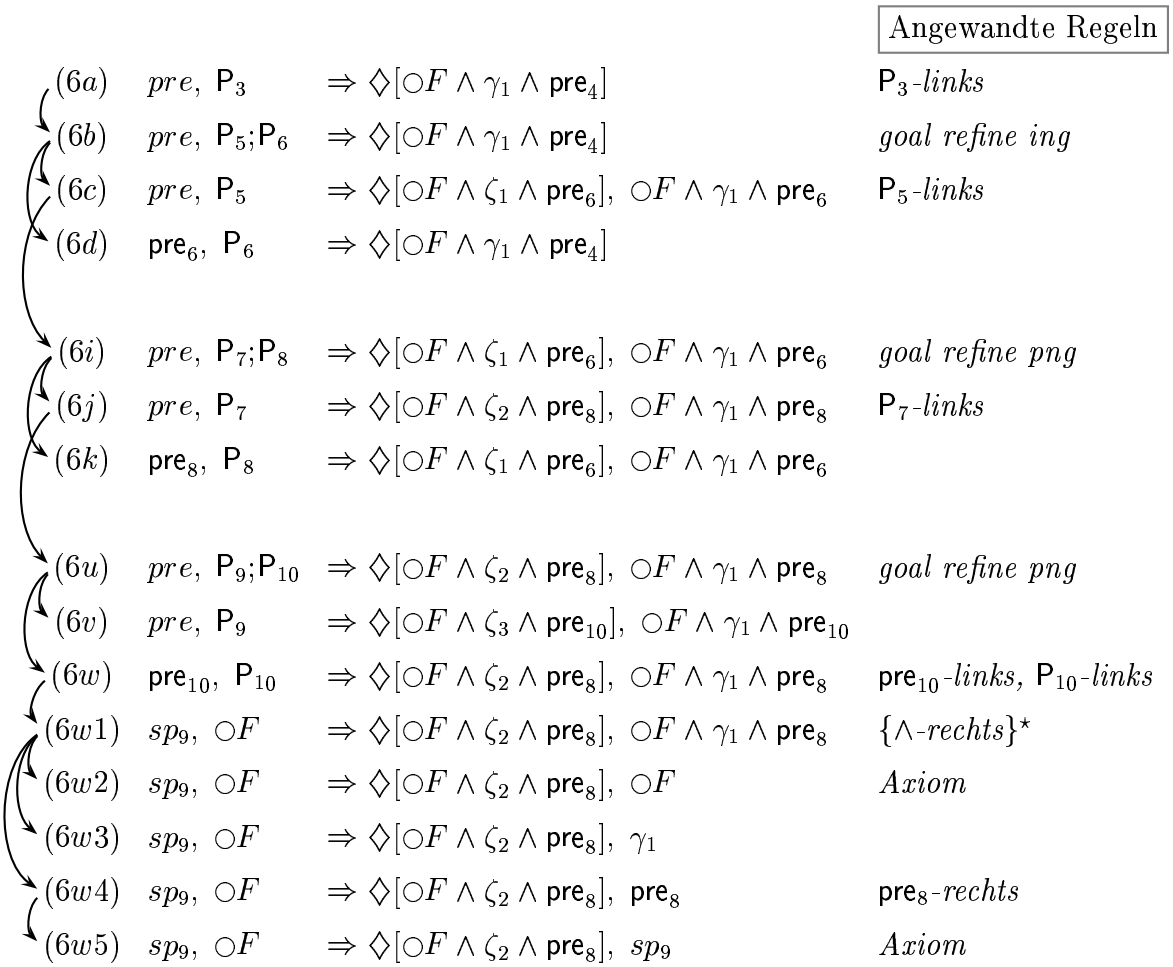


Abbildung 4.19: *Entschärfung* bereits eingeführter Zwischenziele

Nehmen wir also an, daß mit dem Beweis von Sequenz (6v) ein Plan zur Erreichung von Zwischenziel ζ_3 entstanden ist, der gleichzeitig auch schon γ_1 erreicht. Dies hat zur Folge,

daß für die Planmetavariablen P_{10}, P_8 und P_6 jeweils der leere Plan $\bigcirc F$ eingesetzt werden kann, um (6a) zufriedenstellend zu lösen. Für die Sequenz (6w) wird die entsprechende Ableitung gezeigt. Mit Sequenz (6k) wird analog dazu verfahren; zur Ableitung von (6d) muß zusätzlich zunächst die Regel $\Diamond$-*rechts* angewendet werden. Die stärksten Nachbedingungen sp_9 von Plan P_9 werden durch das beschriebene Vorgehen durch die aufgebaute Metavariablenkette bis zu Sequenz (6d) *durchgeschleust*, ohne die Zwischenziele ζ_1 und ζ_2 bearbeiten zu müssen.

Generierung unter Einsatz von Regression Der oben ausführlich beschriebenen, zielgerichteten Progressionsstrategie stellen wir nun eine regressive Beweisstrategie gegenüber. Ein zielgerichtetes Vorgehen ist dieser Strategie bereits per Definition gegeben. Vor der beweistechnischen Durchführung beschreiben wir auch hier zunächst das prozedurale Vorgehen und skizzieren danach graphisch den Kern der Heuristik.

```
regression(Init,Goals) <-
   if erfuellt(Init,Goals)
   then return
   else G:= member(Goals) und nicht_erfuellt(Init,G) % Ruecksetzpunkt
        Tmp:= erreichbar(G)
        Aktlist:= ordne(Tmp,Goals)
        A:= member(Aktlist) % Ruecksetzpunkt
        Goals':= notwendige_Vorbedingung(A,Goals)
        regression(Init,Goals')
```

Die Funktion `ordne` ordnet eine Aktion a vor einer Aktion b an, wenn die Aktion a insgesamt mehr Ziele aus `Goals` erreichen kann als die Aktion b.

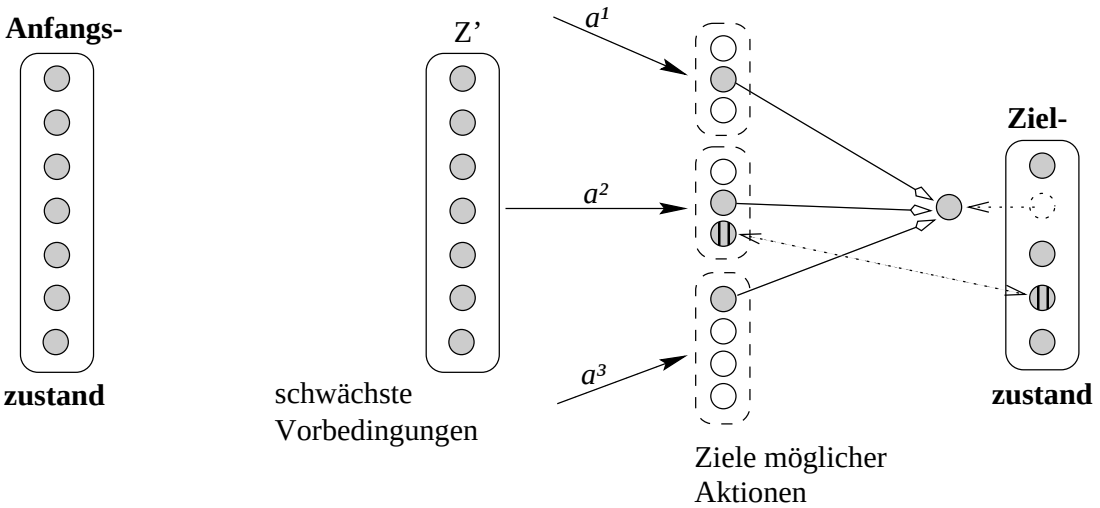


Abbildung 4.20: Skizze einer Regressionsstrategie: Regressionsschritt

Zur Skizzierung der Heuristik gehen wir von einer Planspezifikation aus, die einen Anfangs- bzw. Zielzustand bestehend aus einer Konjunktion von Literalen spezifiziert (vgl. die Abbildung 4.20). Die Strategie wählt ein Teilziel des Zielzustandes aus und bestimmt die

Aktionen, die dieses Ziel erreichen könnten – in der Abbildung die Aktionen $a_1, \ldots, a_3$. Jede dieser Aktionen produziert im allgemeinen neben dem ausgewählten Teilziel, als dem aktuell *primären* Effekt, auch noch weitere, *sekundäre* Effekte; die Gesamtheit der Effekte ist jeweils durch gestrichelte Ovale dargestellt. Die sekundären Effekte können dabei, wie bei Aktion a_2 zu sehen, auch mit anderen Teilzielen des Zielzustandes übereinstimmen (dargestellt durch schraffierte Kreise). Dieser Umstand wird in eine Heuristik umgesetzt, die besagt, daß die Aktion zur Regression ausgewählt wird, deren sekundären Effekte die meisten Teilziele erreicht;[50] In Abbildung 4.20 ist dies die Aktion a_2. Bezüglich der Axiomatisierung dieser Aktion werden nun die schwächsten Vorbedingungen des Zielzustandes berechnet; es ergibt sich der Zustand Z'. Er enthält alle Bedingungen die notwendig sind, damit zum einen Aktion a_2 ausführbar ist und zum anderen nach ihrer Ausführung der spezifizierte Zielzustand erreicht ist. Ausgehend von Z' setzt sich der geschilderte Prozeß nun fort, bis eine schwächste Vorbedingung Z'^n erreicht ist, die aus dem spezifizierten Anfangszustand unmittelbar folgt; Abbildung 4.21 zeigt diesen Status.

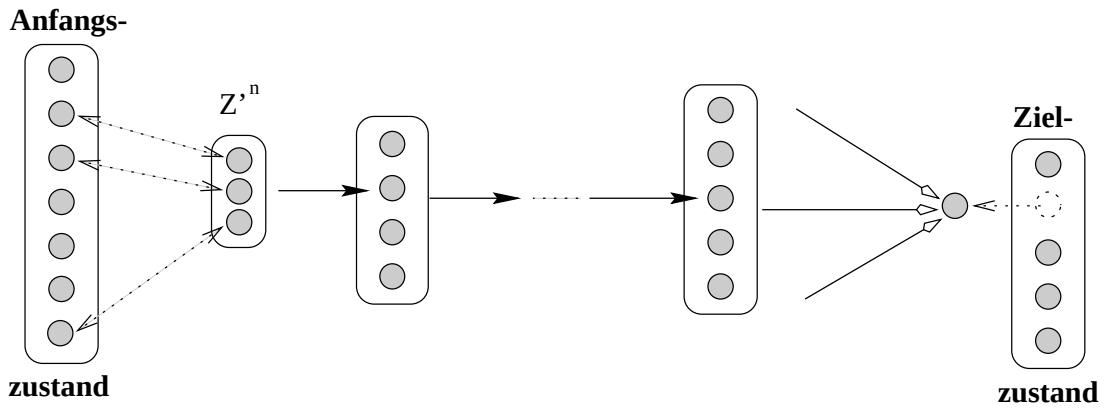


Abbildung 4.21: Skizze einer Regressionsstrategie: Erreichen des Anfangszustandes

Im Vergleich zum progressiven Vorgehen, skizziert in den Abbildungen 4.11-4.13, unterscheidet das geschilderte regressive Vorgehen keine getrennten Such- und Aktionsanwendungsphasen; sie sind vielmehr zusammengefaßt. Die Aktionsanwendung ist während des Planens zunächst jedoch nur eine *hypothetische* Anwendung auf der Kopie eines abgesicherten Beweiszustandes; erst wenn wie in Abbildung 4.21 skizziert eine Verbindung zum Anfangszustand aufgebaut ist, werden alle Aktionsanwendungen abgesichert. Der Test, ob solch ein Verbindungsaufbau möglich ist, muß im allgemeinen nach jeder schrittweisen Verfeinerung des Planes durchgeführt werden, um den Plan nicht unnötig zu verlängern.

[50] Differenziert diese Heuristik nicht eindeutig, so erfolgt eine nichtdeterministische Auswahl unter den verbleibenden Kandidaten.

Betrachten wir nun die beweistechnische Durchführung der Regressionsstrategie. Die initiale Sequenz, bei der die Beweisführung aufsetzt, sei gegeben als

$$pre,\ \mathsf{P} \Rightarrow \Diamond\ [\alpha \wedge \Diamond \bigwedge_{i=1}^{n} \gamma_i]$$

pre eine modalfreie LLP-Formel, P eine Planmetavariable,
γ_i ein LLP-Literal, α eine modalfreie LLP-Formel.

Abbildung 4.22 charakterisiert die globale Struktur des Beweises. Die Struktur der Zielspezifikation führt zunächst zur Einführung einer sequentiellen Struktur in die Planmetavariable P; es resultiert Sequenz (2). Die Anwendung der Regel *liveness split* vollzieht dann die sequentielle Zielstrukturierung. Aufgrund des regressiven Vorgehens wird zunächst nach dem terminalen Teilplan gesucht (Fokussierung von Sequenz (4), Aufschub von Sequenz (3)) und erst nach dessen Existenz wird der initiale Teilplan bestimmt. Es gibt nun aber kein exaktes Kriterium, wann der terminale Teilplan, also der Plan P_2 in Sequenz (4), abgeschlossen ist; dazu müßten die Voraussetzungen, von denen er ausgehen kann, vollständig bekannt sein. Die Frage, wann eine Verfeinerung von P_2 ausreichend ist, ist also eine reine Strategiefrage. Abbildung 4.22 enthält zunächst den Basisfall, daß P_2 dem leeren Plan entspricht.

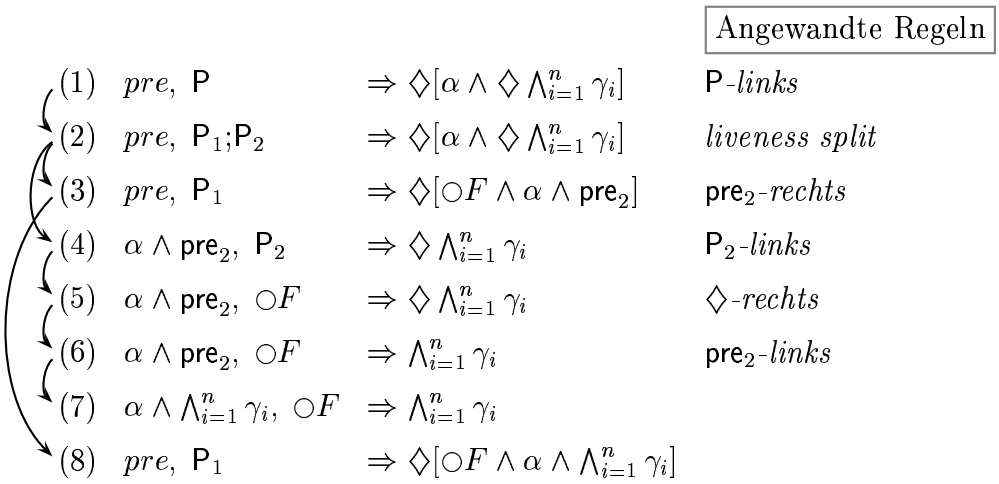


Abbildung 4.22: Durchführung einer Regressionsstrategie: Teil 1

Dadurch hat man nun mit Sequenz (8) möglicherweise ein schwierigeres Problem zu lösen als initial mit Sequenz (1) – ein strukturiertes Ziel wurde zu einem zusammenhängenden Ziel *verschmolzen*. Im allgemeinen wird man also P_2 zunächst verfeinern. Zu bemerken ist noch, daß die Regelanwendung auf Sequenz (3) erst geschieht, wenn Sequenz (7) erreicht ist.
Abbildung 4.23 zeigt einen Verfeinerungsschritt des Teilplanes P_2; es wird angenommen, daß er aus mindestens zwei Teilen besteht. Durch die Instantiierung von P_4 wird die *letzte* Aktion des Gesamtplanes festgelegt. Eingeführt wurde sie durch eine Regel, die einem speziellen Axiom der Aktion a entspricht. Dieses Axiom beschreibt gemäß Definition 4.2.6,

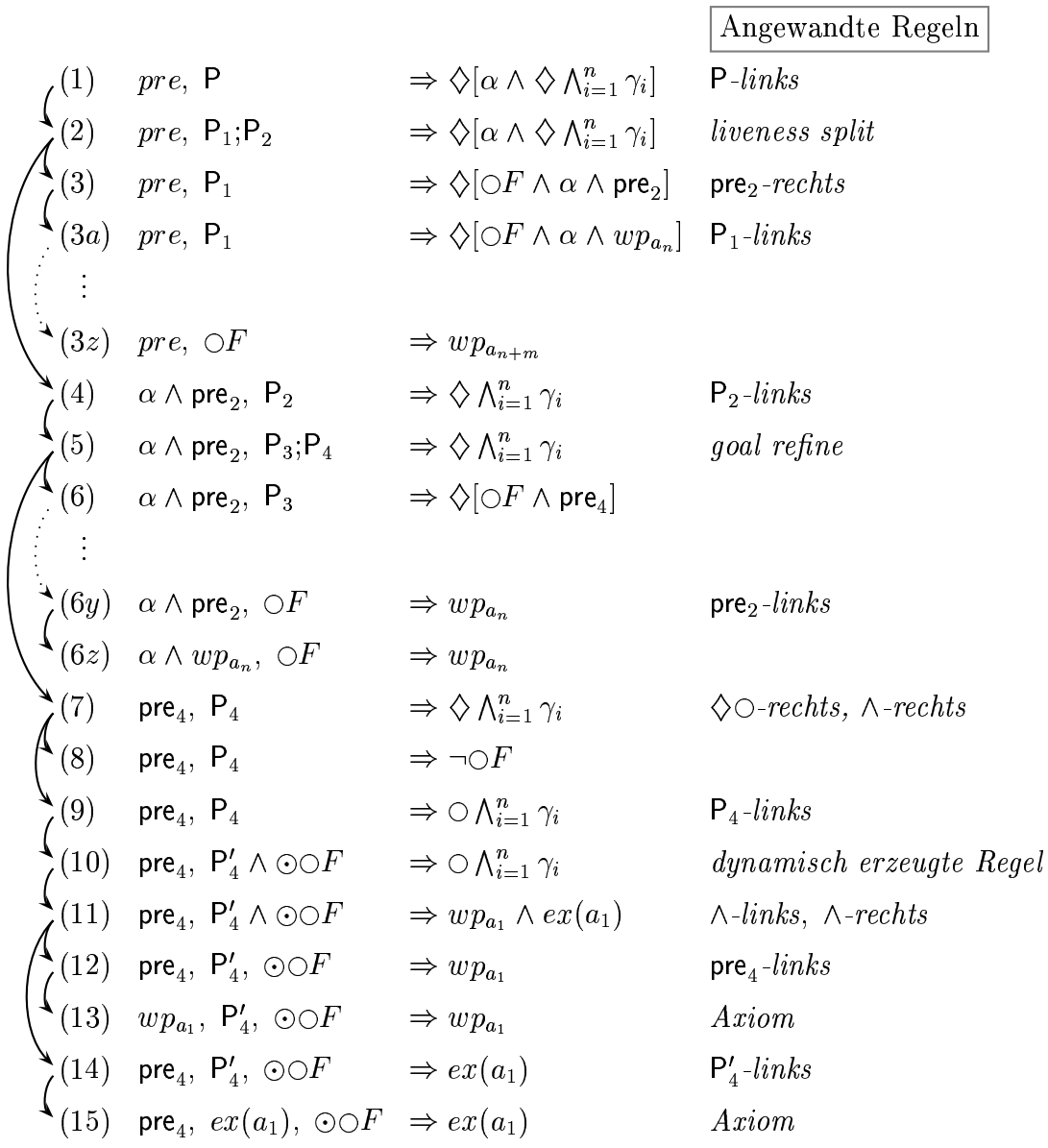


Abbildung 4.23: Durchführung einer Regressionsstrategie: Teil 2

welche schwächsten Vorbedingungen notwendig sind, um den Effekt $\bigwedge_{i=1}^{n} \gamma_i$ mit der Aktion a hervorzurufen. Damit beschreibt dieses Axiom im allgemeinen sowohl primäre Effekte als auch Rahmenbedingungen der Aktion. Die Auswahl der Aktion a geschieht nach dem in Abbildung 4.20 skizzierten Vorgehen.
Wendet man nun auf Sequenz (6) die Regel pre_4-rechts an, so ergibt sich ein neues Zwischenziel als Folge der ersten Verfeinerung. Mit ihm kann nun analog zu dem Vorgehen in Abbildung 4.22 vorgegangen werden oder es erfolgt eine weitere Verfeinerung. Wann dieser Prozeß beendet wird, d.h. wann die Suche nach dem initialen Plan P_1 begonnen wird, obliegt einer entsprechenden Heuristik, die einen Abbruchtest für die rekursive Verfeinerung realisiert. Letztendlich führt dies jedenfalls dazu, eine Instantiierung der Metavariable pre_2 in Form der schwächsten Vorbedingungen für den Plan P_2 zu finden. Danach ist es dann möglich, den Beweis mit Sequenz (3) weiter fortzuführen. Die Verfeinerung des Teilplanes P_1 findet ihr natürliches Ende, sobald eine Verbindung zum Anfangszustand, beschrieben durch *pre*, hergestellt ist (vgl. dazu nochmals die Skizze in Abbildung 4.21); die Sequenz (3z) beschreibt das entstehende Beweisproblem.
Ein sequentieller Plan, der sich nach erfolgreichem Beweis des gesamten Planungsproblems ergibt, hat eine Struktur, die durch folgende Grammatik charakterisiert werden kann:

$$
\begin{array}{rcl}
\texttt{Plan} & ::= & \bigcirc F \ ; \ \texttt{S\_Plan} \ \mid \\
 & & \bigcirc F \\
\texttt{S\_Plan} & ::= & \texttt{S\_Plan} \ ; \ \texttt{S\_Plan} \ \mid \\
 & & ex(Aktionsterm) \wedge \odot\bigcirc F \ \mid \\
 & & \bigcirc F
\end{array}
$$

Die erfolgreiche Durchführung der regressiven Planungsstrategie führt aus der Sicht der Modellverfeinerung zusammenfassend zu der folgenden Intervallstrukturierung:

$$
\begin{array}{llllll}
\langle\sigma_0\rangle & & & & & \\
\langle\sigma_0\sigma_{z-x}\rangle\langle\sigma_{z-x}\sigma_{z-x+1}\rangle & & & & & \\
\vdots & & & & & \\
\langle\sigma_0 & \ldots & & \sigma_{z-2}\rangle\langle\sigma_{z-2}\sigma_{z-1}\rangle & & \\
\langle\sigma_0 & & \ldots & & \sigma_{z-1}\rangle\langle\sigma_{z-1}\sigma_z\rangle & \\
\langle\sigma_0 & & & \ldots & & \sigma_z\rangle
\end{array}
$$

Die Unifikation von x mit z und damit einhergehend die Verfeinerung durch Längeneinführung mit dem Wert 0 zu Beginn des Intervalles σ schließt die gesamte Verfeinerung ab. Die regressive Intervallverfeinerung vollzieht sich anschaulich in entgegengesetzter Richtung zur progressiven Strukturierung (vgl. Seite 148).

Die allgemeine Schwierigkeit, eine geeignete Heuristik zur Steuerung der regressiven Verfeinerung zu definieren, wird durch das Problem, den *Erzeuger* eines Teilzieles zu finden, hervorgerufen.
Für ein Teilziel γ_i gibt es dabei 3 Möglichkeiten für seine Entstehung:

1. γ_i kann aus den initialen Vorbedingungen, *pre*, direkt abgeleitet werden und wird weder durch P_1 noch durch P_2 wieder zerstört;

2. es wird in P_1 oder in P_2 bewußt ein Teilplan eingefügt, der γ_i erreicht und es bleibt erhalten;

3. es gibt in P_1 oder in P_2 einen Teilplan, der γ_i als Nebeneffekt schon miterreicht und es bleibt erhalten.

Nach erfolgreichem Beweis von Sequenz (1) sei ein linearer Plan mit $m+n$ Aktionen entstanden, d.h. mit einem möglichen Modell, das die Planspezifikation erfüllt, ist ein Intervall $\sigma = \langle \sigma_0 \ldots \sigma_{m+n} \rangle$ assoziiert. Zu einem notwendigen Ziel ζ_i, das im Zustand σ_l gilt, gibt es einen Zustand σ_j mit $j \leq l$ und ζ_i gilt in allen Zuständen σ_k mit $j \leq k < l$. Mit $Ee(\zeta_i) := l - j$ bezeichnen wir die *Erzeugerentfernung* von ζ_i.
Liegt ein vollständiger Plan mit assoziiertem Intervall σ vor, so kann zu jedem notwendigen Ziel ζ_i ein Wert

$$MaxEe(\zeta_i) := max(\{e \mid e := Ee(\zeta_i) \text{ eine Erzeugerentfernung bzgl. } \sigma\})$$

als die *maximale Erzeugerentfernung* und

$$MinEe(\zeta_i) := min(\{e \mid e := Ee(\zeta_i) \text{ eine Erzeugerentfernung bzgl. } \sigma\})$$

als die *minimale Erzeugerentfernung* definiert werden.
Bezogen auf eine regressive Planungsstrategie bedeutet dies nun, daß für ein Ziel ζ_i der Wert $MaxEe(\zeta_i)$ während der Generierung eines Planes steigen kann, selbst wenn keine Planungsentscheidungen widerrufen werden (es gilt im allgemeinen $MinEe(\zeta_i) \leq MaxEe(\zeta_i)$). Abbildung 4.24 skizziert diesen Umstand.
Teil (a) zeigt die 3 letzten Aktionen eines Planes zusammen mit der Begründung für ihre Einführung. Aktion a_3 wurde eingeführt, um Teilziel z_1 zu erreichen, Aktion a_2 für Teilziel z_5 und Aktion a_1 für Teilziel z_3; die Annahme sei, daß keine der Aktionen die Effekte anderer Aktionen zerstört. Es gilt u.a. $MaxEe(z_1) = 2$.
In Teil (b) werden die Auswirkungen einer weiteren Aktion gezeigt. Aktion a_4 wurde eingeführt, um das Ziel z_4 zu erreichen. Als Nebeneffekt entstehe durch diese Aktion aber auch das Ziel z_1. Dadurch verändert sich der Wert von $MaxEe(z_1)$ zu 3; es ist also ein weiterer Erzeuger für das Ziel z_1 aufgetreten und damit ist in dieser Situation die Aktion a_3 überflüssig geworden und bestätigt jetzt nur noch einmal, was sowieso schon gilt. Eine rein regressiv vorgehende Strategie kann solch eine Situation nicht erkennen, sie kann Nebeneffekte von Aktionen nur unmittelbar im Kontext einer Aktion verwenden.

Für die zielgerichtete progressive Strategie, wie sie in dem vorhergehenden Abschnitt beschrieben wurde, gilt hingegen für jedes Ziel ζ_i: $MinEe(\zeta_i) = MaxEe(\zeta_i)$, d.h. kein Plan enthält redundante Aktionen. Die Suchphase dieser Strategie, wie sie in den Abbildungen 4.11–4.12 skizziert wurde, entspricht einer partiellen, *minimalen* Regression. Darauf folgt dann ein Progressionsschritt. Aufgrund der lediglich partiell ausgeführten Regression verliert man jedoch die Sicherheit, daß die Zwischenziele, die aufgebaut werden, tatsächlich auch in der erzeugten Reihenfolge erreicht werden können. Es kann sich für ein Zwischenziel z – in der partiell regressiven Suchphase entstanden – während der Progressionsphase herausstellen, daß die Vorbedingungen der einzigen Aktion, die z erzeugen kann, mit den aktuell geltenden stärksten Nachbedingungen inkonsistent sind, d.h. die Aktion, nennen wir sie a, kann nicht angewendet werden und das Erreichen von z

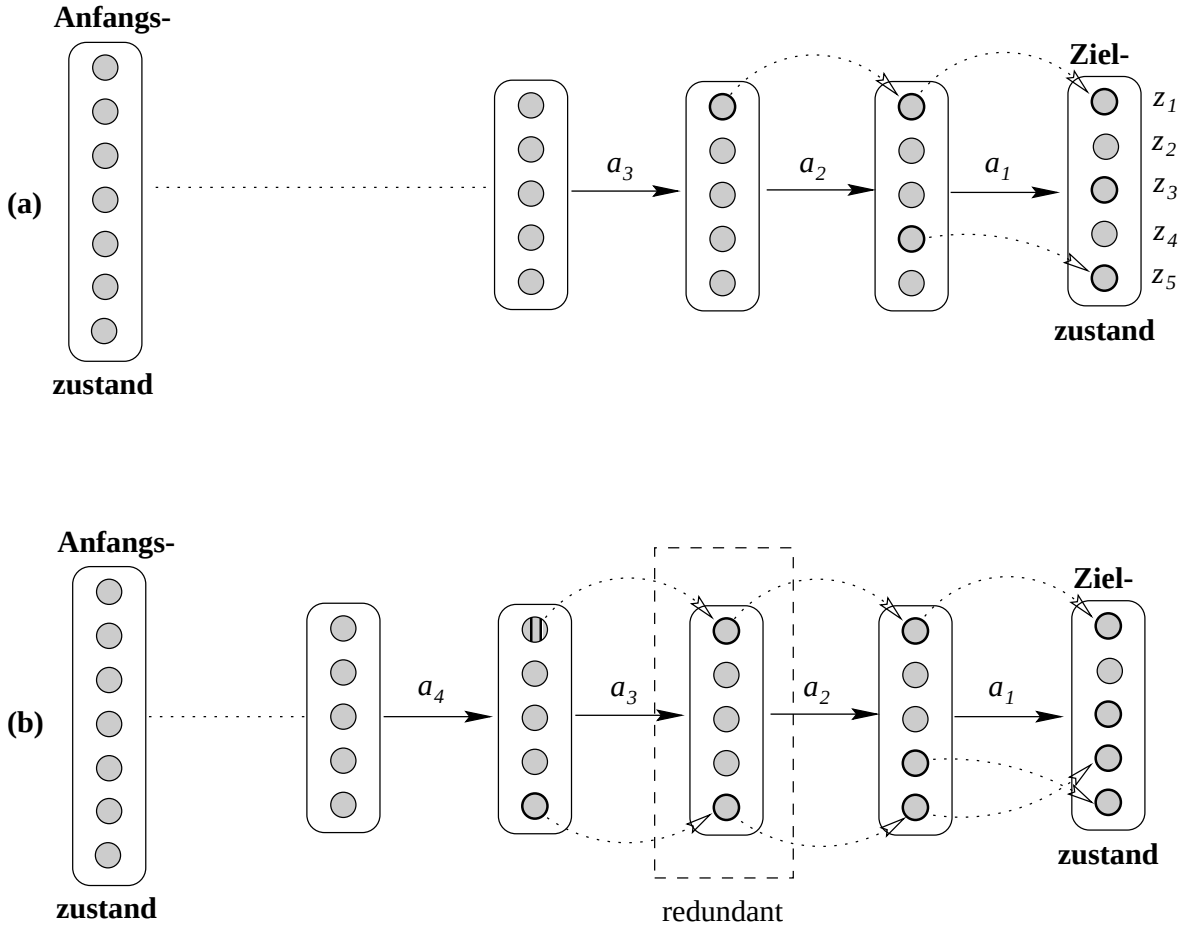


Abbildung 4.24: Skizze einer Regressionsstrategie: Redundante Aktionen

scheitert. Diesen Konflikt hätte man früher aufdecken können, wenn alle Vorbedingungen von a während der Suchphase und der bisherigen Progressionsphase berücksichtigt worden wären. Dies führt in letzter Konsequenz dazu, daß die Suchphase bereits durch Ausführung einer vollständigen Regression realisiert ist, nur dann erhält man absolute Sicherheit über die Effektivität der Zwischenziele. Abbildung 4.25 stellt zwei wesentliche Planungseigenschaften zu Progression und Regression in Beziehung.

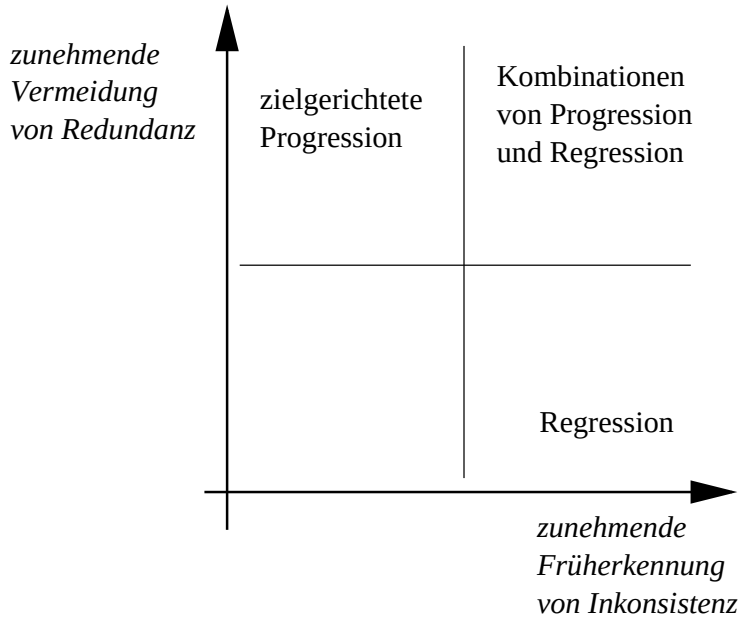


Abbildung 4.25: Vergleich von Progression und Regression

Wenn beide Eigenschaften erwünscht sind, so kann dies durch eine Kombination von regressivem und progressivem Vorgehen erreicht werden.
Die vorgestellte Regressionsstrategie kann in diesem Sinne ergänzt werden um einen nachfolgenden *Progressionsdurchlauf*, der redundante Aktionen wieder aus dem Plan entfernt. Analog kann die zielgerichtete Progressionsstrategie in ihrer Suchphase eine vollständige Regression verwenden, um die Zwischenziele aufzubauen.[51] Dies entspricht dann der Regressionsstrategie mit dem Unterschied, daß während der Regressionsphase noch keine Instantiierung von Teilplänen vorgenommen wurde, und somit das Entfernen von redundanten Aktionen in der Progressionsphase vereinfacht ist.

4.2.7.2 Nichtdeterministische Pläne

Unter nichtdeterministischen Plänen wollen wir Pläne verstehen, die den Nichtdeterminismus explizit im Plan enthalten.[52] Im folgenden betrachten wir zwei Varianten, Nichtdeterminismus auf der Aktionsebene in Plänen einzuführen. Auslöser für eine mögliche Einführung ist die Tatsache, daß im allgemeinen bestimmte Effekte durch die Ausführung

[51] Dabei geschieht die Progression bzgl. aller zu erreichenden Ziele.

[52] Implizit ist der Nichtdeterminismus bei dem verwendeten Planbegriff und entsprechender Planabstraktionsstufe per Definition enthalten.

unterschiedlicher Aktionen erreicht werden können. Die Motivation, die Einführung letztendlich durchzuführen, kann unterschiedlich sein:

- die Entscheidung für die Verwendung einer Aktion soll nicht während des Planungsprozesses gefällt werden;
- Optionen bei der Aktionsauswahl sollen explizit sichtbar sein;
- die Auswirkungen unterschiedlicher Aktionen auf den entstehenden Plan sollen nachvollziehbar sein.

Betrachten wir zunächst die Etablierung nichtdeterministischer Pläne aus Verfeinerungssicht. Gemäß Abschnitt 4.2.4 geschieht dies durch geeignete Disjunktions-Verfeinerungsschritte. Die Positionierung dieser Schritte in der Gesamtstruktur einer Planverfeinerung entscheidet darüber, welche Art von Nichtdeterminismus letztendlich im Plan explizit erscheint.

Aktions-Nichtdeterminismus: Der Nichtdeterminismus erscheint nur direkt zwischen alternativen Aktionen. Unter der Annahme, daß eine progressive Strategie durchgeführt wird, befindet man sich bezogen auf ein Intervall $\sigma = \langle \sigma_0 \ldots \sigma_z \rangle$ während des Verfeinerungsprozesses etwa in folgendem Zustand:

$$\left\langle \boxed{\begin{matrix} pre_k \\ ? \end{matrix}}_{i} \boxed{\begin{matrix} \gamma_k \wedge \mathsf{pre} \\ ? \end{matrix}}_{i+1} \right\rangle \langle \sigma_{i+1} \ \ldots \ \sigma_z \rangle$$

Eine detaillierte Verfeinerung bis zum Zustand i sei also bereits durchgeführt. Die Spezifikation des Teilintervalles $\langle \sigma_i \sigma_{i+1} \rangle$ entspreche bereits der folgenden Sequenz:

$$pre_k,\ \mathsf{P}_i \wedge \odot\bigcirc F \Rightarrow \bigcirc[\gamma_k \wedge \mathsf{pre}]$$

Diese Spezifikation könnte mit Hilfe einer Aktionsetablierung zu einem Aktionsmodell abgesichert werden; die Metavariable pre nimmt dabei Nachbedingungen bzgl. der verwendeten Aktion auf. Die Länge des Intervalles ist auf exakt 1 festgelegt. Nehmen wir nun an, daß prinzipiell n alternative Aktionen $a_1, \ldots, a_n$ existieren, auf deren Grundlage die Aktionsetablierung durchgeführt werden könnte. Um allen Aktionen gerecht zu werden, wird $\langle \sigma_i \sigma_{i+1} \rangle$ zunächst im Sinne einer Disjunktion verfeinert zu:

$$\left\{ \left\langle \boxed{\begin{matrix} pre_k \\ a_1 \end{matrix}}_{i} \boxed{\begin{matrix} \gamma_k \wedge \mathsf{pre} \\ ? \end{matrix}}_{i+1} \right\rangle, \ldots, \left\langle \boxed{\begin{matrix} pre_k \\ a_n \end{matrix}}_{i} \boxed{\begin{matrix} \gamma_k \wedge \mathsf{pre} \\ ? \end{matrix}}_{i+1} \right\rangle \right\}$$

Die Sequenz verfeinert sich dadurch entsprechend zu:

$$pre_k,\ [ex(a_1) \vee \ldots \vee ex(a_n)] \wedge \odot\bigcirc F \Rightarrow \bigcirc[\gamma_k \wedge \mathsf{pre}]$$

Die Anwendung der Sequenzenregel $\wedge$-*links* gefolgt von $n-1$ Anwendungen der Regel $\vee$-*links* führt zu n individuellen aber zusammenhängenden Sequenzen:

$$\begin{aligned} pre_k,\ ex(a_1) \wedge \odot\bigcirc F &\Rightarrow \bigcirc[\gamma_k \wedge \mathsf{pre}] \\ &\vdots \\ pre_k,\ ex(a_n) \wedge \odot\bigcirc F &\Rightarrow \bigcirc[\gamma_k \wedge \mathsf{pre}] \end{aligned}$$

Die weitere Verarbeitung der Sequenzen kann angelehnt an die in Abbildung 4.15 skizzierten Beweisschritte geschehen. Die obigen n Sequenzen enthalten alle die selbe Metavariable pre zur Aufnahme progressiv erzeugter Nachbedingungen der jeweiligen Aktion. Eine Instantiierung von pre, z.B. bzgl. der Aktion a_i, darf aber nun im allgemeinen nicht die *stärksten* Nachbedingungen bzgl. einer Aktion a_i enthalten, sondern muß die übrigen Aktionen gleichermaßen berücksichtigen, d.h. pre sollte höchstens mit dem *Durchschnitt* aller individuellen stärksten Nachbedingungen instantiiert werden. Nur so können die individuellen Beweise erfolgreich geführt werden und die Auswirkungen der Disjunktion lokal begrenzt bleiben.[53]

Plan-Nichtdeterminismus: Der Nichtdeterminismus erscheint in Form alternativer Pläne. Der Verfeinerungsprozeß befinde sich hierzu in einem Zustand, in dem es ein wie folgt charakterisierbares Intervall zu verfeinern gilt:

$$\left\langle \boxed{\begin{matrix} pre_i \\ ? \end{matrix}}_i \cdots \boxed{\begin{matrix} \phi \\ ? \end{matrix}}_j \right\rangle,\ i \leq j.$$

Dieses Intervall korrespondiert etwa mit einer Sequenz der Form

$$pre_i,\ \mathsf{P}_1 \Rightarrow \Diamond[\bigcirc F \wedge \phi]$$

Es können im allgemeinen verschiedene Aktionen existieren, die, initial ausgeführt, einen Beitrag zur Erreichung des Zieles ϕ leisten können; seien dies die Aktionen $a_1, \ldots, a_n$. Eine entsprechende disjunktive Verfeinerung des Intervalles – nützlich für ein progressives Vorgehen – führt zu folgender Verfeinerung:

$$\left\{ \left\langle \boxed{\begin{matrix} pre_i \\ a_1 \end{matrix}}_i \cdots \boxed{\begin{matrix} \phi \\ ? \end{matrix}}_{j_1} \right\rangle, \ldots, \left\langle \boxed{\begin{matrix} pre_i \\ a_n \end{matrix}}_i \cdots \boxed{\begin{matrix} \phi \\ ? \end{matrix}}_{j_n} \right\rangle \right\},\ i \leq j_k,\ 1 \leq k \leq n$$

Eine korrespondierende Spezifikation ist durch eine Sequenz

$$pre_i,\ [ex(a_1);\mathsf{P}_1^1] \vee \ldots \vee [ex(a_n);\mathsf{P}_1^n] \Rightarrow \Diamond[\bigcirc F \wedge \phi]$$

gegeben. Jeder der einzelnen Teilpläne enthält somit bereits eine Einschränkung, welches seine erste Aktion ist. Die $n-1$-malige Anwendung der Regel $\vee$-*links* auf die obige Sequenz führt zu den alternativen Teilproblemen.

Angepaßt an ein regressives Vorgehen stellt sich das Problem einer nichtdeterministischen Verfeinerung unter dem Gesichtspunkt, welche Aktionen das Ziel ϕ durch Regression reduzieren können. Seien dies die Aktionen $a_1, \ldots, a_m$, so führt dies zu folgender Verfeinerung:

$$\left\{ \left\langle \boxed{\begin{matrix} pre_i \\ ? \end{matrix}}_{i_1} \cdots \boxed{a_1}_{j-1} \boxed{\begin{matrix} \phi \\ ? \end{matrix}}_j \right\rangle, \ldots, \left\langle \boxed{\begin{matrix} pre_i \\ ? \end{matrix}}_{i_m} \cdots \boxed{a_m}_{j-1} \boxed{\begin{matrix} \phi \\ ? \end{matrix}}_j \right\rangle \right\},\ i_k \leq j,\ 1 \leq k \leq m$$

[53] Eine Instantiierung von pre im Sinne der stärksten Nachbedingungen des nichtdeterministischen Teilplanes bildet ebenfalls eine Disjunktion.

Eine korrespondierende Spezifikation ist durch eine Sequenz

$$pre_i,\ [\mathsf{P}_1^1;ex(a_1) \wedge \odot\bigcirc F] \vee \ldots \vee [\mathsf{P}_1^m;ex(a_m) \wedge \odot\bigcirc F] \Rightarrow \Diamond[\bigcirc F \wedge \phi]$$

gegeben. Jeder der Teilpläne enthält eine Einschränkung über die letzte Aktion, die er enthalten muß.

Der Aktionsnichtdeterminismus, so wie er oben dargestellt wurde, stellt lediglich einen Spezialfall des eingeführten Plannichtdeterminismus dar.
In einem Intervall $\sigma = \langle\sigma_0 \ldots \sigma_z\rangle$, das mit einer initialen Planspezifikation assoziiert ist, kann eine disjunktive Verfeinerung an verschiedenen Stellen positioniert sein. Man kann prinzipiell vier Möglichkeiten unterscheiden:[54]

1. bzgl. des gesamten Intervalles

$$\{\langle\sigma_{0_1} \ldots \sigma_{z_1}\rangle, \langle\sigma_{0_2} \ldots \sigma_{z_2}\rangle\}$$

2. zu Beginn des Intervalles

$$\{\langle\sigma_{0_1} \ldots \sigma_{i_1}\rangle, \langle\sigma_{0_2} \ldots \sigma_{i_2}\rangle\}\ \langle\sigma_i \ldots \sigma_z\rangle$$

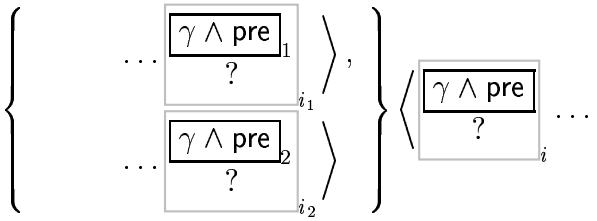
3. am Ende des Intervalles

$$\langle\sigma_0 \ldots \sigma_i\rangle\ \{\langle\sigma_{i_1} \ldots \sigma_{z_1}\rangle, \langle\sigma_{i_2} \ldots \sigma_{z_2}\rangle\}$$

4. mitten im Intervall

$$\langle\sigma_0 \ldots \sigma_i\rangle\ \{\langle\sigma_{i_1} \ldots \sigma_{j_1}\rangle, \langle\sigma_{i_2} \ldots \sigma_{j_2}\rangle\}\ \langle\sigma_j \ldots \sigma_z\rangle$$

Im 1. Fall sind die beiden Alternativen vollständig voneinander entkoppelt. In den Fällen 2. bis 4. stehen die Alternativen über gemeinsame *Verschmelzungspunkte* in Verbindung. Da hier der disjunktiven Verfeinerung stets eine Aufsplittungsverfeinerung vorausging, kann ihr auch die Aufgabe zukommen, eine Instantiierung des Verschmelzungspunktes vorzunehmen. Betrachten wir dazu die Fälle 2. und 3. im Detail, auch unter dem Gesichtspunkt, daß progressives und regressives Vorgehen aufeinandertreffen.[55]
Zunächst der 2. Fall:[56]



Es ergeben sich vier Kombinationen des Aufeinandertreffens von Progression und Regression:

[54] Wir führen die Verfeinerung jeweils bzgl. zweier Alternativen durch und zeigen das strukturelle Ergebnis.

[55] Der Fall 4. ist lediglich die Kombination der Fälle 2. und 3.

[56] γ bezeichne eine modalfrei LLP-Formel, pre bezeichne eine Metavariable, die typischerweise durch die frühere Anwendung einer Aufsplittungsregel entsteht.

Progression trifft Progression: Wir nehmen o.B.d.A. an, daß der nichtdeterministische Teil vor dem deterministischen weiter verfeinert wird. Im allgemeinen ergibt sich, daß die Verfeinerung der beiden Alternativen prinzipiell unterschiedliche Angebote an stärksten Nachbedingungen verfügbar machen. Für die Instantiierung von pre – also dem gemeinsamen Angebot an Nachbedingungen – stehen zwei Optionen zur Verfügung:

- Die Instantiierung mit der Disjunktion der individuellen stärksten Nachbedingungen in der Form $sp_1 \vee sp_2$. Das Angebot dieser effektiv stärksten Nachbedingungen eines nichtdeterministischen Planes führt jedoch dazu, daß sich die Auswirkungen des Nichtdeterminismus *global* fortpflanzen.
- Die Instantiierung mit dem "Durchschnitt" $sp^{\cap}$ der individuellen Nachbedingungen sp_1 und sp_2, d.h. der maximalen Formel, die sowohl aus sp_1 als auch aus sp_2 folgt; es gilt dabei: $sp_1 \rightarrow sp^{\cap}$, $sp_2 \rightarrow sp^{\cap}$ und für jedes ψ mit $sp_1 \rightarrow \psi$ und $sp_2 \rightarrow \psi$ gilt $sp^{\cap} \rightarrow \psi$. $sp^{\cap}$ ist die Formel, die *sicher* für die weitere deterministische Verfeinerung zur Verfügung steht; die Auswirkungen des Nichtdeterminismus sind somit *lokal* geblieben.

Wird der deterministische Teil vor dem nichtdeterministischen verfeinert, so ist die Instantiierung von pre unwesentlich – sie wird mit einem Wert äquivalent zu *true* instantiiert.[57]

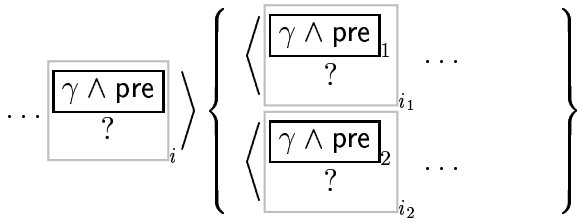
Progression trifft Regression: Wird zunächst der nichtdeterministische Teil verfeinert, so gilt für die Instantiierung von pre das gleiche, wie im Fall **Progression trifft Progression**. Erfolgt zunächst eine Verfeinerung des deterministischen Teiles, so wird die Instantiierung von pre durch den Regressionsprozeß als die jeweilige schwächste Vorbedingung wp für den assoziierten Plan festgelegt. wp erscheint dann in den disjunktiven Alternativen als zusätzliches, notwendigerweise zu erreichendes Ziel.

Regression trifft Regression: Nehmen wir an, daß zunächst der deterministische Teil verfeinert wird. Die Instantiierung von pre wird auch hier durch die jeweilige schwächste Vorbedingung wp für den assoziierten Plan festgelegt. wp wird zum notwendigerweise zu erreichenden Ziel in beiden alternativen Verfeinerungen. Wird der nichtdeterministische Teil als erstes verfeinert, so muß pre zunächst instantiiert werden. Da keine Information verfügbar ist, was zusätzlich zu γ sinnvollerweise erreicht werden könnte, wird die Instantiierung mit einer Formel äquivalent zu *true* vorgenommen. Der deterministische Teil muß dann mit γ als verfügbare Vorbedingung auskommen.

Regression trifft Progression: Hier gilt für beide Reihenfolgevarianten, daß pre mit einer Formel äquivalent zu *true* zu instantiieren ist, solange keine Information über die Nutzung einer spezifischeren Instantiierung vorliegt. Beide Verfeinerungsprozesse arbeiten hier völlig ohne gegenseitige Beeinflussung.

[57] Zusätzlich eingeführte Annahmen erweitern höchstens γ und erscheinen damit als notwendige Ziele.

Hier nun der 3. Fall im Detail:



Auch hier müssen vier Kombinationen des Aufeinandertreffens von Progression und Regression berücksichtigt werden.

Regression trifft Regression: Nehmen wir an, daß der nichtdeterministische Teil als erstes verfeinert wird. Im allgemeinen entstehen in den Alternativen unterschiedliche schwächste Vorbedingungen wp_1 bzw. wp_2. Sie stellen beide individuelle Anforderungen an den Verschmelzungspunkt, also die Instantiierung von pre. Für diese ergeben sich drei Optionen:

- Die Instantiierung durch die *Vereinigung* $wp^{\cup}$ der beiden Anforderungen wp_1 und wp_2, also $wp^{\cup} \equiv wp_1 \vee wp_2$. Durch diese effektiv schwächsten Vorbedingungen des nichtdeterministischen Planes stellt man sicher, daß man aus globaler Sicht im Zustand σ_i *sicher* die Möglichkeit hat, eine der beiden alternativen Pläne auszuführen. Die Entscheidung, welche der Alternativen ausgeführt werden kann, fällt dabei im allgemeinen nicht erst unmittelbar vor dem nichtdeterministischen Teilplan.
- Die Instantiierung durch eine Formel δ mit der Eigenschaft $\delta \rightarrow wp_1$ oder $\delta \rightarrow wp_2$. Damit verliert man u.U. die Garantie, daß beide Alternativen ausführbar sind; die Auswirkungen der disjunktiven Verfeinerung sind jedoch lokal geblieben und die Anforderungen an den deterministischen Teil sind reduziert.
- Die Instantiierung durch die Formel $wp_1 \wedge wp_2$, falls diese konsistent ist. Hiermit stellt man aus globaler Sicht sicher, daß beide Alternativen ausführbar sind; die Entscheidung welche Alternative ausgewählt wird, fällt unmittelbar vor dem nichtdeterministischen Teilplan. Diese Freiheit verschärft natürlicherweise die Anforderungen an den deterministischen Teil.

Wird zuerst der deterministische Teil verfeinert, so kann pre typischerweise nur mit einer Formel äquivalent zu *true* instantiiert werden.

Regression trifft Progression: Beide Verfeinerungsprozesse arbeiten hier gegensätzlich und somit ist ohne zusätzliche Information nur die Instantiierung von pre durch eine mit *true* äquivalente Formel sinnvoll.

Progression trifft Progression: Wird als erstes der deterministische Teil verfeinert, ergibt sich die Instantiierung von pre als die stärkste Nachbedingung des assoziierten Planes; diese steht dann für beide nichtdeterministischen Alternativen zur Verfügung. Erfolgt hingegen zuerst eine Verfeinerung des nichtdeterministischen Teiles, bietet sich typischerweise wiederum nur eine Instantiierung von pre durch eine mit *true* äquivalente Formel an.

Progression trifft Regression: Erfolgt zuerst die Verfeinerung des deterministischen Teiles, bilden wiederum die stärksten Nachbedingungen die Instantiierung von pre. Damit erhöht sich für die disjunktiven Alternativen das Angebot an verfügbaren Voraussetzungen. Erfolgt zunächst die Regression, ergibt sich eine Instantiierung analog zu dem Fall **Regression trifft Regression**. Für die anschließende Progression führt dies dann zu mehr oder weniger spezifischen zusätzlichen Zielen.

Hat man sich beim Beweis einer Planspezifikation für eine disjunktive Verfeinerung eines Teilintervalles entschieden und eine entsprechende nichtdeterministische Planstruktur eingeführt, so müssen nicht alle den Alternativen entsprechenden Teilbeweise zufriedenstellend geführt werden können. Damit durch den Abbruch eines der Teilbeweise nicht der gesamte Beweis ergebnislos abgebrochen wird, kann entweder die entsprechende Planmetavariable (in den oben dargestellten Sequenzen ein entsprechendes P_1^i) mit $false$ instantiiert werden, oder die Einführung der disjunktiven Planstruktur kann modifiziert werden, indem das entsprechende Disjunktionsglied entfernt wird. Die erste der beiden Varianten hat den Vorteil, daß sie explizit das Scheitern eines Teilbeweises als Auswirkung einer bestimmten Aktionsauswahl anzeigt.[58]

Die Planstrukturgrammatik sequentieller Pläne kann durch folgende Regeln erweitert werden, um auch nichtdeterministische Pläne syntaktisch zu beschreiben:

$$\begin{aligned} \texttt{Plan} &::= \texttt{Plan} \vee \texttt{Plan} \\ \texttt{S\_Plan} &::= \texttt{S\_Plan} \vee \texttt{S\_Plan} \end{aligned}$$

Adaptierte Sequenzenregelerzeugung Um den oben diskutierten Auswirkungen von lokalem und globalem Nichtdeterminismus gerecht zu werden, ist es notwendig, einen adaptierten Mechanismus zur Erzeugung von Metavariableninstantiierungsregeln zur Festlegung der Rahmenaxiomverwendung bereitzustellen. Für den Fall der Erzeugung sequentieller Pläne besteht, wie auf Seite 146 beschrieben, nur eine Anforderung an die Instantiierung einer Metavariablen zur Aufnahme einer durch ein Rahmenaxiom abzusichernden Eigenschaft. Unter direkter Verwendung von Definition 4.2.6 wird eine Instantiierung durch stärkste Nachbedingungen vorgenommen.
Am Beispiel des Aktionsnichtdeterminismus wurden auf Seite 160 n verschiedene Anforderungen (entsprechend den Aktionen $a_1, \ldots, a_n$) an die Metavariable pre zur Aufnahme von Rahmeneigenschaften gestellt.[59] Eine Instantiierung kann nun nicht mehr wie im Fall sequentieller Pläne individuell entschieden werden. Allen Anforderungen ist entsprechend dem ausgewählten Modus von lokalem oder globalem Nichtdeterminismus zu genügen. Die strategische Steuerung des Planungsprozesses muß nun, um effizient zu arbeiten, dafür sorgen, daß zunächst alle Anforderungen an die entsprechende Metavariable gesammelt werden, bevor eine Instantiierungsregel bestimmt wird. Bezüglich der Aktionen $a_1, \ldots, a_n$ ergeben sich dann zunächst, wiederum unter Verwendung von Definition 4.2.6, stärkste Nachbedingungen $sp_1, \ldots, sp_n$. Daraus ergibt sich dann eine Instantiierung von pre durch $sp_1 \vee \ldots \vee sp_n$ (für den globalen Fall) oder durch $sp^{\cap}$ (für den lokalen Fall). Aufgrund der Beziehungen $sp_i \rightarrow sp_1 \vee \ldots \vee sp_n$, $\forall 1 \leq i \leq n$ und $sp_i \rightarrow sp^{\cap} \forall 1 \leq i \leq n$ sind beide

[58] Wenngleich der Grund für das Scheitern nicht sichtbar ist.

[59] Obwohl beispielhaft ein progressives Vorgehen geschildert ist, gilt Analoges für ein regressives Vorgehen.

Instantiierungen effektiv und gewährleisten die individuelle Absicherung der disjunktiven Verfeinerung bzgl. der beteiligten alternativen Aktionen.

4.2.7.3 Konditionale Pläne

Die Einführung nichtdeterministischer Pläne war bisher aufgrund von Zieletablierungsalternativen in Form unterschiedlicher Aktionen motiviert. Das ursächliche Problem war, daß eine Auswahl zwischen Aktionen, die ein Ziel entweder direkt erreichten oder indirekt an dessen Erreichung beteiligt waren, zu treffen war. Die Verwendung eines nichtdeterministischen Planes war hier also lediglich eine *Option*.
Es kann während des Planens aber auch zu einer Konfliktsituation kommen, bei der die Einführung eines nichtdeterministischen Planes *notwendig* ist, um den Plan weiterentwickeln oder überhaupt erstellen zu können. Diese Situation tritt z.B. auf, wenn es nicht gelingt, notwendige Vorbedingungen einer Aktion, die zur Erreichung eines Zieles beiträgt, aus den zur Verfügung stehenden Zustandsbeschreibungen herzuleiten. Die Konfliktlösung erfolgt durch die Etablierung einer *vollständigen Fallunterscheidung*.
Gehen wir zur Erläuterung des Sachverhaltes von einer allgemeinen Problemspezifikation

$$pre, \mathsf{Plan} \Rightarrow \Diamond\gamma$$

und zunächst einem progressiven Vorgehen aus. Assoziiert sei im Planungsprozeß die folgende Modellcharakterisierung:

$$\left\langle \boxed{\begin{matrix} pre \\ ? \end{matrix}}_{i} \cdots \boxed{\begin{matrix} \gamma \\ ? \end{matrix}}_{j} \right\rangle \quad i \leq j$$

Die Formel ϕ beschreibe die oben erwähnte kritische Bedingung, die nicht aus pre unmittelbar oder mittelbar herleitbar ist, aber dennoch notwendig ist, um das spezifizierte Problem zu lösen. Das Intervall kann nun durch den folgenden Nichtdeterminismus verfeinert werden:

$$\left\{ \left\langle \boxed{\begin{matrix} pre \wedge \phi \\ ? \end{matrix}}_{i_1} \cdots \boxed{\begin{matrix} \gamma \\ ? \end{matrix}}_{j_1} \right\rangle, \left\langle \boxed{\begin{matrix} pre \wedge \neg\phi \\ ? \end{matrix}}_{i_2} \cdots \boxed{\begin{matrix} \gamma \\ ? \end{matrix}}_{j_2} \right\rangle \right\}$$

Dies entspricht der Sequenz

$$pre, [\phi \wedge \mathsf{Plan}_1] \vee [\neg\phi \wedge \mathsf{Plan}_2] \Rightarrow \Diamond\gamma$$

bzw. unter Verwendung typischer Schreibweisen für konditionale Pläne

$$pre,\ \underline{if}\ \phi\ \underline{then}\ \mathsf{Plan}_1\ \underline{else}\ \mathsf{Plan}_2\ \Rightarrow \Diamond\gamma\ .$$

Für den Fall, daß die disjunktive Verfeinerung progressiv durchgeführt wird, gilt bzgl. der Spezifikation und Behandlung von Verschmelzungspunkten das gleiche, was im vorherigen Abschnitt für allgemeine nichtdeterministische Pläne gesagt wurde.
Wird die Verfeinerung regressiv durchgeführt, so tritt durch die vollständige Fallunterscheidung ein Spezialfall auf. Dieser erzwingt eine Disjunktion von sich *ausschließenden* Alternativen. Dadurch entfällt die Option, beide Alternativen gleichzeitig regressiv zu

verfolgen. Weiterhin vereinfacht sich $wp^{\cup}$ auf natürliche Weise und die Auswirkungen der disjunktiven Verfeinerung sind lokal geblieben.
Die Tatsache, daß die Wirkungsrichtungen eines regressiven Vorgehens (auf der Zeitachse rückwärts) und der Einführung eines Konditionals (auf der Zeitachse vorwärts) entgegengerichtet sind, führt bezogen auf ein progressives Vorgehen zu unterschiedlichen Arten der Konfliktbewältigung durch den Einsatz eines Konditionals. Gehen wir davon aus, daß jeweils ein Intervall

$$\sigma = \left\langle \boxed{\begin{matrix} pre \\ ? \end{matrix}}_{0} \ldots \boxed{\begin{matrix} \gamma \\ ? \end{matrix}}_{z} \right\rangle$$

als Ausgangsintervall für das progressive bzw. regressive Vorgehen zu verfeinern ist. Unter Progression werden alternative Verfeinerungen für Teilintervalle erzeugt, die noch keine Aktionsmodelle darstellen. Das Konditional kann diese Verfeinerungen also noch vollständig beeinflussen. Unter Regression besteht bereits eine Verfeinerung für eine der beiden disjunktiven Alternativen bzw. kann auf triviale Weise gewonnen werden. Nehmen wir an, daß aktuell

$$\sigma' = \left\langle \boxed{\begin{matrix} \mathsf{pre} \\ ? \end{matrix}}_{i} \ldots \boxed{\begin{matrix} \gamma' \wedge \phi \\ a_j \end{matrix}}_{j} \right\rangle$$

als Teilintervall von σ verfeinert wird. ϕ sei die Konfliktbedingung, für die ein Konditional eingeführt wird. Es folgt die folgende Verfeinerung:

$$\left\{ \left\langle \boxed{\begin{matrix} \mathsf{pre} \wedge \phi \\ ? \end{matrix}}_{i_1} \ldots \boxed{\begin{matrix} \gamma' \wedge \phi \\ a_j \end{matrix}}_{j_2} \right\rangle, \left\langle \boxed{\begin{matrix} \mathsf{pre} \wedge \neg\phi \\ ? \end{matrix}}_{i_2} \ldots \boxed{\begin{matrix} \gamma' \wedge \phi \\ a_j \end{matrix}}_{j_2} \right\rangle \right\}, \; i_1 \leq j_1, \; i_2 < j_2$$

Sie korrespondiert mit den beiden Sequenzen

$$\begin{array}{lll} (1) & \mathsf{pre}, \; \phi \wedge \mathsf{Plan}_1 & \Rightarrow \Diamond[\gamma' \wedge \phi] \\ (2) & \mathsf{pre}, \; \neg\phi \wedge \mathsf{Plan}_2 & \Rightarrow \Diamond[\gamma' \wedge \phi] \end{array}$$

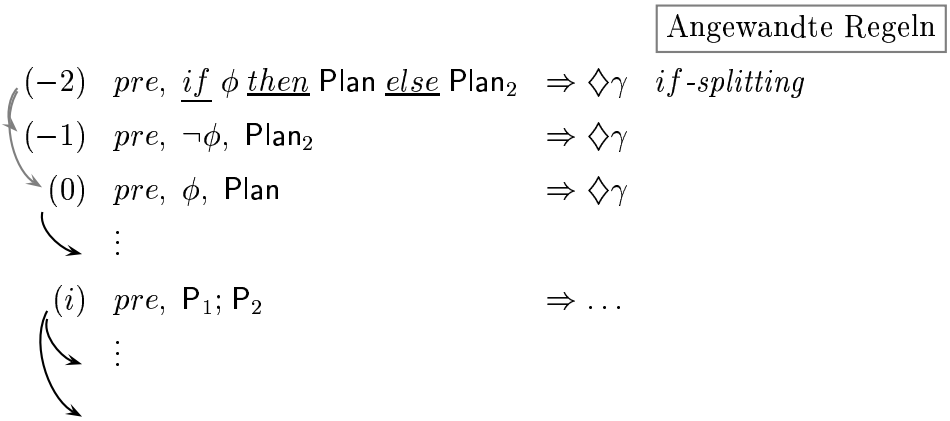
Sequenz (1) kann durch die Instantiierung von Plan_1 mit dem leeren Plan $\circ F$ leicht bewiesen werden; pre könnte mit γ' instantiiert werden. In Sequenz (2) ist das Problem spezifiziert, unter der zusätzlichen Annahme von $\neg\phi$ irgendwann ϕ zu erreichen. Die verwendete Beweisstrategie sollte darauf achten, daß ϕ als Konfliktbedingung nicht erneut zur Einführung eines gleichen Konditionals führt (Beginn einer Endlosschleife!). Wie oben bereits erwähnt, ist es schwierig, die Wirkung eines Konditionals in eine regressive Strategie einzubinden. Es bietet sich für den Beweis von Sequenz (2) daher auch die Verwendung einer progressiven Strategie an. Dieses Vorgehen ist insbesondere zu unterstützen, wenn der Zustand σ_i dem Zustand σ_0 entspricht und γ' aus pre folgt, d.h. das Konditional bildet den Anfang des mit σ assoziierten Planes. Daher kann pre durch pre instantiiert werden und in Sequenz (2) steht maximale Information für ein progressives Vorgehen zur Verfügung.
Die Plazierung der disjunktiven Verfeinerung bezogen auf σ ist folgendermaßen charakterisiert:

$$\left\{ \begin{matrix} \langle\sigma_0 \ldots \sigma_{j_1}\rangle, \\ \langle\sigma_0 \ldots \sigma_{j_2}\rangle \end{matrix} \right\} \langle\sigma_j \; \ldots \; \sigma_z\rangle, \; j_1 = 0, \; 1 \leq j_2$$

Für die durch die Auflösung der Disjunktion resultierenden alternativen Pläne für das Intervall σ, bedeutet dies, daß durch die erste Alternative ein Plan der Länge z entsteht und durch die zweite ein Plan mit mindestens der Länge $z+1$. Man kann das disjunktive Refinement aber auch in einer anderen Variante verwenden:

$$\left\{ \begin{array}{l} \langle \sigma_0 \ \ldots \ \sigma_z \rangle, \\ \langle \sigma_0 \ \ldots \ \sigma_{z_2} \rangle \end{array} \right\}$$

Die erste Alternative entspricht dem Plan, der sich oben auch ergeben hat; die zweite Alternative ist zunächst nur eine Spezifikation entsprechend der initialen Spezifikation von σ, jedoch unter der zusätzlichen Ausgangsbedingung $\neg\phi$. Die Verfeinerung dieser Alternative kann unter Verwendung einer anderen Planungsstrategie auch zu einem Plan mit einer Länge kleiner als z führen; u.U. ist ϕ dann auch keine Konfliktbedingung.
Ein Nachteil der zweiten Variante ist, daß die Etablierung ihrer Struktur eine *Modifikation* der bestehenden Verfeinerungsstruktur notwendig macht. Als erster Verfeinerungsschritt für σ wird eine disjunktive Verfeinerung eingefügt, wobei die erste Alternative den bisher bestehenden Verfeinerungszustand übernimmt.
In der Beweisstruktur spiegelt sich dies wie folgt wieder:



Eine Rechtfertigung für die Verwendung konditionaler Pläne ergab sich bisher aus dem Umstand, daß lediglich *partielle* Information für die Plangenerierung zur Verfügung stand.

Disjunktive Vorbedingungen Eine weitere Rechtfertigung für die Verwendung von konditionalen Plänen ergibt sich, wenn disjunktive Vorbedingungen in Problemspezifikationen berücksichtigt werden, d.h. wenn die Ausgangssituation für die Plangenerierung partielle Information enthält. Diese Unsicherheit kann zum einen als Folge eines disjunktiven Refinements auftreten (vgl. dazu den Abschnitt "Nichtdeterministische Pläne"), zum anderen aber auch schon durch die initiale Planspezifikation vorgegeben sein. Eine Unterscheidung kann nun noch dahingehend getroffen werden, ob die partielle Information allgemeingültige (tautologische) disjunktive Aussagen enthält, z.B. eine Tautologie der Art $\phi \vee \neg\phi$, oder nicht.
Eine Möglichkeit, mit partieller Information umzugehen, ist, sie durch eine nicht notwendigerweise vollständige Fallunterscheidung abzufangen. Sei zur Erläuterung eine Sequenz der Form

$$pre, \ \phi \vee \psi, \ \mathsf{Plan} \Rightarrow \Diamond\gamma$$

gegeben. ϕ und ψ seien modalfreie Formeln, die eine nicht allgemeingültige partielle Aussage über den Ausgangszustand machen. Eine Fallunterscheidung wird wie folgt eingeführt:

			Angewandte Regeln
(1)	$pre,\ \phi \vee \psi,\ \mathsf{Plan}$	$\Rightarrow \Diamond\gamma$	Plan-*links*
(2)	$pre,\ \phi \vee \psi,\ [\mathsf{Plan}_1 \wedge \phi] \vee [\mathsf{Plan}_2 \wedge \psi]$	$\Rightarrow \Diamond\gamma$	$\vee$-*links*
(3)	$pre,\ \phi \vee \psi,\ \mathsf{Plan}_1 \wedge \phi$	$\Rightarrow \Diamond\gamma$	
(4)	$pre,\ \phi \vee \psi,\ \mathsf{Plan}_2 \wedge \psi$	$\Rightarrow \Diamond\gamma$	
	$\vdots$		

In den Sequenzen (3) und (4) ist die partielle Aussage nun redundant. Für die Art der in Sequenz (2) eingeführten Planstruktur kann man auch die folgende Schreibweise verwenden:

$$\underline{if}\ \phi\ \underline{then}\ \mathsf{Plan}_1\ \underline{and}\ \psi\ \underline{then}\ \mathsf{Plan}_2$$

Die rein syntaktische Struktur der bisher diskutierten Pläne wird durch folgende Grammatik zusammengefaßt:[60]

```
Plan   ::=  S_Plan ; ○F  |
            ○F ; S_Plan |
            Plan ∨ Plan |
            ○F
S_Plan ::=  S_Plan ; S_Plan |
            S_Plan ∨ S_Plan |
            modalfreie LLP-Formel ∧ S_Plan |
            ex(Aktionsterm) ∧ ⊙○F |
            ○F
```

4.2.7.4 Pläne mit Sicherheitsbedingungen

Die Aktionsintervalle, wie sie bisher durch die vorgestellten Pläne charakterisiert werden, haben eine fixe Länge, die unmittelbar durch Aktionssequenzen explizit bestimmt ist; die Pläne sind dicht bezüglich der in ihnen explizit spezifizierten Aktionen. Je nach Verwendungszweck eines Planes kann dies eine unnötige Einschränkung bedeuten: mit dem Plan ist keine Beschreibung redundanten oder opportunistischen Verhaltens möglich, d.h. unkritische Unterbrechungen der explizit geforderten Aktionssequenzen sind nicht vorgesehen.
Daß diese Einschränkung leicht überwunden werden kann, kann an der folgenden Modellcharakterisierung diskutiert werden:

$$\overbrace{\left\langle \begin{array}{|c|}\hline pre \\ ? \\ \hline\end{array}_0 \cdots \begin{array}{|c|}\hline pre_a \\ a \\ \hline\end{array}_i \right\rangle}^{\sigma'} \overbrace{\left\langle \begin{array}{|c|}\hline pre_a \\ a \\ \hline\end{array}_i \begin{array}{|c|}\hline goal \\ ? \\ \hline\end{array}_{i+1} \right\rangle}^{\sigma''} \quad 0 \leq i$$

[60] Alternative Schreibweisen konditionaler Pläne sind hier nicht berücksichtigt.

Diese Verfeinerung sei der aktuelle Zustand im Planungsprozeß der Spezifikation

$$pre,\ \mathsf{Plan} \Rightarrow \Diamond goal\ .$$

Nehmen wir an, daß die Aktion a etabliert sei und das Ziel *goal* erreicht, pre_a sind die dazu notwendigen Voraussetzungen; σ'' beschreibt somit also ein Aktionsmodell. Nehmen wir weiterhin an, daß pre_a aus pre unmittelbar herleitbar ist; σ' könnte dann also durch eine entsprechende Längeneinführungsverfeinerung auf die Länge 0 reduziert werden – ein dichter Plan wäre entstanden.
σ' kann aber auch durch eine Verfeinerung im Sinne einer *Sicherheitsbedingungseinführung* (vgl. Abschnitt 4.2.4) verfeinert werden. Es resultiert dann z.B.

$$\left\langle \boxed{\begin{matrix} pre \wedge pre_a \\ ? \end{matrix}}_0 \boxed{\begin{matrix} pre_a \\ ? \end{matrix}}_1 \cdots \boxed{\begin{matrix} pre_a \\ ? \end{matrix}}_{i-1} \boxed{\begin{matrix} pre_a \\ a \end{matrix}}_i \right\rangle \left\langle \boxed{\begin{matrix} pre_a \\ a \end{matrix}}_i \boxed{\begin{matrix} goal \\ ? \end{matrix}}_{i+1} \right\rangle \ 0 \leq i.$$

pre_a ist eine *strenge* Sicherheitsbedingung an das Intervall σ', es wird gefordert, daß pre_a in jedem Zustand von σ' gilt. Um die Modellexistenz der ursprünglichen Planspezifikation zu sichern, muß σ' ein Aktionsintervall sein. Dies ist bereits gewährleistet, wenn die Länge von σ' auf 0 gesetzt wird und die Beziehung $pre \rightarrow pre_a$ gilt. Im allgemeinen existieren zu σ' sehr viele, u.U. unendlich viele, Aktionsmodelle. Der korrespondierende Plan zu σ' ist durch die Formel $\Box pre_a \wedge \Diamond \bigcirc F$ gegeben. Die Charakterisierung eines Planes geschieht also ausschließlich unter Verwendung von Information aus dem assoziierten Beschreibungsintervall (vgl. auch Abschnitt 4.2.3). Die Regel *strong safety* übernimmt die Einführung der entsprechenden Planstruktur:

Regel 9 (strong safety)

$$\frac{\phi,\ \Box\mathsf{pre} \wedge \Diamond\bigcirc F \Rightarrow \Diamond[\bigcirc F \wedge \mathsf{pre}] \qquad \phi \Rightarrow \mathsf{pre}}{\phi,\ \mathsf{P} \Rightarrow \Diamond[\bigcirc F \wedge \mathsf{pre}]}$$

P *eine Planmetavariable,* pre *eine modalfreie LLP*-Formel.*

□

Die Rechtfertigung dieser Regel ergibt sich unmittelbar aufgrund der Semantik des $\Box-$Operators aus der temporallogisch geltenden Folgerung

$$[\Box\mathsf{pre} \wedge \Diamond\bigcirc F] \rightarrow [\Box\mathsf{pre} \wedge \Diamond[\bigcirc F \wedge \mathsf{pre}]].$$

Die in der Regel zusätzlich geforderte Voraussetzung $\phi \Rightarrow \mathsf{pre}$ ist beweistechnisch nicht notwendig, jedoch aus planungsspezifischem Gesichtspunkt durchaus; sind durch die Formel ϕ die Vorbedingungen eines Planes bezeichnet, so wird durch die Beziehung $\phi \Rightarrow \mathsf{pre}$ sichergestellt, daß der Plan mit den verfügbaren Vorbedingungen einerseits verträglich ist und sie andererseits auch in Teilen für einen nachfolgenden Plan verfügbar macht.
Abbildung 4.26 skizziert die beweistechnische Durchführung der beschriebenen Verfeinerung. Unter der Annahme, daß Sequenz (3) durch eine regressive Strategie bewiesen wird, ergeben sich schwächste Vorbedingungen pre_a für die Erreichung von *goal* mit Hilfe der Aktion a. Um zu verifizieren, daß auch zu P_1 ein assoziiertes Aktionsintervall existiert, muß Sequenz (12) bewiesen werden. Damit erhält man die Sicherheit, daß zumindest ein

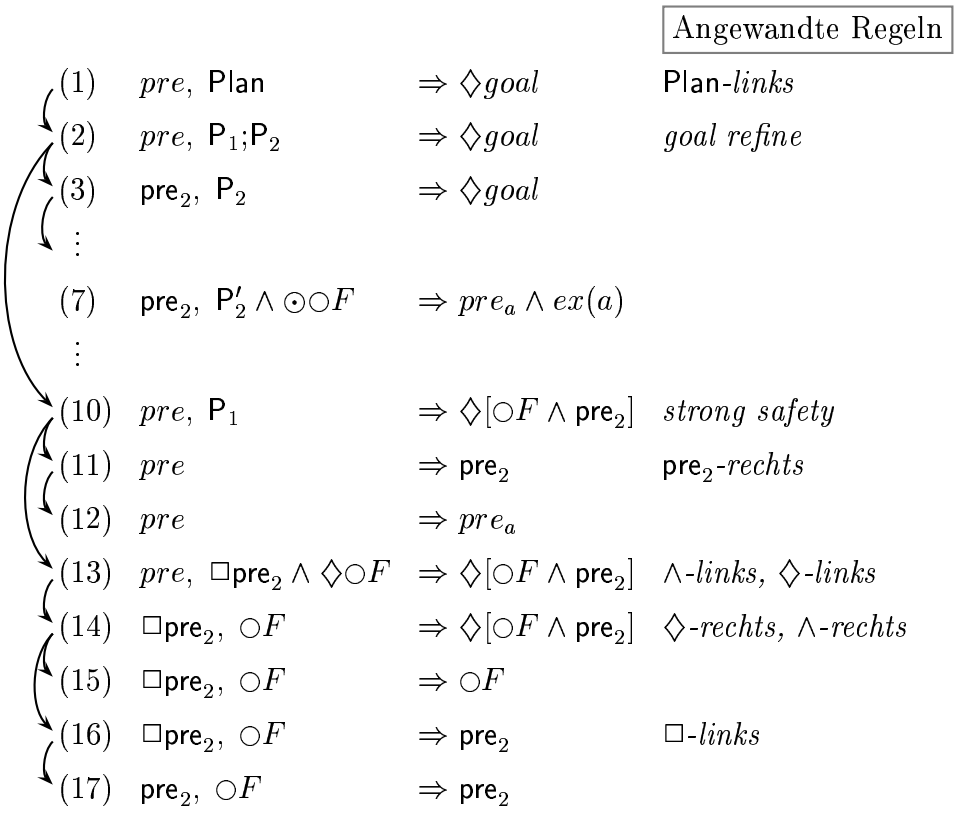


Abbildung 4.26: Verfeinerung durch Sicherheitsbedingungseinführung

Aktionsintervall der Länge 0 existiert. An alle potentiell existierenden weiteren Aktionsintervalle wird eine strenge Anforderung gestellt: alle ihre Zustände müssen konsistent mit pre_a sein, d.h. die Vorbedingung für die Ausführung der Aktion a ist jederzeit gegeben. Interpretiert man das Intervall σ' als eine *Unterbrechung* des Planes P, der mit σ'' assoziiert ist, so heißt dies, daß das unterbrechende Verhalten jederzeit abgebrochen und P sofort ausgeführt werden kann. Die Sicherheitsbedingungseinführung kann aber auch abgeschwächt werden, wie dies die Regel *weak safety* ausdrückt.

Regel 10 (weak safety)

$$\frac{\phi,\ \Box[\bigcirc F \rightarrow \mathsf{pre}] \wedge \Diamond\bigcirc F \Rightarrow \Diamond[\bigcirc F \wedge \mathsf{pre}] \qquad \phi \Rightarrow \mathsf{pre}}{\phi,\ \mathsf{P} \Rightarrow \Diamond[\bigcirc F \wedge \mathsf{pre}]}$$

P *eine Planmetavariable,* pre *eine modalfreie LLP*-Formel.*

□

Es resultiert dann für das obige Beispiel folgende Modellcharakterisierung:

$$\left\langle \boxed{\begin{matrix} pre \wedge pre_a \\ ? \end{matrix}}_0 \boxed{\begin{matrix} ? \\ ? \end{matrix}}_1 \cdots \boxed{\begin{matrix} ? \\ ? \end{matrix}}_{i-1} \boxed{\begin{matrix} pre_a \\ a \end{matrix}}_i \right\rangle \left\langle \boxed{\begin{matrix} pre_a \\ a \end{matrix}}_i \boxed{\begin{matrix} goal \\ ? \end{matrix}}_{i+1} \right\rangle\ 0 \leq i.$$

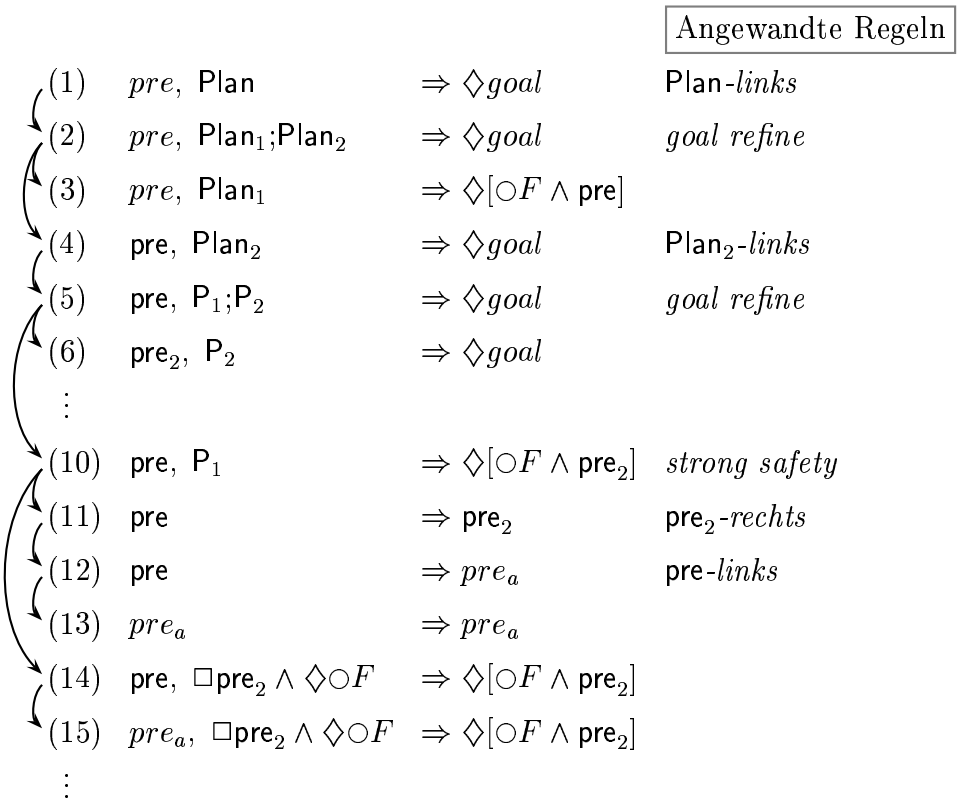


Abbildung 4.27: Kontinuierliche Sicherheitsbedingungseinführung

Es wird nur noch gefordert, daß pre_a zu Beginn und am Ende der Unterbrechung gilt. Dies bedeutet, daß das unterbrechende Verhalten dafür sorgen muß, daß, wenn pre_a in einem Zwischenzustand nicht mehr gilt, es im letzten Zustand aber wiederhergestellt sein muß. Andererseits bedeutet dies aber auch, daß das unterbrechende Verhalten im allgemeinen nicht jederzeit abgebrochen werden kann, sondern ein spezifisches Ende abgewartet werden muß. Möchte man Sicherheitsbedingungen vor der Etablierung jeder einzelnen Aktion eines linearen Planes einführen, so muß die Vorgehensweise aus Abbildung 4.26 etwas modifiziert werden; Abbildung 4.27 zeigt einen entsprechenden Ausschnitt.
Der weitere Beweis von Sequenz (6) erzeugt hier die Instantiierung für die Metavariable pre_2, die bestimmt, welche Sicherheitsbedingung eingeführt wird. Die Spezifikation für den Plan assoziiert mit der Metavariable P_1 ist so gestaltet, daß sie auch durch den leeren Plan direkt erfüllbar gemacht werden könnte. Dieser Umstand spiegelt nochmals wider, daß die Einführung von Sicherheitsbedingungen rein optional ist. Wird der Gesamtplan nun weiter verfeinert, so wird der Teilplan, assoziiert mit Plan_1, ebenso bearbeitet, wie es beginnend mit Sequenz (1) dargestellt wurde.
Die Regeln 9 und 10 charakterisieren die Basisformen von durch Sicherheitsbedingungen angereicherten, zeitlich abstrakten Plänen; die durch Regel 10 eingeführte schwache Unterbrechbarkeit ist dabei notwendiger Bestandteil aller Formen zeitlich abstrakter Pläne. Eine mögliche Form der weiteren Verfeinerung bildet die Etablierung von Zeitbeschränkun-

gen durch entsprechende Längeneinführungsverfeinerungen. Neben der Festlegung einer fixen Länge durch eine Formel $\odot^i \bigcirc F$ ist ebenso die Festlegung einer maximalen Länge – die Formel $[\neg \Diamond \odot^i \bigcirc F]; \bigcirc F$ schränkt die Länge eines Intervalles auf maximal $i-1$ ein – beziehungsweise einer minimalen Länge möglich – die Formel $\odot^i \bigcirc F; T$ erzwingt mindestens die Intervallänge i.

Neben der rein temporalen Verfeinerung sind auch erweiterte Formen von Sicherheitsbedingungen möglich. Die bisher nur implizite Berücksichtigung von Aktionsintervallinformationen kann zu einer expliziten erweitert werden. Ein Zweck der bisher verwendeten Beschreibungsintervallinformation in der Formulierung einer starken Sicherheitsbedingung ist, die Ausführung von Aktionen, die diese Bedingung verletzen, zu verhindern. Den gleichen Zweck erfüllt damit eine Sicherheitsbedingung, die alle störenden Aktionen explizit ausschließt. Zu der bereits oben verwendeten Bedingung pre_a seien $d_1, \ldots, d_k$ alle Aktionen, die folgende Anforderung erfüllen:

$$\forall i,\ 1 \leq i \leq k:\ pre_a \rightarrow pre_{d_i},\quad \mathit{eff}_{d_i} \rightarrow \neg\phi \text{ mit } pre_a \rightarrow \phi$$

Das heißt, die Vorbedingungen einer Aktion d_i sind aus pre_a ableitbar und der Effekt von d_i führt unter diesen Bedingungen zusammen mit pre_a zu Inkonsistenz.
Die Formulierung einer starken Sicherheitsbedingung kann dann auch durch den Plan

$$\Box \bigvee_{i=1}^{k} \neg ex(d_i) \quad \wedge \Diamond \bigcirc F$$

geschehen.

Eine Variante in der Verwendung schwacher Sicherheitsbedingungen besteht darin, die dort erlaubte zwischenzeitliche Verletzung der Sicherheitsbedingung dadurch abzuschwächen, indem ein *Reparaturplan* angegeben wird, der die Bedingung wieder herstellen kann. Die folgende Regel beschreibt ein entsprechendes Schema:

Regel 11 (weak safety repair)

$$\frac{\begin{array}{c}\phi,\ \Box[\bigcirc F \rightarrow \mathsf{pre}] \wedge \Box[[\neg \mathsf{pre} \rightarrow \mathsf{Q}; T] \vee T] \wedge \Diamond \bigcirc F \Rightarrow \Diamond[\bigcirc F \wedge \mathsf{pre}] \\ \neg \mathsf{pre}, \mathsf{Q} \Rightarrow \Diamond \mathsf{pre} \qquad \phi \Rightarrow \mathsf{pre}\end{array}}{\phi,\ \mathsf{P} \Rightarrow \Diamond[\bigcirc F \wedge \mathsf{pre}]}$$

P, Q *Planmetavariablen,* pre *eine modalfreie LLP-Formel.*

□

Hierin wird dem den Plan P unterbrechenden Verhalten die Option gewährt, eine Verletzung der Bedingung pre durch den Plan Q zu beseitigen.[61] Die Verwendung des Planes Q bzw. überhaupt eines Reparaturplanes steht dabei jedoch frei; Q stellt lediglich einen Vorschlag dar, die verletzte Bedingung zu restaurieren.
Je mehr die Basisformen der Sicherheitsbedingungseinführung erweitert werden, um so mehr spezialisiert man den entstehenden Plan auf einen bestimmten Verwendungszweck.

Die auf Seite 169 beschriebene Grammatik syntaktisch zulässiger Pläne erweitert sich bei der Berücksichtigung von zeitlicher Abstraktion durch die folgenden Regeln:

[61] Der Verbindung von Q mit der Formel T (*true*) sichert, daß die Planunterbrechung nach Wiederherstellung der verletzten Bedingung nicht beendet ist.

$$
\begin{array}{rcl}
\texttt{S\_Plan} & ::= & \texttt{Strong} \wedge \Diamond\bigcirc F \mid \\
 & & \texttt{Weak} \wedge \Diamond\bigcirc F \\
\texttt{Strong} & ::= & \Box \texttt{ modalfreie LLP-Formel} \mid \\
 & & \texttt{Strong} \wedge \texttt{Length} \\
\texttt{Weak} & ::= & \Box[\bigcirc F \rightarrow \texttt{ modalfreie LLP-Formel }] \mid \\
 & & \texttt{Weak} \wedge \texttt{Length} \mid \\
 & & \texttt{Weak} \wedge \texttt{Repair} \\
\texttt{Length} & ::= & \odot^i \bigcirc F \mid \\
 & & [\neg\Diamond \odot^i \bigcirc F];\bigcirc F \mid \\
 & & \odot^i \bigcirc F;T \\
\texttt{Repair} & ::= & \Box[[\texttt{ modalfreie LLP-Formel} \rightarrow \texttt{S\_Plan} \,;T] \vee T]
\end{array}
$$

Notwendige Sicherheitsbedingungen Die Art und Weise, Sicherheitsbedingungen zu verwenden, wird bisher als zusätzliches Merkmal eines Planes angesehen, ohne jedoch dessen essentiellen Aktionsanteil zu beeinflussen. Die Sicherheitsbedingungen geben an, wie ein Plan unterbrochen werden *kann*, um ein nicht konkret vorbestimmtes Zusatzverhalten abzudecken. Pläne mit zeitlicher Abstraktion bieten z.B. interpretiert als Handlungsanweisungen die Möglichkeit zum *opportunistischen* Handeln. Sie enthalten einen Freiheitsgrad zum Einschieben von neuen Plänen bzw. Aktionen, solange diese nicht bestimmte Bedingungen verletzen.
Sicherheitsbedingungen können aber auch aktiv die Entstehung eines Planes beeinflußen. Entweder, indem sie bereits Teil der Planspezifikation sind, also von Außen vorgegeben werden, oder, indem ganz allgemein einer Sicherheitsbedingungseinführung durch den Planungsprozeß notwendige aktionsspezifische Verfeinerungen folgen. Hier legen sie dann fest, welche inhaltlichen Zusammenhänge ein entstehender Plan einhalten *muß*. In beiden Fällen resultiert dann etwa eine Planspezifikation der folgenden Art (für den Fall einer starken Sicherheitsbedingung):

$$pre, \Box\phi \wedge \mathsf{Plan} \Rightarrow \Diamond goal$$

Die Sicherheitsbedingung ϕ muß in diesem Fall in jedem durch Plan näher charakterisierten Zustand gelten. Das heißt nun für die weitere Verfeinerung der Spezifikation, daß insbesondere eine folgende Aufsplittungsverfeinerung diesem Aspekt gerecht werden muß. Um eine Aufsplittung durchführen zu können, muß die Bedingung ϕ also zunächst auf alle Zustände verteilt werden.

			Angewandte Regeln
(1)	$pre,\ \Box\phi \wedge \mathsf{Plan}$	$\Rightarrow \Diamond goal$	Plan-*links*
(2)	$pre,\ \Box\phi \wedge \mathsf{P}_1;\mathsf{P}_2$	$\Rightarrow \Diamond goal$	$\Box$;-*distribution*
(3)	$pre,\ [\Box\phi \wedge \mathsf{P}_1];[\Box\phi \wedge \mathsf{P}_2]$	$\Rightarrow \Diamond goal$	*goal refine*
(4)	$pre,\ \Box\phi \wedge \mathsf{P}_1$	$\Rightarrow \Diamond[\bigcirc F \wedge \psi]$	
(5)	$\psi,\ \Box\phi \wedge \mathsf{P}_2$	$\Rightarrow \Diamond goal$	

Dies erreicht man durch die Ausnutzung temporallogischer Theoreme, wie es der Übergang von Sequenz (2) zu Sequenz (3) z.B. widerspiegelt. Aus planungstechnischer Sicht besteht das Problem beim Umgang mit Sequenzen wie (4) oder (5) darin, die Modellexistenz

unter einer zusätzlichen Randbedingung zu sichern. Die Sicherung der Modellexistenz übernimmt, wie bei der Generierung sequentieller Pläne gesehen (vgl. Abschnitt 4.2.7.1), im wesentlichen der Mechanismus zur Generierung von Aktionsaxiomen. Sehen wir uns dazu den Fall des durch eine explizite Aktion bedingten Zustandsübergangs im Detail sowohl für die progressive als auch für die regressive Vorgehensweise an.
Bei der progressiven Verfeinerung erhalten wir eine Sequenz der folgenden Art:

$$(k)\ pre,\ \Box\phi,\ \mathsf{P}_x \wedge \odot\bigcirc F \Rightarrow \bigcirc[\gamma \wedge \mathsf{pre}]$$

Nehmen wir an, daß die Aktion a ausgehend von pre und ϕ das Ziel γ erreichen kann. Die Modellexistenz ist aber nur dann gesichert, wenn ϕ invariant ist bzgl. der Aktion a, d.h. ϕ muß auch noch unverändert nach Ausführung von a gelten. Erreichen kann man dies, indem man den Beweis von Sequenz (k) auf den Beweis der spezialisierten Sequenz (k') zurückführt.

$$(k')\ pre,\ \phi,\ \mathsf{P}_x \wedge \odot\bigcirc F \Rightarrow \bigcirc[\gamma \wedge \mathsf{pre} \wedge \phi]$$

Es ergibt sich beim Beweis dann u.a. die Sequenz

$$(r)\ pre,\ \phi,\ ex(a) \wedge \odot\bigcirc F \Rightarrow \bigcirc\phi$$

Um sie abzuschließen bildet man regressiv eine Axiomeninstanz von a bzgl. der Bedingung ϕ und beweist dann, ob die resultierenden schwächsten Vorbedingungen aus den Voraussetzungen in (r) ableitbar sind.
Wird die Aktion durch Regression etabliert, steht eine Sequenz der Art

$$(l)\ \mathsf{pre},\ \Box\phi,\ \mathsf{P}_x \wedge \odot\bigcirc F \Rightarrow \bigcirc\gamma$$

zur Beweisführung an. Zur Sicherung der Modellexistenz muß auch hier die Invarianz von ϕ bzgl. a gezeigt werden. Die Situation ist hier jedoch etwas schwieriger als im progressiven Fall, da nicht bekannt ist, von welchen Voraussetzungen insgesamt ausgegangen werden kann. Betrachten wir dazu die unterschiedlichen Möglichkeiten, wie ϕ mit der Axiomatisierung der Aktion a in Beziehung stehen kann; wp_ϕ (wp_γ) seien die schwächsten Vorbedingungen von ϕ (γ) bzgl. a.

1. ϕ verhält sich vollkommen interferenzfrei zur Aktion a, d.h. es gilt

   $$wp_\phi \leftrightarrow [wp_\gamma \wedge \phi]$$

   und wp_ϕ ist widerspruchsfrei.

2. ϕ verhält sich nur dann interferenzfrei, wenn eine zusätzliche Bedingung π gilt, d.h. es gilt

   $$wp_\phi \leftrightarrow [wp_\gamma \wedge \phi \wedge \pi]$$

   und wp_ϕ ist widerspruchsfrei.

3. ϕ erscheint (partiell) als Effekt der Aktion a, d.h. es gilt nun nicht mehr $wp_\phi \rightarrow \phi$, sondern nur noch allgemein

   $$wp_\phi \leftrightarrow [wp_\gamma \wedge \pi]$$

Für den Fall, daß ϕ ein primärer Effekt der Aktion a ist, ergibt sich $\pi \leftrightarrow T$. Was nun zu überprüfen ist, ist die Widerspruchsfreiheit von $wp_\phi \wedge \phi$, die sichert, daß diese (partielle) Erzeugung von ϕ auch unter der Zusatzbedingung ϕ gilt[62]. Wenn man die Widerspruchsfreiheit nicht sichert, so führt man den Gesamtbeweis fort mit dem Ziel, einen Widerspruch herzuleiten, was nur gelingt, wenn die Anfangsbedingungen der Planspezifikation bereits widersprüchlich sind.

4. Ist wp_ϕ nicht widerspruchsfrei, so sollte der Beweis von Sequenz (l) unter Verwendung der Aktion a abgebrochen werden.

Zeigt man die Invarianz während des Beweises nur gemäß den Punkten 1. und 2. so genügt es, sich auf den Mechanismus zur Erzeugung effektiver Aktionsaxiome zu beschränken. Zur Sicherung der Sequenz (l) genügt es dann, von der spezialisierten Sequenz (l') auszugehen:

$$(l')\ \mathsf{pre},\ \mathsf{P}_x \wedge \odot\bigcirc F \Rightarrow \phi \wedge \bigcirc[\gamma \wedge \phi]$$

Ihr erfolgreicher Beweis sichert die Invarianz von ϕ bzgl. der das Ziel γ erreichenden Aktion. Sei dazu a eine Aktion, die γ erreichen kann; $pre_{a\phi}$ als Instantiierung von pre seien die notwendigen Vorbedingungen von a, um γ und ϕ zu erreichen. Wenn aus $pre_{a\phi}$ wiederum ϕ ableitbar ist, bedeutet dies, daß die Invarianz von ϕ bzgl. a gesichert ist.

Läßt man auch zusätzlich den oben beschriebenen Punkt 3. zu, so ist zur Erreichung eines effektiven Vorgehens ein zusätzlicher Beweis von Widerspruchsfreiheit notwendig, d.h. man beweist dann

$$(l'')\ \mathsf{pre},\ \mathsf{P}_x \wedge \odot\bigcirc F \Rightarrow \bigcirc[\gamma \wedge \phi]$$

und

$$(l''')\ wp_\phi \wedge \phi \qquad \textit{widerspruchsfrei.}$$

Durch die Entscheidung, innerhalb des Planungsprozesses z.B. nur das Verfahren gemäß Sequenz (l') zu verwenden, kann man eine spezielle Ansteuerung des Planers erreichen. Dieser vermeidet dadurch, Aktionen zu verwenden, die die Sicherheitsbedingungen explizit als Effekte etablieren.
Präferiert man dagegen das kombinierte Verfahren gemäß (l''), (l''') und dem Fehlschlagen von (l') gegenüber der reinen Verwendung von (l') so werden in dem entstehenden Plan vorallem Aktionen verwendet, die das Gelten der Sicherheitsbedingungen *verstärken*, d.h. die Pläne sind in ihrer späteren Ausführung robuster gegenüber durch Ereignisse hervorgerufene Verletzungen der Sicherheitsbedingungen.

Durch die Verwendung von notwendigen Sicherheitsbedingungen in einer Planspezifikation erreicht man, daß nur Verfeinerungen erzeugt werden, die diese Bedingungen auch erfüllen, d.h. durch die Einführung der Sicherheitsbedingungen wird eine *aktive Suchraumkontrolle* sowohl bei regressivem als auch (sogar verstärkt) bei progressivem Vorgehen ermöglicht bzw. erzwungen. Man kann also auf diese Weise insbesondere domänenspezifisches, aktuell relevantes Wissen in den Planverfeinerungsprozeß integrieren. Löst man sich von der Annahme, daß notwendige Sicherheitsbedingungen jeweils nur zustandsbasierte Aussagen beschreiben und berücksichtigt auch ausgedehnte Intervallaussagen, so können durch die

[62]Dies gilt natürlich trivialerweise, wenn $wp_\phi \wedge \phi$ widersprüchlich also logisch *falsch* ist; solch eine Voraussetzung ist jedoch wenig effektiv.

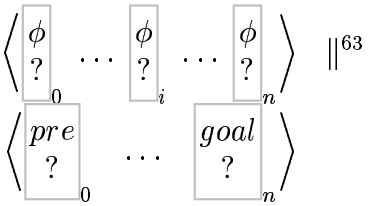
dann beschreibbaren Zustandszusammenhänge komplexere Sicherheitsbedingungen formuliert und insbesondere stärkere Suchraumbeschränkungen während der Planverfeinerung möglich werden. Zeitliche Strukturen in Sicherheitsbedingungen erfordern jedoch auch einen zusätzlichen Aufwand bei der Evaluierung dieser Bedingungen; im wesentlichen erhöht sich der Aufwand proportional zur temporalen Struktur der Sicherheitsbedingungen, d.h. der Intervallänge, die mindestens notwendig ist, um eine Sicherheitsbedingung vollständig evaluieren zu können (vgl. auch Abschnitt 4.2.2).

4.2.7.5 Nichtlineare Pläne

Analysiert man die im letzten Abschnitt eingeführte Generierung von Plänen unter Berücksichtigung aktiver Sicherheitsbedingungen detaillierter, so kann man zu folgender Beobachtung kommen: es findet eine *nebenläufige Verfeinerung* statt. Die korrespondierende Intervallstruktur für eine Sequenz

$$pre, \Box\phi \ \wedge \mathsf{Plan} \Rightarrow \Diamond goal$$

ist gegeben durch



Ein zur Erfüllung der Planspezifikation relevantes Modell muß nun beiden Verfeinerungen genügen. Um den Nachweis der Modellexistenz zu erbringen, bedeutet dies insbesondere, daß die Interferenzfreiheit der Verfeinerungen garantiert werden muß. In dem obigen Fall heißt das, daß die Invarianz der Bedingung ϕ gegenüber allen vorgenommenen Verfeinerungen im zweiten Intervall bewiesen werden muß. Diese Notwendigkeit resultiert dann in der spezialisierten Aktionsetablierung, wie sie im letzten Abschnitt beschrieben wurde. Eine *Nebenläufigkeitsverfeinerung* kann also nur dann erfolgreich eingeführt werden, wenn alle dadurch induzierten Randbedingungen zur Sicherung der Interferenzfreiheit eingehalten werden können.

Die oben verwendete Nebenläufigkeitsverfeinerung ist strukturell einfach, da nur eines der Intervalle durch Aktionsetablierungen verfeinert wird. Das zweite Intervall hat lediglich die Aufgabe, diese Etablierung zu steuern, indem es Vorgaben macht. Unter Ausnutzung des Konzeptes unterbrechbarer Pläne kann man nun auch eine Nebenläufigkeitsverfeinerung einführen, die in beiden Intervallen weitere aktionsspezifische Verfeinerungen vornimmt. Somit beeinflussen sich konkurrierende Intervalle einer Nebenläufigkeitsverfeinerung wechselseitig. Man nutzt dies aus, um die Basis für die Repräsentation und die Beweisführung *nichtlinearer* Pläne – auch *partiell geordnete* Pläne genannt – zu schaffen. Nichtlineare Pläne bilden die Abstraktion einer Menge von linearen Plänen, die sich durch permutierte Anordnung der beteiligten Aktionen voneinander unterscheiden. Die Reihenfolge bestimmter Aktionen ist dabei unerheblich, da sie sich gegenseitig nicht stören.

[63] Wir verwenden dieses Zeichen in Zukunft zur Kennzeichnung nebenläufiger Verfeinerungen.

Als Beispiel gehen wir von folgender Planspezifikation aus:

$$pre,\ \mathsf{Plan} \Rightarrow \Diamond[\gamma_1 \wedge \gamma_2]$$

Hierin seien γ_1 und γ_2 modalfreie Formeln. Weiterhin nehmen wir an, daß zwei interferenzfreie Aktionen a und b existieren, wobei a das Teilziel γ_1 und b das Ziel γ_2 erreichen kann. Dieser Umstand kann z.B. durch folgende Nebenläufigkeitsverfeinerung in seiner Allgemeinheit charakterisiert werden:

$$\left\langle \boxed{\begin{matrix} pre \wedge pre_a \\ ? \end{matrix}}_0 \boxed{\begin{matrix} pre_a \\ ? \end{matrix}}_1 \cdots \boxed{\begin{matrix} pre_a \\ a \end{matrix}}_i \boxed{\begin{matrix} \gamma_1 \\ ? \end{matrix}}_{i+1} \cdots \boxed{\begin{matrix} \gamma_1 \\ ? \end{matrix}}_z \right\rangle \ \|$$
$$\left\langle \boxed{\begin{matrix} pre \wedge pre_b \\ ? \end{matrix}}_0 \boxed{\begin{matrix} pre_b \\ ? \end{matrix}}_1 \cdots \boxed{\begin{matrix} pre_b \\ b \end{matrix}}_j \boxed{\begin{matrix} \gamma_2 \\ ? \end{matrix}}_{j+1} \cdots \boxed{\begin{matrix} \gamma_2 \\ ? \end{matrix}}_z \right\rangle \ i \neq j,\ 2 \leq z.$$

Diese Verwendung konkurrierender unterbrechbarer Pläne legt keine Reihenfolge zwischen den beiden Aktionen fest; daß sie nicht gleichzeitig auftreten (sofern sie voneinander verschieden sind), verbietet die Semantik von LLP (vgl. dazu Abschnitt 4.1.2). Weiterhin ist sichergestellt, daß die Vorbedingungen für jede Aktion zu ihrem Ausführungszeitpunkt gelten und daß deren Ziele erhalten bleiben.
Die Interferenzfreiheit zwischen zwei Aktionen ist folgendermaßen definiert:

Definition 4.2.7 *Sei γ (δ) ein Ziel, das durch die Aktion α (β) unter der notwendigen Voraussetzung pre_α (pre_β) erreicht werden kann; neben den unmittelbaren Effekten der Aktion kann hierin auch ein Effekt enthalten sein, der nur mittelbar durch die Aktion erreicht wird (indem er von ihr nicht zerstört wird). Bezüglich der beiden Aktionen gelte folgendes:*

- *die Vorbedingungen und das Ziel von α sind invariant bzgl. der spezifischen Aktion β, d.h. es gilt:*
  (1) $\mathsf{pre}_1,\ ex(\beta) \wedge \odot\bigcirc F \Rightarrow \mathsf{vb}^1_{\alpha\beta} \wedge pre_\alpha \wedge \bigcirc[\delta \wedge pre_\alpha]$
  (2) $\mathsf{pre}_2,\ ex(\beta) \wedge \odot\bigcirc F \Rightarrow \mathsf{vb}^2_{\alpha\beta} \wedge \gamma \wedge \bigcirc[\delta \wedge \gamma]$
- *die Vorbedingungen und das Ziel von β sind invariant bzgl. der spezifischen Aktion α:*
  (3) $\mathsf{pre}_3,\ ex(\alpha) \wedge \odot\bigcirc F \Rightarrow \mathsf{vb}^3_{\alpha\beta} \wedge pre_\beta \wedge \bigcirc[\gamma \wedge pre_\beta]$
  (4) $\mathsf{pre}_4,\ ex(\alpha) \wedge \odot\bigcirc F \Rightarrow \mathsf{vb}^4_{\alpha\beta} \wedge \delta \wedge \bigcirc[\gamma \wedge \delta]$

Gilt $\mathsf{vb}^i_{\alpha\beta} \not\equiv T$ für ein i, so sind die beiden Aktionen α und β schwach interferenzfrei, sonst (vollständig) interferenzfrei.

Die schwächere Variante der Interferenzfreiheit ist dadurch gekennzeichnet, daß die Interferenzfreiheit nur unter zusätzlichen Bedingungen an Variablenbindungen garantiert ist; die beiden Aktionen schränken ihre Verwendung also gegenseitig ein.
Am Beispiel der Invarianzbedingung (4) betrachten wir die Strategie, mit der sie erfüllt werden kann:

			Angewandte Regeln
(a)	$\mathsf{pre}_4,\ ex(\alpha) \wedge \odot\bigcirc F$	$\Rightarrow \mathsf{vb}^4_{\alpha\beta} \wedge \delta \wedge \bigcirc[\gamma \wedge \delta]$	$\wedge$-*rechts*
(b)	$\mathsf{pre}_4,\ ex(\alpha) \wedge \odot\bigcirc F$	$\Rightarrow \mathsf{vb}^4_{\alpha\beta} \wedge \delta$	pre_4-*links*
(c)	$\mathsf{pre}_4,\ ex(\alpha) \wedge \odot\bigcirc F$	$\Rightarrow \bigcirc[\gamma \wedge \delta]$	
(d)	$\mathsf{pre}_4,\ ex(\alpha) \wedge \odot\bigcirc F$	$\Rightarrow pre_{\alpha\delta} \wedge ex(\alpha)$	
(e)	$pre_{\alpha\delta},\ ex(\alpha) \wedge \odot\bigcirc F$	$\Rightarrow \mathsf{vb}^4_{\alpha\beta} \wedge \delta$	

Um Sequenz (c) zu schließen, benötigt man ein entsprechend instantiiertes Rahmenaxiom der Aktion α; die damit assoziierte Regel wurde auf Sequenz (c) angewandt und eine Instantiierung für die Metavariable pre_4 ist verfügbar. Sequenz (b) kann nun damit instantiiert werden und mit dem Beweis von Sequenz (e) stellt sich heraus, ob die Invarianz von δ gegeben ist. Die Metavariable $\mathsf{vb}^4_{\alpha\beta}$ kann dann mit Variablenbedingungen, die möglicherweise in $pre_{\alpha\delta}$ aber nicht in δ enthalten sind, instantiiert werden.
Für das obige Beispiel der beiden nebenläufigen Aktionen a and b ergibt sich unter der Annahme einer schwachen Interferenzfreiheit die folgende Intervallcharakterisierung:

$$\left\langle \begin{array}{ccccccccc} vb_{ab} & vb_{ab} & \cdots & vb_{ab} & \cdots & vb_{ab} \\ ? & ? & & ? & & ? \\ 0 & 1 & & i & & z \end{array} \right\rangle \ \| \ \left\langle \begin{array}{ccccccc} pre \wedge pre_a & pre_a & \cdots & pre_a & \gamma_1 & \cdots & \gamma_1 \\ ? & ? & & a & ? & & ? \\ 0 & 1 & & i & i+1 & & z \end{array} \right\rangle \ \| \ \left\langle \begin{array}{ccccccc} pre \wedge pre_b & pre_b & \cdots & pre_b & \gamma_2 & \cdots & \gamma_2 \\ ? & ? & & b & ? & & ? \\ 0 & 1 & & j & j+1 & & z \end{array} \right\rangle \ i \neq j,\ 2 \leq z.$$

Ein Plan, der diese Intervallstruktur widerspiegelt, kann durch folgende LLP-Formel ausgedrückt werden:[64]

$$[\ [\Box pre_a\ ;\ ex(a) \wedge \odot\bigcirc F\ ;\ \Box\gamma_1] \wedge [\Box pre_b\ ;\ ex(b) \wedge \odot\bigcirc F\ ;\ \Box\gamma_2] \wedge \Box vb_{ab}\ ]\ ;\ \bigcirc F.$$

Statt wie im Beispiel strenge Sicherheitsbedingungen zu verwenden, genügt es auch, sich auf schwache zu beschränken. Ein entsprechender Plan wird ausgedrückt als:

$$\left[\begin{array}{r} [\Box[\bigcirc F \rightarrow pre_a]\ ;\ ex(a) \wedge \odot\bigcirc F\ ;\ \Box[\bigcirc F \rightarrow \gamma_1]] \wedge \\ [\Box[\bigcirc F \rightarrow pre_b]\ ;\ ex(b) \wedge \odot\bigcirc F\ ;\ \Box[\bigcirc F \rightarrow \gamma_2]] \wedge\ \Box vb_{ab} \end{array} \right]\ ;\ \bigcirc F.$$

Diese Struktur beschreibt jetzt nur noch, wo Vorbedingungen und Ziele gelten müssen und welche Aktionen beteiligt sind; unspezifizierte Unterbrechungen des Planes sind aber noch erlaubt. Unter Ausnutzung der temporallogischen Semantik und entsprechender äquivalenzerhaltender Transformationen kann die Planstruktur auch etwas vereinfacht werden zu:

$$\begin{array}{l} [\Diamond[pre_a \wedge ex(a) \wedge \odot\bigcirc F]\ ;\ \Diamond[\gamma_1 \wedge \bigcirc F]] \wedge \\ [\Diamond[pre_b \wedge ex(b) \wedge \odot\bigcirc F]\ ;\ \Diamond[\gamma_2 \wedge \bigcirc F]]\ \ \wedge \Box vb_{ab}. \end{array}$$

[64] Die Sicherheitsbedingungen wurden hier der kompakteren Schreibweise wegen temporallogischen Transformationen unterzogen.

bzw. auch zu

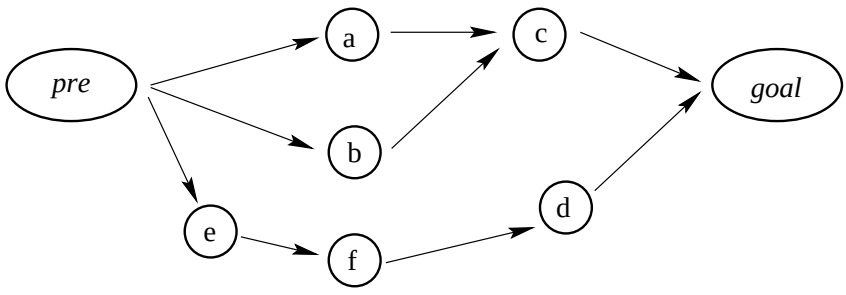
$$T\ ;\ pre_a \wedge ex(a) \wedge \odot\bigcirc F\ ;\ T\ ;\ \gamma_1 \wedge \bigcirc F\ \ \wedge$$
$$T\ ;\ pre_b \wedge ex(b) \wedge \odot\bigcirc F\ ;\ T\ ;\ \gamma_2 \wedge \bigcirc F\ \ \wedge\ \Box vb_{ab}.$$

Um die Unterbrechbarkeit auszuschließen, kann der Plan um eine Längeneinführung erweitert werden, für zwei Aktionen ist die Länge 2 notwendig, also die konjunktive Verknüpfung mit der Formel $\odot^2\bigcirc F$. Besteht nun aber nur ein Interesse an den am Plan beteiligten Aktionen und ihrer Beziehung zur ursprünglichen Planspezifikation, kann auch ein dichter Plan kompakt formuliert werden:

$$[[\Diamond ex(a) \wedge \Diamond ex(b)] \wedge \overbrace{\odot\bigcirc F}^{\boxed{1}}]\ ;\underbrace{\odot\bigcirc F}_{\boxed{2}}$$

Durch die semantische Interpretation des $\Diamond$-Operators im Zusammenhang mit der konjunktiven Verknüpfung wird die Abstraktion von der Reihenfolge der Aktionen ausgedrückt. Die Intervallängenbeschränkung $\boxed{2}$ ist notwendig, damit die beteiligten Aktionen das intendierte Aktionsmodell beschreiben. Die Intervallbeschränkung $\boxed{1}$ stellt die Dichtheit des Intervalles bzgl. der beteiligten Aktionen sicher. Da die kompakte Planbeschreibung keine Beschreibungsintervallinformation mehr enthält, ergibt sich nun jedoch die Notwendigkeit, daß die explizit spezifizierten Aktionen alle voneinander verschieden sein müssen; ansonsten kann die Modellexistenz nicht gesichert werden, da eine unspezifizierte "Lücke" im Plan entstehen kann.

Generierung einfacher nichtlinearer Pläne Nichtlineare Pläne werden typischerweise durch gerichtete, zusammenhängende und zyklenfreie Graphen repräsentiert; z.B. im Plangraph



bezeichnen die Knoten $a - f$ Aktionen, die Knoten *pre* bzw. *goal* bezeichnen die Vorbedingungen bzw. das zu erreichende Ziel; die Kanten beschreiben Bedingungen, die ein Knoten erzeugt bzw. als Vorbedingung benötigt. Entsprechend dem oben beschriebenen Konzept der Nebenläufigkeitsverfeinerung werden nun Regeln beschrieben, mit deren Hilfe Pläne, zusammengesetzt aus Aufsplittungs- und Nebenläufigkeitsverfeinerung generiert werden können. Regel 12 etabliert eine Nebenläufigkeitsstruktur in einem entstehenden Plan. Wie bereits beschrieben, ist es bei nebenläufigen Plänen notwendig zu zeigen, daß die beteiligten Aktionen in der intendierten partiellen Ordnung auch tatsächlich auftreten können, d.h. die Interferenzfreiheit ist in einem bestimmten Umfang zu zeigen.
Betrachten wir zunächst die entstehenden Graphen etwas genauer.

Definition 4.2.8 *Ein Plangraph $PG = (V, E)$ besteht aus einer Menge $V = AK \cup EK$ von Knoten und einer Menge E von gerichteten Kanten; AK heißt die Menge der Aktionsknoten und EK die Menge der Eigenschaftsknoten. Es gibt zwei ausgezeichnete Knoten pre, $goal \in EK$. $indegree(v)$ für $v \in V$ bezeichne die Anzahl der in den Knoten v eingehenden Kanten; $outdegree(v)$ entsprechend die Anzahl der von v ausgehenden Kanten. Es gelte $indegree(pre) = 0$ und $outdegree(goal) = 0$ und pre und $goal$ seien die einzigen Knoten, für die diese Eigenschaften gelten. Es gelte weiterhin $indegree(v) \leq 2,\ \forall v \in V$.*

Wir beschränken uns zunächst auf Plangraphen mit $outdegree(v) \leq 1,\ v \in V \setminus \{pre\}$. Jedem Knoten in einem Plangraphen kann man einen bestimmten Teilgraphen zuordnen, der analog zu einem Plangraphen aufgebaut ist:

Definition 4.2.9 *Sei $PG = (V, E)$ ein Plangraph mit $v \in V$. $TGP(v) = (TV, TE)$ bezeichne den maximalen Teilgraphen von PG mit $pre \in TV$, $outdegree(v) = 0$ und v der einzige Knoten, für den dies gilt.*

Regel 12 (weak nonlinear)

$$\frac{\Rightarrow \mathsf{conflict\_free}(\mathsf{P}_1;T;g_1 \wedge \bigcirc F, \mathsf{P}_2;T;g_2 \wedge \bigcirc F, \mathsf{bc}) \qquad pre \wedge \mathsf{P}_1;T;g_1 \wedge \bigcirc F \Rightarrow \Diamond g_1 \qquad pre \wedge \mathsf{P}_2;T;g_2 \wedge \bigcirc F \Rightarrow \Diamond g_2}{pre \wedge \mathsf{P}_1;T;g_1 \wedge \bigcirc F \ \wedge \mathsf{P}_2;T;g_2 \wedge \bigcirc F \ \wedge \Box \mathsf{bc} \Rightarrow \Diamond[g_1 \wedge g_2]}$$

P_1, P_2 *und* bc *sind Metavariablen,* $\mathsf{conflict\_free}$ *ist ein Metaprädikat, das durch geeignete Regeln reduziert wird* □

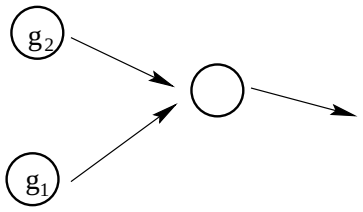
Die Konklusion von Regel 12 ist dabei entweder direkt durch eine Metavariableneinführung entstanden oder als Antezedens der folgenden Regel:

Regel 13 (weak nonlinear cs)

$$\frac{pre \wedge \mathsf{P}_1;T;g_1 \wedge \bigcirc F \ \wedge \ \mathsf{P}_2;T;g_2 \wedge \bigcirc F \ \wedge \Box \mathsf{bc} \Rightarrow \Diamond[g_1 \wedge g_2]}{pre \wedge \mathsf{P};T;[g_1 \wedge g_2] \wedge \bigcirc F \wedge \Box \mathsf{bc} \Rightarrow \Diamond[g_1 \wedge g_2]}$$

□

Durch die Anwendung von Regel 12 führt man nun eine Struktur der Art



in den sich entwickelnden Plangraphen ein. Gleichzeitig etabliert man ein Schema zur Zusicherung der Interferenzfreiheit zwischen den in den Graphen $TPG(g_1)$ und $TPG(g_2)$ zukünftig erscheinenden Aktionen.[65] In diesen beiden Graphen sind offensichtlich alle Aktionen enthalten, für die ein Test auf Interferenzfreiheit relevant werden kann; die Graphen werden inkrementell im Zuge des Beweisfortschrittes erzeugt.
Die Menge der Knoten, für die die Interferenzfreiheit zu einem gegebenen Aktionsknoten $v \in AK$ zu zeigen ist, ist offensichtlich gegeben durch:

[65] Wir setzen hier der Einfachheit halber einen Knoten mit der durch ihn repräsentierten Eigenschaft gleich.

Definition 4.2.10 *Für einen Aktionsknoten n und einen Plangraphen PG ergibt sich der Teilgraph IPG des Graphen, für den die Interferenzfreiheit mit n zu zeigen ist, offensichtlich durch:*

$$IPG(n) := PG \setminus \{p \mid p \text{ Pfad von } pre \text{ nach } goal \text{ in } PG,\ n \text{ Knoten auf } p\}$$

Die erste offene Voraussetzung in Regel 12 dient dazu, entsprechende Zusicherungen über die Interferenzfreiheit der beiden nebenläufigen Verfeinerungen gemäß Definition 4.2.7 in Form entsprechender offener Teilprobleme zu erzeugen. Eine Verfeinerung des Metaprädikates conflict_free findet dabei korrespondierend zu der jeweiligen Instantiierung der Planmetavariablen P_1 bzw. P_2 statt. Die ersten beiden Argumente von conflict_free repräsentieren die Mengen $TPG(g_1)$ bzw. $TPG(g_2)$. Die Regeln 14-18 dienen dazu, diese Mengen zu erzeugen, und die zu überprüfenden Interferenzbedingungen aufzustellen.

Definieren wir dazu ein inkrementelles Verfahren über einem Plangraphen PG.

Schritt 1: Wir ordnen jedem Knoten $n \in V$ eine Menge $cfs(n)$ als die Menge der auf Interferenzfreiheit mit n zu überprüfenden Knoten zu. Initial erhält *goal* den Wert $cfs(goal) = \emptyset$. Zur Entwicklung des Graphen stehen zwei wesentliche Operationen zur Verfügung, die Nebenläufigkeits- und die Aufsplittungsverfeinerung.

Schritt 2: Bei der Nebenläufigkeitsverfeinerung entstehen zwei neue Knoten v_1, v_2, die eine Verbindung zu einem bestehenden Knoten v_0 durch die Kanten (v_1, v_0), (v_2, v_0) eingehen. Es ergeben sich damit:[66]

$$cfs(v_1) := cfs(v_0) \cup V(TPG(v_2)) \text{ und } cfs(v_2) := cfs(v_0) \cup V(TPG(v_1))$$

Wird realisiert durch Regel 12 und 15.

Schritt 3: Bei der Aufsplittungsverfeinerung und der assoziierten Etablierung einer Aktion entsteht eine Sequenz von zwei Knoten $ve \in EK$, $va \in AK$ verbunden mit einem bestehenden Knoten v_0 über die Kanten (ve, va), (va, v_0). Es ergeben sich:

$$cfs(va) := cfs(ve) := cfs(v_0) \text{ und } erzeuge\_test(va, cfs(va))$$

Realisiert durch Regel 14.

Schritt 4: Wird in den partiellen Graphen für ein $ve \in EK$ mit $indegree(ve) = 0$ eine Verbindung (pre, ve) eingefügt, so ergibt sich:

$$erzeuge\_test(ve, cfs(ve))$$

Schritt 5: Die entstehenden Funktionsaufrufe $erzeuge\_test(v, cfs(v))$ werden inkrementell evaluiert, entsprechend den jeweils vollzogenen Instantiierungen in $cfs(v)$. Für die in $cfs(v)$ enthaltenen Aktionsknoten $a_1, \ldots, a_n$ werden Aufrufe

$$Interferenzfrei\_zusicherung(v, a_1), \ldots, Interferenzfrei\_zusicherung(v, a_n)$$

erzeugt, die den jeweils notwendigen Test auf Interferenzfreiheit durchführen.

Realisiert durch Regel 17 und 18.

[66] $V(G)$ bezeichne die Menge der Knoten des Graphen G.

Regel 14 (conflict_free_1)

$$\frac{\Rightarrow \mathsf{conflict\_free}(\mathsf{P}_{11}\ ;\ T\ ;\ pre_a \wedge \bigcirc F, \Delta, \mathsf{bc}_1) \quad \Rightarrow \mathsf{conflict\_free}(ex(a) \wedge \odot\bigcirc F; T; g_1 \wedge \bigcirc F, \Delta, \mathsf{bc}_2)}{\Rightarrow \mathsf{conflict\_free}(\mathsf{P}_1; T; g_1 \wedge \bigcirc F, \Delta, \mathsf{bc})}$$

unter der Voraussetzung, daß die Ersetzung $\{\mathsf{P}_{11}\ ;\ T\ ;\ pre_a \wedge ex(a) \wedge \odot\bigcirc F/\mathsf{P}_1\}$ *bereits in der übrigen Beweisführung entstanden ist; es gelte die Ersetzung* $\{\mathsf{bc}_1 \wedge \mathsf{bc}_2/\mathsf{bc}\}$. □

Analog dazu existiert eine Variante für den Fall $\mathsf{conflict\_free}(\Delta, \mathsf{P}_1; T; g_1 \wedge \bigcirc F, \mathsf{bc})$.
Die nächste Regel propagiert die parallele Aufsplittung einer Planmetavariable auch in die Verwaltung von conflict_free-Prädikaten.

Regel 15 (conflict_free_2)

$$\frac{\Rightarrow \mathsf{conflict\_free}(\mathsf{P}_1\ ;\ T\ ;\ g_1 \wedge \bigcirc F, \Delta, \mathsf{bc}_1) \quad \Rightarrow \mathsf{conflict\_free}(\mathsf{P}_2\ ;\ T\ ;\ g_2 \wedge \bigcirc F, \Delta, \mathsf{bc}_2)}{\Rightarrow \mathsf{conflict\_free}(\mathsf{P}; T; [g_1 \wedge g_2 \wedge \bigcirc F], \Delta, \mathsf{bc})}$$

unter der Voraussetzung, daß die Ersetzung $\{\mathsf{P}_1 \wedge \mathsf{P}_2/\mathsf{P}\}$ *bereits in der übrigen Beweisführung entstanden ist; es gelte die Ersetzung* $\{\mathsf{bc}_1 \wedge \mathsf{bc}_2/\mathsf{bc}\}$. □

Analog dazu existiert eine Variante für den Fall $\mathsf{conflict\_free}(\Delta, \mathsf{P}; T; [g_1 \wedge g_2 \wedge \bigcirc F], \mathsf{bc})$.
Eine Zusicherung zur Entscheidung der Interferenzfreiheit entsteht durch die folgende Regel:

Regel 16 (conflict_free_3)

$$\frac{\mathsf{pre}, ex(a) \wedge \odot\bigcirc F \Rightarrow \phi \wedge \mathsf{bc} \wedge \bigcirc[\phi \wedge g_1]}{\Rightarrow \mathsf{conflict\_free}(\bigcirc F; T; \phi \wedge \bigcirc F, ex(a) \wedge \odot\bigcirc F; T; g_1 \wedge \bigcirc F, \mathsf{bc})}$$

□

Analog dazu existiert eine Variante für den Fall

$$\mathsf{conflict\_free}(ex(a) \wedge \odot\bigcirc F; T; g_1 \wedge \bigcirc F, \bigcirc F; T; \phi \wedge \bigcirc F, \mathsf{bc}).$$

Durch die Anwendung der folgenden Regel entstehen weitere Zusicherungen, die über die Interferenzfreiheit entscheiden. Das Scheitern des Beweises einer dieser Zusicherungen belegt, daß die nebenläufige Planverfeinerung in der verfolgten Weise keinen Erfolg verspricht.

Regel 17 (conflict_free_4)

$$\frac{\mathsf{pre}_1, ex(a) \wedge \odot\bigcirc F \Rightarrow g_2 \wedge \mathit{eff}_b \wedge \mathsf{bc}_1 \wedge \bigcirc[g_1 \wedge g_2 \wedge \mathit{eff}_b] \quad \mathsf{pre}_2, ex(b) \wedge \odot\bigcirc F \Rightarrow g_1 \wedge \mathit{eff}_a \wedge \mathsf{bc}_2 \wedge \bigcirc[g_2 \wedge g_1 \wedge \mathit{eff}_a]}{\Rightarrow \mathsf{conflict\_free}(ex(a) \wedge \odot\bigcirc F; T; g_1 \wedge \bigcirc F, ex(b) \wedge \odot\bigcirc F; T; g_2 \wedge \bigcirc F, \mathsf{bc})}$$

es gelte die Ersetzung $\{\mathsf{bc}_1 \wedge \mathsf{bc}_2/\mathsf{bc}\}$*; die mit den beiden Teilplänen assoziierten Intervalle müssen eine gegenseitige Überlappung erlauben*[67]*, sonst ist der Beweis der Zusicherungen für den aktuellen Planungsprozeß irrelevant.* □

Mit Regel 18 kann der Abschluß der conflict_free-Verfeinerung bzgl. eines nebenläufigen Teilplanes vollzogen werden, wenn dieser in der Teilplanverfeinerungsphase vollständig bestimmt wurde.

Regel 18 (conflict_free_5)

$$\frac{\Rightarrow T}{\Rightarrow \mathsf{conflict\_free}(\Gamma, \Delta, \mathsf{bc})} \; \{T/\mathsf{bc}\}$$

wenn Γ *und* Δ *in der Form* $\circ F;T;\phi \wedge \circ F$ *vorliegen.* □

Die Regeln 14 - 18 sind auf eine rein regressiv erfolgte Verfeinerung der nebenläufigen Teilpläne hin ausgerichtet; sie lassen sich jedoch in analoger Weise auch für rein progressiv oder progressiv/regressiv gemischt erfolgte Teilplanverfeinerungen formulieren.

Betrachtet man z.B. die offene Voraussetzung $pre \wedge \mathsf{P}_1;T;g_1 \wedge \circ F \Rightarrow \Diamond g_1$ im Antezedens der Regel 12 so sieht man leicht, daß sie auf triviale Weise geschlossen werden kann: es genügt, die Planmetavariable P_1 mit dem leeren Plan $\circ F$ zu instantiieren, so daß damit Voraussetzung und Schlußfolgerung der Implikation übereinstimmen. Zu rechtfertigen ist dies jedoch nur, wenn $pre \Rightarrow g_1$ gelten würde, d.h. wenn das Ziel bereits aus den tatsächlichen Vorbedingungen unmittelbar herleitbar wäre, wovon im allgemeinen jedoch nicht ausgegangen werden kann. Der auf triviale Weise gewonnene "Plan" würde damit auch der in Definition 4.2.3 geforderten Aktions*kausalität* eines Planes widersprechen, da die Existenz eines Aktionsmodelles nicht gesichert ist. Es ist deshalb notwendig, die beiden Voraussetzungen zur Teilplanverfeinerung in Regel 12 durch die folgende Regel zu spezialisieren:

Regel 19 (plan_causality)

$$\frac{pre \wedge \mathsf{P} \Rightarrow \Diamond g}{pre \wedge \mathsf{P};T;g \wedge \circ F \Rightarrow \Diamond g}$$

□

Diese Regel faßt das im Abschnitt 4.2.7.4 beschriebene Vorgehen bei der Sicherheitsbedingungsverfeinerung kompakt zusammen.

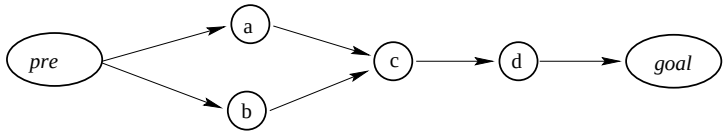
Aus den Ergebnissen der bisherigen Diskussion nebenläufiger Verfeinerungen und den in vorherigen Abschnitten untersuchten Formen *sequentieller* Verfeinerungen ist leicht zu schließen, daß beliebige *seriell-parallele* Ordnungen zwischen an einem Plan beteiligten Aktionen repräsentiert und abgesichert werden können.
Eine seriell-parallele Ordnung ist eine Sonderform einer partiellen Ordnung, bei der eine Ordnung nur aus serieller(sequentieller) und paralleler(nebenläufiger) Komposition bereits

[67] Dies kann über die Beziehungen der assoziierten Planmetavariablen leicht entschieden werden.

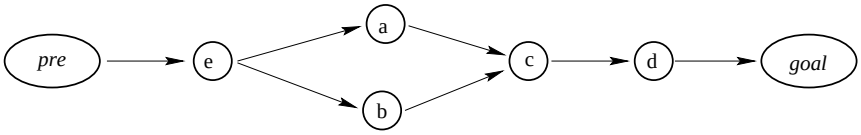
bestehender Ordnungen hervorgehen kann. Zu dieser Kategorie gehört z.B. ein Plan der Form:

$$\left[\begin{array}{l} T\ ;\ pre_a \wedge ex(a) \wedge \odot\bigcirc F\ ;\ T\ ;\ pre_{c1} \wedge \bigcirc F\ \ \wedge \\ T\ ;\ pre_b \wedge ex(b) \wedge \odot\bigcirc F\ ;\ T\ ;\ pre_{c2} \wedge \bigcirc F\ \ \wedge \\ \Box vb_{ab} \end{array}\right]\ ;\ ex(c) \wedge \odot\bigcirc F\ ;\ ex(d) \wedge \odot\bigcirc F$$

Unterstellen wir, daß der Plan einer Spezifikation $pre \wedge \mathsf{Plan} \rightarrow \Diamond goal$ genügt, so beschreibt er die folgende kausale Struktur zwischen den beteiligten Aktionen:



Bezüglich der Aktionen a und b besteht eine nebenläufige Anordnung, die vor der Sequenz der Aktionen c und d lokalisiert ist.
Ein weiteres Beispiel seriell-paralleler Anordnung von Aktionen sei gegeben als:

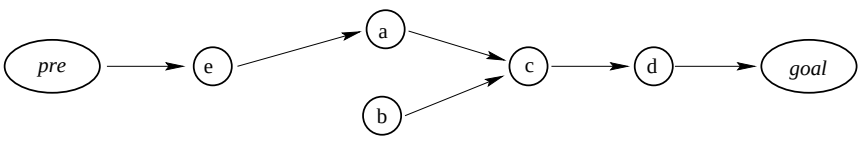


mit einer assoziierten Planstruktur der Form:

$$Ex(e)\ ;\ \left[\begin{array}{l} T\ ;\ pre_a \wedge Ex(a)\ ;\ T\ ;\ pre_{c1} \wedge \bigcirc F\ \ \wedge \\ T\ ;\ pre_b \wedge Ex(b)\ ;\ T\ ;\ pre_{c2} \wedge \bigcirc F\ \ \wedge \\ \Box vb_{ab} \end{array}\right]\ ;\ Ex(c)\ ;\ Ex(d)^{68}$$

Diese Planstruktur kann mit den bisher verfügbaren Verfeinerungsmethoden nur durch einen vorausschauenden Einsatz von Verfeinerungen erreicht werden. Bevor die nebenläufige Verfeinerung eingeführt werden kann, muß bereits eine Zielstrukturierung vorgenommen worden sein, die eine Spezifikation des ersten Teilplanes, der letztendlich nur aus der Aktion e besteht, erzeugt hat.

Generierung beliebiger nichtlinearer Pläne Um die gerade beobachtete Einschränkung zu überwinden und damit auch nach einer bereits erfolgten Nebenläufigkeitsverfeinerung den Effekt einer dieser vorgeschalteten seriellen Verfeinerung zu erreichen, wird eine *Korrelationsverfeinerung* verwendet. Zur Erläuterung nehmen wir an, daß der obige Plan bereits partiell bis zu dem folgenden Stadium verfeinert ist:



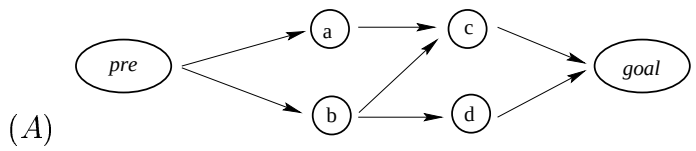
[68] Wir verwenden die Abkürzung $Ex(\alpha) \leftrightarrow ex(\alpha) \wedge \odot\bigcirc F$.

und als Formel der Form

$$\overbrace{\underbrace{\overbrace{Ex(e)}^{\mathsf{P}_8};\overbrace{T\,;\,pre_a \wedge Ex(a)}^{\mathsf{P}_7}\,;\,T\,;\,pre_{c1} \wedge \bigcirc F\ \wedge}_{\mathsf{P}_5}\ \Box vb_{ab}\ \wedge}^{\mathsf{P}_2}\ ;\overbrace{\underbrace{Ex(c)}_{\mathsf{P}_3}\,;\underbrace{Ex(d)}_{\mathsf{P}_4}}^{\mathsf{P}_1}$$
$$\underbrace{\underbrace{\boxed{\mathsf{P}_{10}}\,;\underbrace{T\,;\,pre_b \wedge Ex(b)}_{\mathsf{P}_9}\,;\,T\,;\,pre_{c2} \wedge \bigcirc F\ \wedge}_{\mathsf{P}_9}}_{\mathsf{P}_9}$$

beschrieben ist. Es sind jeweils die entsprechenden Planmetavariablen den Teilformeln, mit denen sie instantiiert wurden, zugeordnet; P_{10} ist als Einzige noch nicht instantiiert. Nehmen wir nun an, daß die Aktion e auch als Effekt das von P_{10} zu erreichende Ziel pre_b unmittelbar erreichen kann. Dieser Umstand kann am effektivsten ausgenutzt werden, indem die Planmetavariable P_{10} mit der Variablen P_8 *korreliert*, d.h. es findet eine Korrelationsverfeinerung des mit der Variablen P_{10} assoziierten Intervalles statt. Es ergibt sich dann die oben beschriebene vollständige Planstruktur. Die Verfeinerung des mit der Metavariablen P_{10} assoziierten Teilintervalles ist nun identisch mit der Verfeinerung des mit P_8 assoziierten Intervalles.
Zu beachten ist, daß die mit einer Korrelationsverfeinerung einhergehende Substitution von Planmetavariablen bzgl. der bereits bestehenden Substitutionsbeziehungen zyklenfrei sein muß.

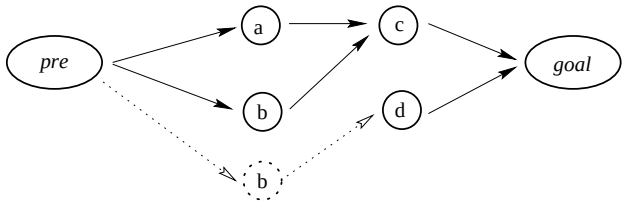
Aus Sicht der Plangraphentwicklung und der assoziierten Verwaltung von zu überprüfenden Interferenzbedingungen bedeutet die Einführung einer Korrelationsverfeinerung bei Beibehaltung des inkrementellen Verfahrens ihrer Erzeugung automatisch ein Rückgang der zu prüfenden Bedingungen. Durch die Korrelationsverfeinerung wird eine neue Kante ausgehend von einem bestehenden Aktionsknoten v_1 hinzu einem Eigenschaftsknoten v_2 mit $indegree(v_2) = 0$ in den partiellen Graphen eingefügt; v_1 darf nicht auf einem Pfad von v_2 nach *goal* liegen, da der Plangraph sonst seine Zyklenfreiheit verlieren würde. Damit wird ein Teilgraph von $TPG(v_2)$ identisch zu dem Graphen $TPG(v_1)$. Dies hat zur Folge, daß sich die Anzahl notwendiger Interferenzzusicherungen automatisch reduziert.
Durch die Verwendung von Nebenläufigkeits- und Aufsplittungsverfeinerungen konnten bisher (rückwärts betrachtet) nur Verzweigungen in einen Graphen eingeführt werden. Mit dem Mittel der Korrelationsverfeinerung kann man nun die Anzahl der bestehenden Verzweigungen wieder beliebig reduzieren, d.h. man kann beliebige partielle Ordnungen zwischen Aktionsknoten realisieren.
Der kritische Fall, der über die Darstellung seriell-paralleler Ordnungen hinausgeht, ist der im folgenden beschriebene Fall einer "N"-Ordnung zwischen Aktionen:



(A)

Man kann diese Ordnung nicht durch die alleinige Anwendung sequentieller und nebenläufiger Kompositionen erreichen. Was man jedoch erreichen kann und der "N"-Ordnung

sehr nahe kommt, ist die folgende Ordnung:

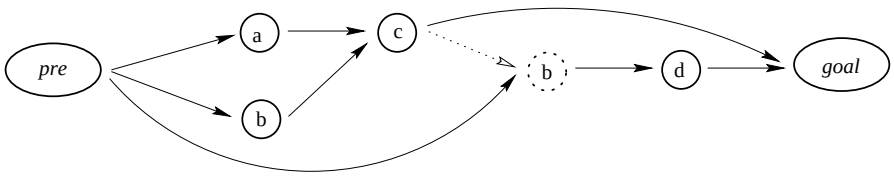


(A')

Die Aktion b tritt hierbei einmal redundant auf. Da das zweite Auftreten der Aktion b aber nicht notwendig ist, vielleicht sogar so gar nicht möglich ist, da eine notwendige Ressource verbraucht ist (die Vorbedingungen für b könnten zwischenzeitlich nicht mehr gelten), kann man sich der Korrelationsverfeinerung bedienen, um das Problem zu lösen. Nehmen wir hierzu an, daß die Planstruktur bereits soweit entwickelt ist, daß nur noch ein kausaler Link fehlt, der zur Aktion d führt; wir integrieren wiederum die mit Teilplänen assoziierten Planmetavariablen in die Darstellung:

$$\overbrace{\overbrace{\overbrace{T\ ;\ pre_a \wedge Ex(a)}^{\mathsf{P}_4}\ ;\ T\ ;\ pre_{c1} \wedge \bigcirc F\ \wedge}^{\mathsf{P}_3}}^{\mathsf{P}_1}$$
$$\Box vb_{ab} \wedge \qquad ;\ T\ ;\ pre_c \wedge Ex(c)\ ;\ T\ ;\ g_1 \wedge \bigcirc F\ \wedge$$
$$\underbrace{T\ ;\ pre_b \wedge Ex(b)}_{\mathsf{P}_5}\ ;\ T\ ;\ pre_{c2} \wedge \bigcirc F$$
$$\Box vb_{abcd} \wedge$$
$$\underbrace{\boxed{\mathsf{P}_6}\ ;\ T\ ;\ pre_d \wedge Ex(d)}_{\mathsf{P}_2}\ ;\ T\ ;\ g_2 \wedge \bigcirc F$$

Wir nehmen nun wiederum an, daß pre_d von der Aktion b schon erreicht wird und korrelieren die Metavariable P_6 mit der Metavariablen P_5. Der entwickelte Plan entspricht damit der in (A) gezeigten Kausalitätsstruktur; die Beschreibung als Planformel entspricht jedoch der in (A') gezeigten Struktur. Dies ist zunächst natürlich damit begründet, daß die bereits partiell entwickelte Planformel keine strukturelle Zusammenführung mehr erlaubt, andererseits kann natürlich eine "N"-Ordnung nicht alleine mit sequentieller und nebenläufiger Komposition redundanzfrei dargestellt werden; es ist zusätzlich eine disjunktive Komposition erforderlich. Die mit Redundanz versehene Planformel läßt nun zusätzlich zu der intendierten Kausalitätsstruktur prinzipiell auch noch eine weitere Lesart zu:



Die Verbindung zwischen der Aktion c zu der zweiten Aktion b stellt nun nur eine reine Ordnungsbeziehung und keine Kausalitätsbeziehung dar. Ob diese Struktursicht jedoch im konkreten Fall semantisch möglich ist, wird während des Planungsvorganges in keiner

Weise überprüft; dies kann jedoch z.B. während einer später folgenden Planinterpretation geschehen.

Durch den vorgestellten kombinierten Einsatz von Nebenläufigkeits- und Korrelationsverfeinerungen ergibt sich in der Verallgemeinerung die in der folgenden Grammatik beschriebene Planstruktur:

$$
\begin{array}{rcl}
\texttt{Plan} & ::= & \texttt{Plan} \wedge \texttt{Plan} \wedge \Box\ \texttt{MfF} \mid \\
 & & \texttt{S\_Plan} \mid \\
 & & \texttt{Plan ; S\_Plan} \\
\texttt{S\_Plan} & ::= & \texttt{S\_Plan ;}\ T\ \texttt{;}\ \texttt{MfF} \wedge ex(Aktionsterm) \wedge \odot\bigcirc F \mid \\
 & & \texttt{S\_Plan ;}\ T\ \texttt{;}\ \texttt{MfF} \wedge \bigcirc F \mid \\
 & & \bigcirc F \\
\texttt{MfF} & ::= & \texttt{modalfreie LLP-Formel}
\end{array}
$$

Es ergibt sich hieraus aus der Sicht der Intervallstrukturierung eines mit einer Planformel assoziierten Modelles die folgende Verfeinerung: Sei $\sigma = \langle \sigma_0 \ldots \sigma_n \rangle$ das durch den Gesamtplan verfeinerte Intervall. Dann entstehen durch die vorgenommene Strukturierung Verfeinerungen $v_1, \ldots, v_m$ mit assoziierten Intervallen $\sigma^1, \ldots, \sigma^m$ mit $|\sigma^i| = n$ für $i = 1, \ldots, m$. Die einzelnen Verfeinerungen stehen wie folgt miteinander in Beziehung:

- Gilt für zwei Intervalle σ^k und σ^l, daß es ein maximales terminales Subintervall τ gibt mit $\sigma^k = \sigma^{k_1} \circ \tau$ und $\sigma^l = \sigma^{l_1} \circ \tau$ so sind σ^{k_1} und σ^{l_1} durch eine Nebenläufigkeitsverfeinerung entstanden.
- Die Anwendung einer Korrelationsverfeinerung führt dazu, daß es zwei Intervalle σ^k und σ^l gibt, die ein gemeinsames initiales Teilintervall λ aufweisen, $\sigma^k = \lambda \circ \sigma^{k_1}$ und $\sigma^l = \lambda \circ \sigma^{l_1}$.

Zu bemerken ist nochmals, daß die entstehende Planstruktur, bestehend aus sequentiellen und nebenläufigen Kompositionen, die bestehenden und abgesicherten Teilintervallbeziehungen nicht notwendigerweise eindeutig charakterisiert. Die gerade beschriebenen Zusammenhänge erlauben es jedoch, die abgesicherte Interpretation leicht zu finden. Als alternative, eindeutige Art der Klassifizierung des kritischen Falles einer "N-Ordnung" wurde bereits oben die Möglichkeit einer disjunktiven Komposition erwähnt. Daraus ergibt sich für die Kausalitätsstruktur (A) die Planformel

$$
\left[\begin{array}{l}
T;pre_a \wedge Ex(a);T;pre_{c1} \wedge \bigcirc F \wedge \\
\Box vb_{ab} \wedge \qquad\qquad ;T;pre_c \wedge Ex(c);T;g_1 \wedge pre_d \wedge Ex(d);T;g_1 \wedge g_2 \wedge \bigcirc F \wedge \\
T;pre_b \wedge Ex(b);T;pre_{c2} \wedge \bigcirc F \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \Box vb_{abcd} \wedge
\end{array}\right]
$$

$$
\vee
$$

$$
\left[\begin{array}{l}
T;pre_a \wedge Ex(a);T;pre_{c1} \wedge \bigcirc F \wedge \\
\Box vb_{abd} \wedge \qquad\qquad\qquad\qquad ;T;pre_c \wedge Ex(c);T;g_1 \wedge \bigcirc F \wedge \\
T;pre_b \wedge Ex(b);T;pre_d \wedge Ex(d);T;pre_{c2} \wedge \bigcirc F \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \Box vb_{abcd} \wedge
\end{array}\right]
$$

Nichtlineare Pläne, wie sie bisher vorgestellt wurden, sind zunächst unterbrechbare Pläne; diese Eigenschaft ergibt sich aus der Strategie, nach der sie erstellt wurden. Für ihre Verwendung kann diese Eigenschaft aber unnötig sind, d.h. es muß möglich sein, einen *dichten*,

nur die explizit benannten Aktionen betreffenden, nichtlinearen Plan zu gewinnen. Analog zu der Einführung einer Intervallängenbeschränkung (siehe Abschnitt 4.2.7.4) gibt es auch hier die Möglichkeit, die Planformel durch eine entsprechende Modalformel so zu ergänzen, daß die gewünschte Intervallänge resultiert; diese Ergänzung bildet in unserem Sinne auch eine Nebenläufigkeitsverfeinerung, jedoch nur zu dem Zweck, mögliche Aktionsmodelle durch rein zeitliche Aussagen zu beschränken.
Wenn P ein unterbrechbarer nichtlinearer Plan ist und n die Anzahl explizit benannter Aktionen (bereinigt um die Anzahl der durch Korrelationsverfeinerung entstandenen Aktionen) in P ist, dann beschreibt die nebenläufige Komposition

$$P \wedge \odot^n \bigcirc F$$

den mit P assoziierten dichten Plan. Die Zeitbeschränkung ist hier durch eine rein optionale Nachbearbeitung von P entstanden.
Der entstandene Plan P kann gegebenenfalls weiter verfeinert werden, z.B. indem die nichtlineare Abstraktion in eine strenge Linearisierung verfeinert wird, oder er wird in eine andere Form transformiert, z.B. indem schwache Sicherheitsbedingungen zu starken konkretisiert werden. Ein dichter nichtlinearer Plan P', für den gilt, daß keine nebenläufigen Verfeinerungen identische Aktionen enthalten, kann unter Ausnutzung der Semantik von LLP strukturell stark vereinfacht werden.
Für den oben beschriebenen Plan zur Darstellung einer "N"-Ordnung ergibt sich:

$$\left[\begin{array}{l}\Diamond ex(a) \wedge \\ \Diamond ex(b)\end{array} ;ex(c);ex(d); \odot \bigcirc F \ \wedge \odot^4 \bigcirc F\right] \vee \left[\begin{array}{l}\Diamond ex(a) \wedge \\ \Diamond ex(b);ex(d)\end{array} ;ex(c); \odot \bigcirc F \ \wedge \odot^4 \bigcirc F\right]$$

Durch die Beschränkung der Intervallänge auf die Anzahl der explizit benannten Aktionen, $\ldots; \odot^4 \bigcirc F$ können alle Sicherheitsbedingungen entfallen, sie sind redundant. Es ist lediglich zu fordern, daß die zeitlich letzte Aktion des Planes im vorletzten Zustand des beschränkten Intervalles ausgeführt werden muß[69]; dies erreicht man durch die Verfeinerung $\ldots; \odot \bigcirc F$.

Konflikterkennung und -lösung Wie oben beschrieben entstehen durch die Anwendung von Regel 12 Zusicherungen über die Konsistenz der vorgenommenen Abstraktion, die Erfüllbarkeit des Metaprädikates conflict_free ist zu beweisen. Scheitert der Beweis einer dieser Zusicherungen, so kann mithilfe einer geeigneten Korrelationsverfeinerung eine Konfliktlösung vorgenommen werden. Gehen wir zur Erläuterung von folgendem partiellen Plan aus:

$$\begin{array}{r}\overbrace{\mathsf{P}_{11};T;pre_a \wedge Ex(a)}^{\mathsf{P}_1};T;g_1 \wedge \bigcirc F \ \wedge \\ \mathsf{P}_{21};T;pre_b \wedge Ex(b);T;g_2 \wedge \bigcirc F \ \wedge \\ \Box bc\end{array}$$

Abbildung 4.28 zeigt relevante Ausschnitte des Beweisvorganges zur Entstehung dieses Teilplanes und seiner weiteren Verfeinerung für den Fall eines Konfliktes zwischen den Aktionen a und b.
Es scheitere nun beispielhaft der Beweis der Zusicherung in Sequenz (33) daran, daß der Effekt *eff_b* der Aktion b nicht invariant ist bzgl. Aktion a, und zwar in dem Sinne, daß

[69] Damit ihr Effekt im letzten Zustand verfügbar ist.

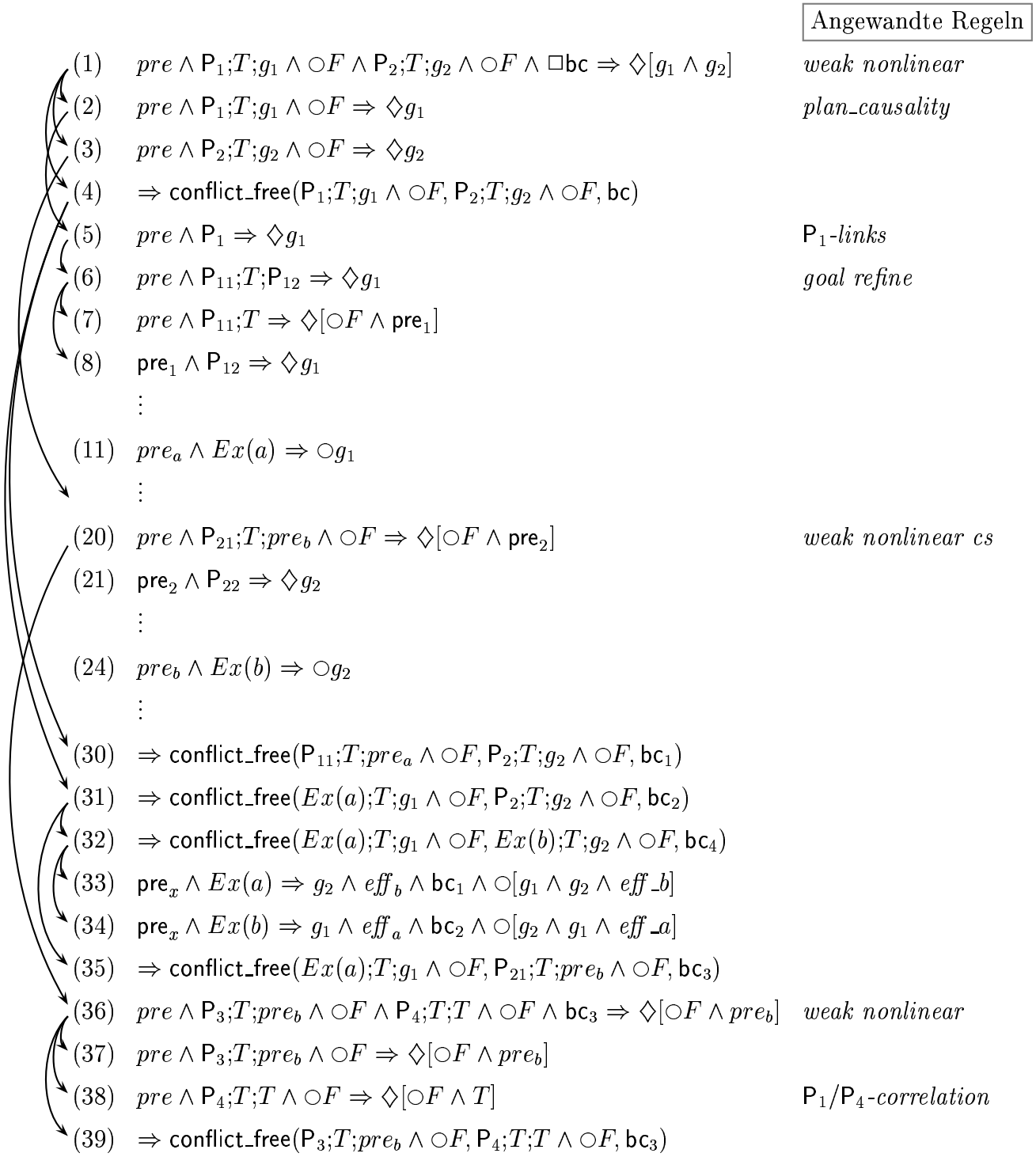


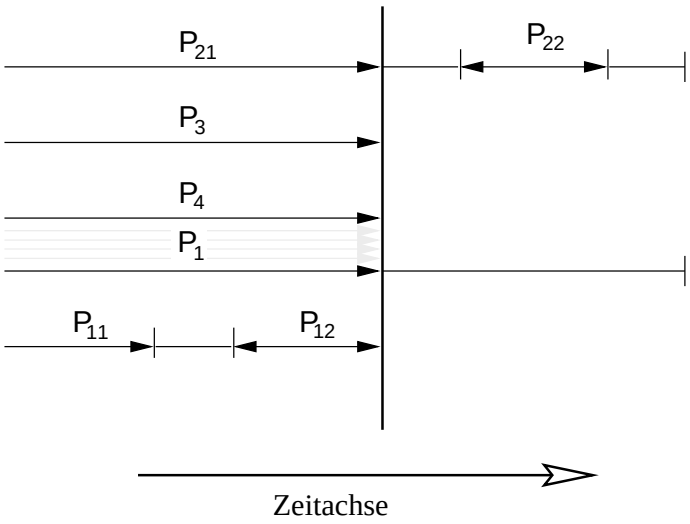
Abbildung 4.28: Einführung einer Ordnungsbeziehung in nebenläufige Verfeinerung

pre_a, die Vorbedingungen der Aktion a, inkonsistent sind mit eff_b. Dies bedeutet, daß Aktion b nicht vor Aktion a angeordnet sein kann, ihr jedoch folgen kann. Die notwendige totale Ordnung kann auf die folgende Art erreicht werden:

1. Expansion des partiellen Teilplanes P_{21};T;$pre_b \wedge \bigcirc F$ in Sequenz (20) mithilfe einer geeigneten Instanz der Regel 13.

2. Nach Anwendung von Regel 12 erfolgt eine Lösung des Teilproblemes in Sequenz (37) durch eine Korrelationsverfeinerung, indem die Metavariable P_4 mit der Metavariable P_1 gleichgesetzt wird, es ist nun nur noch die weitere Verfeinerung von P_1 relevant.

Durch die Anwendung einer Korrelationsverfeinerung verringert sich die Anzahl der zu berücksichtigenden Zusicherungen über die Konfliktfreiheit nebenläufiger Verfeinerungen. Für den obigen Beweisausschnitt hat dies aufgrund der nun geltenden Beziehungen zwischen den Planmetavariablen



folgende Auswirkungen auf die Expansion von conflict_free-Prädikaten:

- Die Expansion von Sequenz (32) reduziert sich auf die Erzeugung der Voraussetzung gemäß Sequenz (34); aufgrund der nun geltenden Sequenzialität der mit den Metavariablen P_{12} und P_{22} assoziierten Intervalle ist das in Sequenz (33) gestellte Problem nun irrelevant.

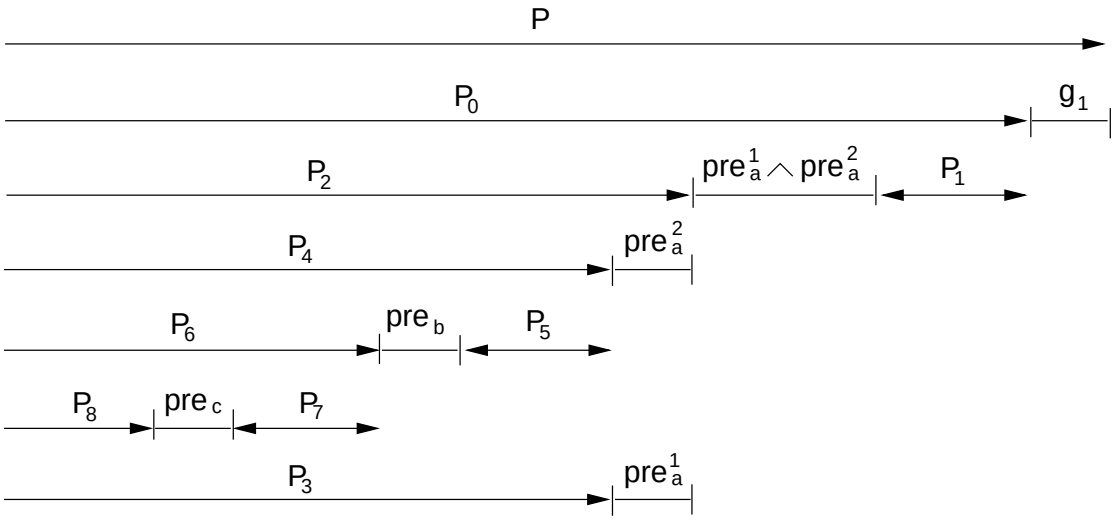
- Bei der Expansion von Sequenz (30) werden alle Teilprobleme irrelevant, die sich auf Teilintervalle von P_2 beziehen, die dem mit P_{21} assoziierten Intervall folgen, d.h. die Berücksichtigung der Aktion b bzgl. des Teilplanes P_{11} ist irrelevant.

- Die bei der weiteren Expansion von Sequenz (35) entstehenden Konfliktfreiheitstests mit Beteiligung der Metavariable P_4 sind irrelevant, da durch die Korrelation von P_4 mit P_1 bedeutungslose Tests zwischen identischen Intervallen entstehen.

- Durch die Korrelation von P_4 mit P_1 überprüft die Zusicherung in Sequenz (39) nun die Beziehung von P_3 zu P_1, d.h. die verbleibenden Zusicherungen bzgl. Sequenz (30) sind nun redundant und werden von Sequenz (39) abgedeckt.

Aufgrund des regressiven Vorgehens im Planungsprozeß kann es vorkommen, daß die Bildung der schwächsten Vorbedingungen eines Zieles bzgl. einer bestimmten Aktion in der Entstehung einer disjunktiven Bedingung endet. Dadurch entsteht im allgemeinen ein neues, disjunktives Teilziel. Um die Uniformität und Einfachheit des gesamten nebenläufigen Planungsprozesses dennoch zu gewährleisten, werden die disjunktiven Vorbedingungen direkt nach ihrer Entstehung determiniert; gerechtfertigt ist ein solches Vorgehen durch die allgemeine Folgerungsbeziehung $\phi \rightarrow \phi \vee \psi$. Die Determinierungsauswahl zwischen zwei disjunktiven Vorbedingungen kann durch eine Heuristik vorgenommen werden z.B. kann man sich für das Disjunktionsglied mit der geringsten Anzahl atomarer Formeln entscheiden.

Das vorgestellte Verfahren zur Erzeugung nebenläufiger Pläne ist nicht, wenn gleich dies die Beweisskizze in Abbildung 4.28 suggerieren könnte, auf ein rein regressives Vorgehen bei der Planerstellung fixiert. Die Teilzielverfeinerung in Sequenz (6) kann ebenso analog zu dem Vorgehen der zielgerichteten Progression bei der Generierung rein sequentieller Pläne (vgl. Abbildung 4.16 und die entsprechenden Erläuterungen) geschehen. Der wesentliche Unterschied besteht darin, daß während der "Suchphase" nach einer Aktion, deren Vorbedingungen alle im initialen Zustand gelten, auch nebenläufige Verfeinerungen vorgenommen werden können. Die dadurch entstehenden offenen Teilziele können dann im Verlaufe der Progressionsphase für den Fall, daß kein Erzeuger für sie explizit etabliert werden muß, durch entsprechende Korrelationsverfeinerungen abgeschlossen werden. Die Problematik, daß in der partiellen Regressionsphase, also der Suchphase, Zwischenziele als *notwendige* Teilziele eingeführt werden, konnte bei der Erstellung sequentieller Pläne effektiv durch eine modifizierte Zielverfeinerung gelöst werden (vgl. den Abschnitt auf Seite 149 ff.). Dadurch daß die nebenläufigen Verfeinerungen, genauergenommen die notwendigen Verfeinerungen zur Gewinnung unterbrechbarer Pläne, die Zwischenziele explizit in die Planstruktur einbauen, ist ein einfaches Vorgehen wie im Zusammenhang mit linearen Plänen hier nicht möglich. Die einzig effektive Variante ist hier vielmehr die gezielte und kontrollierte Modifikation der bisher verfeinerten Planstruktur. Betrachten wir zu der Planspezifikation $pre \wedge \mathsf{P} \rightarrow \Diamond g_1$ beispielhaft die durch folgende Beziehungen zwischen Planmetavariablen charakterisierte Situation während der Planerstellung:



Wir gehen dabei von folgenden Annahmen aus:

- eine Aktion a mit Vorbedingungen $pre_a^1 \wedge pre_a^2$ kann das notwendige Ziel g_1 erreichen;
- eine Aktion b mit Vorbedingung pre_b kann das Hilfsziel pre_a^2 erreichen;
- eine Aktion c mit Vorbedingung pre_c kann das Hilfsziel pre_b erreichen;
- es gilt: $pre \rightarrow pre_c$ und $pre_c \wedge Ex(c) \rightarrow \bigcirc g_1$.

Aufgrund des Nebeneffektes der Aktion c, die im initialen Zustand ausgeführt auch schon das einzig notwendige Ziel g_1 erreicht,[70] könnte mit einer einzigen Aktion die gesamte Planspezifikation erfüllt werden. Die vorgenommene Planstrukturverfeinerung besteht jedoch noch auf der Erreichung des Zieles pre_a^1, wofür ein beliebig großer Aufwand notwendig sein kann. Umgehen könnte man dies, indem man die Verfeinerung der Planmetavariable P_0 modifiziert, derart daß sie einer Korrelation $\mathsf{P}_6/\mathsf{P}_0$ entspricht. Nimmt man eine solche Modifikation einer Substitution vor, so müssen alle Teile des bisherigen Beweises, die auf irgendeine Art mit den Metavariablen $\mathsf{P}_1, \ldots, \mathsf{P}_5$ assoziiert sind, deaktiviert werden; eine Weiterführung dieser Beweisteile ist nicht mehr notwendig, um einen erfolgreichen Gesamtbeweis zu führen. Die Modifikation einer früheren Verfeinerung durch die Anwendung einer Korrelationsverfeinerung wie sie gerade beschrieben wurde, ist nur möglich, wenn die damit assoziierte Substitution von Metavariablen zyklenfrei ist. Die praktische Durchführung der Deaktivierung von noch offenen Teilbeweisen kann aufgrund bereits bestehender komplexer Planstrukturbeziehungen u.U. mit hohem Aufwand verbunden sein. Zu einem gegebenen offenen Teilbeweis, muß dazu untersucht werden, ob die mit ihm assoziierte Teilplanstruktur noch eine Substitutionsverbindung zu der initialen Planmetavariable P aufweist. Da jedoch alle Substitutionsbeziehungen zusammengenommen einem gerichteten azyklischen Graphen entsprechen, kann dies eindeutig entschieden werden.

4.2.7.6 Planen mit konditionalen Effekten

Die in Abschnitt 4.2.1 eingeführte Form von Aktionsaxiomen erlaubt die Spezifikation von Aktionen mit konditionalen Effekten, d.h. in Abhängigkeit unterschiedlicher Voraussetzungen erzeugt die gleiche Aktion unterschiedliche Effekte. Die Formulierung solcher Aktionen stellt dabei bereits eine domänenspezifische Abstraktion dar, da sie die Aktion von einem expliziten kontextdeterminierenden Parameter – die Bedingung, die entscheidend für das spezifische Verhalten ist – befreit. Die Situationsabhängigkeit der Aktion ist dadurch *verschleiert* und intransparent, d.h. dem Auftreten der Aktion in einem Plan kann sie nicht mehr angesehen werden. In Abschnitt 4.2.5 wurden folgende Vereinbarungen für die Formulierung einer Aktion mit konditionalen Effekten getroffen:

- streng unterschiedliche Voraussetzungen determinieren einen bestimmten Effekt und
- Effekte können sich höchstens teilweise überschneiden.

Die in Abschnitt 4.2.2 beschriebenen Formen zulässiger Planspezifikationen erlauben die Möglichkeit zur Spezifizierung unsicheren Wissens in Form nichtdeterministischer Vorbedingungen.

[70] Eine Instantiierung von P_6 würde sich dabei z.B. ergeben als $\bigcirc F \,;\, T \,;\, pre_c \wedge Ex(c)$.

Es werden nun die Auswirkungen der Verwendung von Aktionen mit konditionalen Effekten bzgl. verschiedener Planverfeinerungskategorien und Verfahren insbesondere auch unter Berücksichtigung unsicheren Wissens diskutiert.
Als Ausgangspunkt sei folgende Modellcharakterisierung gegeben:

$$\sigma = \left\langle \boxed{\begin{matrix} pre \\ ? \end{matrix}}_0 \ldots \boxed{goal}_n \right\rangle,\ 0 \leq n \text{ und nicht fest}$$

Eine Aktion a sei durch die folgenden beiden Axiome beschrieben:

$$pre_1 \wedge ex(a) \rightarrow \bigcirc[\phi \wedge \mathit{eff}_1]$$
$$pre_2 \wedge ex(a) \rightarrow \bigcirc[\phi \wedge \mathit{eff}_2]$$

Nehmen wir nun an, daß die Beziehung $\phi \rightarrow goal$ gilt und σ regressiv verfeinert wird (vgl. dazu den Abschnitt 4.2.7.1). Dies führt dann zu der folgenden disjunktiven Verfeinerung σ':

$$\left\{ \left\langle \boxed{\begin{matrix} pre \\ ? \end{matrix}}_0 \cdots \boxed{\begin{matrix} pre_1 \\ a \end{matrix}}_{n_1-1} \boxed{goal}_{n_1} \right\rangle, \left\langle \boxed{\begin{matrix} pre \\ ? \end{matrix}}_0 \cdots \boxed{\begin{matrix} pre_2 \\ a \end{matrix}}_{n_2-1} \boxed{goal}_{n_2} \right\rangle \right\}$$

Der Nichtdeterminismus taucht hierbei nur im Beschreibungsintervall auf. Man hat nun optional die Wahl die alternativen Verfeinerungen beide fortzuführen, notwendig ist nur der Abschluß einer Alternative. Die Determiniertheit des Anfangszustandes pre zeigt auf das regressive Vorgehen keinen Einfluß.
Nehmen wir nun an, daß die Beziehung $pre \rightarrow [pre_1 \vee pre_2]$ gilt, aber nicht $pre \rightarrow pre_1$ und $pre \rightarrow pre_2$. Es gelte ebenso die Beziehung $goal \leftrightarrow \psi \wedge \mathit{eff}_1$, d.h. die Aktion a kann zur Erreichung des Zieles beitragen. Eine progressive Verfeinerung von σ ergibt sich als:

$$\left\{ \left\langle \boxed{\begin{matrix} pre \\ a \end{matrix}}_0 \boxed{\begin{matrix} \phi \wedge \mathit{eff}_1 \\ ? \end{matrix}}_1 \cdots \boxed{goal}_{n_1} \right\rangle, \left\langle \boxed{\begin{matrix} pre \\ a \end{matrix}}_0 \boxed{\begin{matrix} \phi \wedge \mathit{eff}_2 \\ ? \end{matrix}}_1 \cdots \boxed{goal}_{n_2} \right\rangle \right\}$$

Hier pflanzt sich der Nichtdeterminismus des Anfangszustandes fort und führt dazu, daß die Aktion a zu einer nichtdeterministischen Aktion wird; beide Verfeinerungen sind hier jedoch notwendig und nicht wie im Falle der Regression nur optional.
Sind jedoch deterministische Vorbedingungen pre gegeben, kann nur entweder pre_1 oder pre_2 gelten und es entsteht eine deterministische Verfeinerung.

Durch das bei der Erstellung unterbrechbarer Pläne eingeführte Konzept, auch Beschreibungsintervallinformation im Plan explizit sichtbar zu machen, ist es möglich, einen Plan, der Aktionen mit konditionalen Effekten enthält, transparenter zu gestalten. Die verschleierte Situationsabhängigkeit kann somit je nach Verwendung des Planes entsprechend explizit gemacht werden. Bei einem regressiv erzeugten Plan, der nach jeder Aktionsetablierung nur jeweils eine Alternative weiterverfolgt, kann so die aktuell verwendete Aktionsausprägung im Plan sichtbar gemacht werden.[71]
Betrachten wir dazu in einem Beweisausschnitt das prinzipielle Vorgehen:

[71] Es kann hierbei durchaus ein dichter linearer Plan entstehen, indem die Unterbrechungsintervalle mit einer Intervallängenbeschränkung von Null versehen werden.

			Angewandte Regeln
(1)	pre, P	$\Rightarrow \bigcirc goal$	P-*links*
(2)	pre, $P_1 \wedge$ wp $\wedge \odot\bigcirc F$	$\Rightarrow \bigcirc goal$	
(3)	pre, $P_1 \wedge$ wp $\wedge \odot\bigcirc F$	$\Rightarrow \bigcirc goal \wedge \bigcirc goal_a$	$\wedge$-*links*,* $\wedge$-*rechts*
(4)	pre, P_1, wp, $\odot\bigcirc F$	$\Rightarrow \bigcirc goal$	*dynamisch erzeugte Regel*
5()	pre, P_1, wp, $\odot\bigcirc F$	$\Rightarrow wp_{goal} \wedge ex(a)$	P_1-*links,* pre-*links*
(6)	wp_{goal}, $ex(a)$, wp, $\odot\bigcirc F$	$\Rightarrow wp_{goal} \wedge ex(a)$	wp-*links*
(7)	pre, P_1, wp, $\odot\bigcirc F$	$\Rightarrow \bigcirc goal_a$	P_1-*links,* pre-*links*
(8)	wp_{goal}, $ex(a)$, wp, $\odot\bigcirc F$	$\Rightarrow \bigcirc goal_a$	*dynamisch erzeugte Regel*
(9)	wp_{goal}, $ex(a)$, wp, $\odot\bigcirc F$	$\Rightarrow wp_{goal_a} \wedge ex(a)$	wp-*links*
(10)	wp_{goal}, $ex(a)$, wp_{goal_a}, $\odot\bigcirc F$	$\Rightarrow wp_{goal_a} \wedge ex(a)$	
(11)	wp_{goal}, $ex(a)$, wp_{goal_a}, $\odot\bigcirc F$	$\Rightarrow wp_{goal} \wedge ex(a)$	

Sequenz (2) wird hier redundant transformiert, d.h. es gelte $goal \rightarrow goal_a$. Das Teilziel $goal_a$ steuere hier die Auswahl der Aktion, bzgl. der eine Regression vorgenommen wird, es beschreibt einen primären Effekt der Aktion a. Die Metavariable wp als Teil der Planstruktur dient der Aufnahme der schwächsten Vorbedingungen dieses primären Effektes bzgl. der Aktion a. Ihre Instantiierung reflektiert demnach den aktuellen Kontext der Ausführung von Aktion a. Der Übergang von Sequenz (7) nach Sequenz (8) übernimmt die bereits bestehenden Instantiierungen der beteiligten Metavariablen. Um Sequenz (9) zu erhalten, bildet man eine Axiomeninstanz der Aktion a bzgl. ihres primären Effektes $goal_a$ und bildet daraus die assoziierte Sequenzenregel. Der Plan ist nun angereichert um den primären aktionsrelevanten Situationskontext.

4.2.7.7 Planen durch Aufgabenreduktion

Die Verwendung von Aufsplittungsverfeinerungen als eine Methode der Problemreduktion war in den bisher beschriebenen Methoden zur Planerstellung stets rein durch die Fokussierung auf eine bestimmte Aktion motiviert – eine Aktion, die direkt zur Zielreduktion beiträgt, oder eine Aktion, die im aktuellen Zustand ausführbar ist. Es *entstanden* dadurch jeweils neue Zwischenziele durch die Anwendung regressiv oder progressiv erzeugter Aktionsrahmenaxiome. Diese wurden zum Teil nur heuristisch eingesetzt, um die Suche nach einer Lösung zu steuern.[72] Diese Art der Planungsproblemreduktion stellt aber nur einen Spezialfall einer allgemeinen *Aufgabenreduktionsverfeinerung* dar. Die Methodik des Einsatzes hierarchischer Aufgabenreduktionen zur Erstellung von Plänen zeichnet eine ganze Klasse von Planern, meist sehr praktisch orientierte Ansätze, aus (vgl. [Erol et al. 94] und auch Abschnitt 3.1.4.3). Aus unserer Sicht der Modellverfeinerung geschieht dabei

[72]Vgl. etwa die parametrisierte Verwendung der Regel *goal refine* in der Abbildung 4.16. Das dort eingeführte Zwischenziel wird durch eine partielle Regression bzgl. einer geeigneten Aktion gewonnen.

folgendes. Gegeben sei eine Intervallcharakterisierung σ:

$$\sigma = \left\langle \boxed{\begin{matrix} pre \\ ? \end{matrix}}_0 \ldots \boxed{goal}_n \right\rangle, \ 0 \leq n \text{ und nicht fest},$$

die ein Planungsproblem in einer spezifischen Domäne $\mathcal{D}$ spezifiziert. Nun gebe es in $\mathcal{D}$ Wissen darüber, wie ein Planungsproblem hierarchisch aufgeteilt werden kann in immer kleinere Teilprobleme, bis hinunter zu der elementaren Aufteilung gemäß einer Aktionsetablierung. Das Intervall σ wird durch eine grobe Intervallverfeinerung z.B. näher beschrieben in der Form:

$$\sigma' = \left\langle \boxed{\begin{matrix} pre \\ ? \end{matrix}}_0 \ldots \boxed{goal_1(x) \wedge \phi(x)}_i \ldots \boxed{goal_2(x)}_j \ldots \boxed{goal}_n \right\rangle$$

Es werden hier Zwischenzustände partiell spezifiziert, die zur Lösung des Gesamtproblemes notwendig zu erreichen sind; im Zustand i muß etwa das Teilziel $goal_1(x)$ gelten mit der Randbedingung $\phi(x)$ an den Parameter x; nachfolgend muß noch ein Teilziel $goal_2(x)$ ebenfalls parametrisiert mit der gleichen Variablen x erreicht werden. σ' kann nun entsprechend den einzelnen Hilfszielen aufgesplittet werden zu:

$$\sigma'^1 \circ \sigma'^2 \circ \sigma'^3 = \left\langle \boxed{\begin{matrix} pre \\ ? \end{matrix}}_0 \ldots \boxed{goal_1(x) \wedge \phi(x)}_i \right\rangle \left\langle \boxed{\ }_0 \ldots \boxed{goal_2(x)}_{j-i} \right\rangle \left\langle \boxed{\ }_0 \ldots \boxed{goal}_{n-j} \right\rangle$$

Die Intervalle σ'^* können nun analog zu σ weiter verfeinert werden, z.B. gemäß:

$$\sigma'^{2'} = \left\langle \boxed{\ }_0 \ldots \boxed{goal_{21}}_u \ldots \boxed{goal_{22}}_v \ldots \boxed{goal_{23}}_w \ldots \boxed{goal_2(x)}_{j-i} \right\rangle$$

Es wird aus globaler Sicht betrachtet versucht, eine Abfolge von Aufgabenreduktionen und Aufsplittungen zu finden, so daß schließlich Intervalle entstehen, in denen durch Aktionsetablierungen die Modellexistenz abgesichert werden kann. Die eingeführten Hilfsziele sind auf den abstrakteren Verfeinerungsebenen hier nun nicht mehr durch Aktionsfokussierung entstanden, sondern vielmehr durch eine domänenabhängige *Aufgabenfokussierung.*
Die Repräsentation des Wissens über mögliche Aufgabenreduktionen geschieht durch die Formulierung entsprechender Axiome in LLP. Dem obigen Verfeinerungsschritt von σ nach σ' unterliegt ein Axiom

$$\Diamond[goal_1(x) \wedge \phi(x) \wedge \Diamond[goal_2(x) \wedge \Diamond goal]] \rightarrow \Diamond goal,$$

das wiederum einer Verfeinerungsregel

$$\frac{\Gamma \Rightarrow \Diamond[goal_1(x) \wedge \phi(x) \wedge \Diamond[goal_2(x) \wedge \Diamond goal]]}{\Gamma \Rightarrow \Diamond goal}$$

entspricht. Mithilfe solcher domänenspezifischer Aufgabenreduktionsverfeinerungen kann ein (Teil-)Planungsproblem geeignet strukturiert werden. Die zunächst eingeführten Teilaufgaben können nun, wie auch ebenfalls das ursprüngliche Ziel, durch abstrakte, d.h.

nicht unmittelbar durch Aktionen erreichbare, Eigenschaften beschrieben sein. Es muß aber schließlich Axiome geben, die eine Reduktion auf aktionsrelevante Eigenschaften vornehmen, d.h. es muß Verfeinerungsregeln

$$\frac{\Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Lambda}$$

geben, mit der Eigenschaft, daß alle in Δ vorkommenden Eigenschaften aktionsrelevant sind, Λ kann auch abstrakte Eigenschaften enthalten. Aktionsetablierungsverfeinerungen können nur in Intervallen ausgeführt werden, die allein durch aktionsrelevante Eigenschaften beschrieben sind; die Regressions- bzw. Progressionsprozesse bei der Bildung von Aktionsaxiomen haben ansonsten keine Semantik.

Neben diesen fixen, domänenspezifischen Aufgabenreduktionsverfeinerungen gibt es auch dynamisch während des Planungsprozesses entstehende Aufgabenreduktionen. Gemeint sind damit die bereits zu Beginn dieses Abschnittes angesprochenen, aktionsrelevanten Zielreduktionen. In den Abbildungen 4.16 und 4.23 wird die Regel *goal refine* parametrisiert verwendet. Der Parameter ist dabei das jeweils einzuführende Zwischenziel. Diese Regelverwendung stellt dabei eine Abkürzung für eine Folge zweier Verfeinerungen dar; einer spezifischen Aufgabenreduktion gefolgt von einer Verfeinerung durch die Regel *liveness split*. Dieses Verfahren ist im Hinblick auf eine praktische Operationalisierung des Beweisvorganges der Verwendung der Regel *goal refine* vorzuziehen. Beide Methoden führen zum selben Effekt.

Indirekte Plandetermination Steht Wissen über hierarchisch angeordnete Aufgabenreduktionen zur Verfügung, so kann es durchaus sinnvoll sein, Abstraktionen, die durch dieses Wissen ausgedrückt werden, auch in einem Plan zu reflektieren. Dies bedeutet, daß entsprechende Beschreibungsintervallinformation in einem Plan als Ergänzung zu Aktionen auftritt, oder diese in Teilen sogar vollständig ersetzt. Der letzt genannte Aspekt kann mithilfe der folgenden Planabstraktionsregel sehr einfach umgesetzt werden:

Regel 20 (plan abstraction)

$$\frac{pre,\ \Gamma \Rightarrow \Gamma \qquad pre,\ \mathsf{Q} \Rightarrow \Gamma}{pre,\ \mathsf{P} \Rightarrow \Gamma}$$

pre und Γ *sind LLP-Formeln,* P *und* Q *sind Planmetavariablen; es gibt bisher keine Einführungsregel mit Beteiligung der Variable* Q*. Es erfolgt die Substitution* $\{\Gamma/\mathsf{P}\}$*.* □

Durch die Anwendung dieser Regel wird der aus Planungssicht notwendige Nachweis der Modellexistenz, der zweite Antezedens der Regel (pre, $\mathsf{Q} \Rightarrow \Gamma$), abgetrennt von der konkreten Beschreibung eines relevanten Modelles als Plan; beim Beweis dieser Sequenz darf Regel 20 nicht verwendet werden. Eine mehr oder weniger abstrakte Aufgabenbeschreibung kann somit auch als Plan dienen, ohne eine konkrete Aktion zur Ausführung des Planes zu nennen. Man erhält somit eine intensionale Beschreibung eines konkreten Aktionsverhaltens, die für die jeweilige spätere Planverwendung von Vorteil sein kann.

Setzt man z.B. Regel 20 bei der Erstellung eines dichten linearen Planes erst direkt vor jeder Aktionsetablierungsverfeinerung ein, so erhält man Pläne der folgenden Art:

$$\bigcirc g_1 \wedge \odot\bigcirc F \ ; \bigcirc g_2 \wedge \odot\bigcirc F \ ; \ \ldots \ ; \bigcirc goal \wedge \odot\bigcirc F \ , \quad g_1,\ g_2, \ldots,\ goal \text{ seien modalfrei}$$

Ein solcher Plan beschreibt ein zeitliches Verhalten, indem ein *Pfad* zur Erreichung des spezifizierten Zieles über sukzessiv aufeinanderfolgende Teilziele führt. Da sich der Plan nur an Information aus dem entsprechenden Beschreibungsintervall bedient, enthält er jetzt im allgemeinen einen Nichtdeterminismus bzgl. der auszuwählenden Aktionen, die benötigt werden, um den Plan auszuführen. Des weiteren ist diese Option zur Aktionsauswahl aber im allgemeinen mit Unsicherheit behaftet. Es kann nicht notwendigerweise lokal entschieden werden, ob die aktuell ausgewählte Aktion das Erreichen später folgender Teilziele behindert, oder gar unmöglich macht. Auf solche Weise abstrahierte Pläne eignen sich also eher als Grundlage zur Beobachtung eines bestimmten Verhaltens, als daß sie zu einem deterministischen Plan konkretisiert werden.
Einen reinen Zielplan wie oben beschrieben kann man nun, gerade wenn er als Grundlage zur Beobachtung eines bestimmten Verhaltens dienen soll, auf einfache Art robuster gestalten. Die Darstellung $\ldots \ ; \bigcirc g_i \wedge \odot\bigcirc F \ ; \ \ldots$ verlangt nach expliziten Zustandsübergängen zwischen den einzelnen Zielen. Ausreichend kann jedoch auch sein, den Zwang des Zustandsüberganges in eine *Option* zum Zustandswechsel überzuführen. Dies bedeutet, daß das Erreichen aller Ziele g_i gefordert wird, daß aber gleichzeitig auch die Möglichkeit besteht, zwei aufeinanderfolgende Ziele g_j und g_{j+1} im gleichen Zustand zu erreichen. Dazu muß man nun die Einführung einer fixen Intervallänge vor einer Aktionsetablierung ersetzen durch die Einschränkung auf eine maximale Intervallänge. Ein Teilplan bzgl. des Teilzieles g_i wird dann formuliert als

$$\ldots \ ; \Diamond[g_i \wedge \bigcirc F] \wedge [\bigcirc F \vee \odot\bigcirc F] \ ; \ \ldots$$

Mit einem so aufgebauten Zielplan kann man nun Ausgangssituationen gerecht werden, die mehr Information, als bei der Generierung des Planes verfügbar war, enthält oder die Dynamik einer Domäne, die sich durch externe Ereignisse, die geforderte Ziele schon erreichen, ausdrückt, kann eher berücksichtigt werden.

Soll ein Plan nur durch die seine Entstehung triggernden, abstrakten Ziele angereichert werden, so kann dies wie folgt erreicht werden:

			Angewandte Regeln
(1)	$pre,\ \mathsf{P}$	$\Rightarrow \Diamond[goal \wedge \bigcirc F]$	P-*links*
(2)	$pre,\ \mathsf{P}';goal \wedge \bigcirc F$	$\Rightarrow \Diamond[goal \wedge \bigcirc F]$	*plan_causality*
(3)	$pre,\ \mathsf{P}'$	$\Rightarrow \Diamond[goal \wedge \bigcirc F]$	

goal sei hier eine modalfreie Formel, die ein abstraktes Teilziel ausdrückt.[73] Die Anwendung der *plan_causality*-Regel (vgl. Seite 184) dient auch hier dem Herbeiführen eines Zieles durch Aktionskausalität.
Die Grammatik syntaktisch zulässiger Pläne erweitert sich nun durch folgende Regeln:

[73] Das Vorgehen kann natürlich auch bei aktionsrelevanten Teilzielen angewandt werden.

$$\begin{array}{lll} \texttt{S\_Plan} & ::= & \texttt{S\_Plan} \ ; \ \texttt{MfF} \ \wedge \bigcirc F \ \mid \\ & & \Diamond[\texttt{MfF} \wedge \bigcirc F] \wedge [\bigcirc F \wedge \odot\bigcirc F] \ \mid \\ & & \bigcirc\texttt{MfF} \wedge \odot\bigcirc F \ \mid \\ & & \Diamond[\texttt{MfF} \wedge \bigcirc F] \end{array}$$

Wiederverwendung von Problemlösungen Bei den domänenspezifischen Aufgabenreduktionen wurden bisher nur solche diskutiert, die Verfeinerungen auf der Beschreibungsintervallebene vornehmen. Läßt man aber auch Verfeinerungen auf der Aktionsintervallebene zu, so befindet man sich unmittelbar im Gebiet der *Planwiederverwendung*[74]. Grundlage der Aufgabenreduktion ist hierbei ein Theorem der spezifizierten Domäne, d.h. eine bereits gelöste Problemspezifikation inklusive ihrer Lösung, z.B.

$$pre' \wedge P \rightarrow \Diamond goal \quad (P \text{ eine Planformel}).$$

Dieses Theorem entspricht wiederum einer Aufgabenreduktionsregel:

$$\frac{\Gamma \Rightarrow pre' \wedge P}{\Gamma \Rightarrow \Diamond goal}$$

Wendet man diese Regel auf eine aktuelle Problemspezifikation $pre \wedge \mathsf{Q} \Rightarrow \Diamond goal$ (Q sei eine Planmetavariable, *pre* eine modalfreie Formel) an, so bleibt als einziges zu lösendes Beweisproblem, die Aussage $pre \Rightarrow pre'$ zuzusichern, d.h. ob die Vorbedingungen der wiederzuverwendenden Lösung aus den aktuell geltenden Voraussetzungen herleitbar sind. Dies ist der einzig notwendige Schritt zum Nachweis der Modellexistenz; für die wiederverwendete Problemlösung wurde die Modellexistenz bereits früher bewiesen. Um eine *Korrelation* zwischen dem aktuellen Problem und einer vorhandenen Problemlösung herzustellen, muß also die Kompatibilität zwischen beiden Modellcharakterisierungen zugesichert werden. Man kann in diesem Fall auch von einer *erweiterten* Korrelationsverfeinerung sprechen. Im Zusammenhang mit nichtlinearen Plänen war die Kompatibilität zwischen den korrelierten Verfeinerungen automatisch gegeben, beide Verfeinerungen gingen von gleichen Voraussetzungen aus.

Aufgrund des verwendeten erweiterten Planbegriffes[75] ist es natürlich auch möglich, Lösungen wiederzuverwenden, die lediglich oder vor allem Beschreibungsintervallinformationen enthalten. Da sie Theoreme darstellen, ist die Modellexistenz nach erfolgreichem Kompatibilitätstest gesichert. Die Möglichkeit, auch von Aktionen abstrahierende Lösungen wiederzuverwenden, erweitert traditionelle Ansätze zur Planwiederverwendung.

4.2.8 Planinterpretation

Auf der Grundlage des Ansatzes, Planen als Modellverfeinerung anzusehen und Spezifikationen sowie Pläne objektsprachlich gleich zu behandeln, können Pläne, wie auch in Abschnitt 4.2.7 ausgeführt, sehr flexibel durch verschiedene Abstraktionen gekennzeichnet sein. Häufig dient die Erstellung eines Planes, auch eines abstrakten, dazu, letztendlich

[74] Vgl. etwa [Koehler 94], wo ein auf LLP basierter Ansatz zur deduktiven Planwiederverwendung auf einer eingeschränkten Klasse von Plänen und Problemspezifikationen vorgestellt wird.

[75] Pläne müssen nicht notwendigerweise Aktionen explizit beinhalten.

ein Verhalten in Form von konkreten Aktionen zu bestimmen, das dann zur Erreichung eines konkreten Zieles oder zur Erledigung einer bestimmten Aufgabe führt.
Ausgangspunkt war bisher alleine eine Planspezifikation von der Struktur

$$\forall x\ pre(x) \wedge \mathsf{Plan} \rightarrow goal(x)\ .$$

$pre(x)$ und $goal(x)$ charakterisierten dabei potentielle Aktionsintervalle, die dann durch eine konstruierte Instantiierung $P(x)$ für Plan mehr oder weniger streng eingeschränkt wurden; die Existenz mindestens eines Aktionsintervalles wurde dabei gesichert.
Die Situation für die Ausführung eines konkreten Planes ist ähnlich strukturiert. Es existiert dabei eine Beschreibung *init*, deren enthaltenen atomaren Formeln alle Grundformeln sind; sie beschreibt den konkreten Ausgangszustand. Eine Beschreibung des vom konkreten Plan cP zu erreichenden Zieles *real_goal* ist ebenfalls gegeben. Das Problem, diesen konkreten Plan zu finden, ist dann entsprechend spezifiziert als

$$init \wedge \mathsf{CPlan} \Rightarrow real\_goal.$$

Verwendet man das abstrakte, gelöste Planungsproblem $\forall x\ pre(x) \wedge P(x) \rightarrow goal(x)$ als Lemma, so ergibt sich nach Anwendung der *cut-Regel* das folgende Beweisproblem:

$$\forall x\ pre(x) \wedge P(x) \rightarrow goal(x),\ init \wedge \mathsf{CPlan} \Rightarrow real\_goal$$

Durch die Anwendung der Regeln $\forall$-*links* und $\rightarrow$-*links* entstehen die beiden offenen Sequenzen

$$(a)\quad init \wedge \mathsf{CPlan} \Rightarrow pre(t) \wedge P(t)\ \text{ und}$$
$$(b)\quad goal(t),\ init \wedge \mathsf{CPlan} \Rightarrow real\_goal$$

Um eine adäquate Anwendbarkeit des Planes $P(x)$ zu gewährleisten, muß die Substitution $\{t/x\}$ so gewählt werden können, daß die Teilprobleme $init \Rightarrow pre(t)$ und $goal(t) \Rightarrow real\_goal$ gelöst werden; erst dann ist das Lemma in der aktuellen Situation effektiv verwendbar. Gelten diese Voraussetzungen, so bleibt als eigentliche Planinterpretationsaufgabe die Sequenz

$$(I)\quad init \wedge \mathsf{CPlan} \Rightarrow P(t)$$

zu beweisen. Aus Sicht der Modellverfeinerung heißt dies, daß eine Verfeinerung CPlan gesucht wird, die die durch den aktuellen Ausgangszustand *init* und den (abstrakten) Plan $P(t)$ charakterisierten Modelle auf geeignete Weise weiter einschränkt. Man kann nun eine *inkrementelle* und eine *vollständige* Interpretation unterscheiden. Als Ergebnis einer vollständigen Interpretation entsteht eine LLP-Planformel cP als Instantiierung der Planmetavariable CPlan, so daß die Sequenz (I) bewiesen werden kann und die Anzahl der durch $init \wedge cP \Rightarrow P(t)$ beschriebenen Aktionsintervalle Eins beträgt. Bei einer inkrementellen Interpretation entsteht eine Instantiierung $cP_i;\mathsf{CPlan}_r$ für CPlan mit den Eigenschaften:

- es existiert mindestens ein Aktionsintervall, das durch $init \wedge cP_i;\mathsf{CPlan}_r \Rightarrow P(t)$ charakterisiert ist;
- es gilt nicht notwendigerweise $init \wedge cP_i \Rightarrow P(t)$;
- $init \wedge cP_i$ charakterisiert genau ein Aktionsintervall.

Eine Planinterpretationsaufgabe wird also demnach prinzipiell nach den gleichen Verfeinerungsprinzipien bearbeitet wie eine Planungsaufgabe. Die Unterscheidung liegt in der Art der zu verwendenden Verfeinerungen und den zu erwartenden Voraussetzungen; die Planinterpretation hat den Vorteil, daß die Modellexistenz bereits gesichert ist.
Typische Fälle für eine Planinterpretation sind z.B. die Auflösung eines Konditionales, die Konkretisierung einer intensionalen Beschreibung oder die Linearisierung eines nichtlinearen Planes. Betrachten wir dazu charakteristische Situationen.
Bei einem konditionalen Plan ergibt sich z.B. folgende Spezifikation:[76]

$$init,\ \mathsf{CPlan} \Rightarrow [\phi \wedge Ex(a)] \vee [\neg\phi \wedge Ex(b)]$$

In Abhängigkeit davon, ob ϕ oder $\neg\phi$ aus $init$ hergeleitet werden kann, ergibt sich $Ex(a)$ oder $Ex(b)$ als Instantiierung für CPlan.
Als Spezifikation zur Interpretation eines nichtlinearen Planes betrachten wir:

$$init,\ \mathsf{CPlan} \Rightarrow [T;pre_a \wedge Ex(a);T;pre_b \wedge Ex(b);T;goal_1 \wedge \bigcirc F] \wedge [T;pre_c \wedge Ex(c);T;goal_2 \wedge \bigcirc F].$$

Die unterbrechbare Sequenz der Aktionen a und b ist hier nebenläufig zu der Aktion c angeordnet. Durch temporallogische Transformationen wird zunächst eine Linearisierung vorgenommen, d.h. man erhält z.B.

$$init,\ \mathsf{CPlan} \Rightarrow T;pre_a \wedge Ex(a);T;pre_b \wedge Ex(b);T;goal_1 \wedge \bigcirc F;T;pre_c \wedge Ex(c);T;goal_2 \wedge \bigcirc F.$$

Danach erfolgt eine "Verdichtung", indem die temporalen Abstraktionen auf eine Intervallänge von Null konkretisiert werden:

$$init,\ \mathsf{CPlan} \Rightarrow pre_a \wedge Ex(a);pre_b \wedge Ex(b);goal_1 \wedge \bigcirc F;pre_c \wedge Ex(c);goal_2 \wedge \bigcirc F.$$

Nun kann eine Instantiierung von CPlan schrittweise vorgenommen werden und man erhält letztendlich:

$$init,\ Ex(a);Ex(b);Ex(c) \Rightarrow pre_a \wedge Ex(a);pre_b \wedge Ex(b);goal_1 \wedge \bigcirc F;pre_c \wedge Ex(c);goal_2 \wedge \bigcirc F.$$

Man verwendet dazu im wesentlichen die Regel 21, die eine Anpassung von Regel 1 an eine veränderte Zielspezifikation darstellt.

Regel 21 (chop split)

$$\frac{\phi,\ \mathsf{P}_1 \Rightarrow \omega_1 \wedge \Diamond[\bigcirc F \wedge \mathsf{pre}] \qquad \mathsf{pre},\ \mathsf{P}_2 \Rightarrow \omega_2}{\phi,\ \mathsf{P}_1;\mathsf{P}_2 \Rightarrow \omega_1;\omega_2}$$

□

[76] Es gelte wiederum die bereits eingeführte Abkürzung $Ex(a) \equiv ex(a) \wedge \odot\bigcirc F$.

Die Korrektheit von Regel 21 ergibt sich durch folgende Ableitung:

			Angewandte Regeln
(1)	$\phi,\ \mathsf{P}_1;\mathsf{P}_2$	$\Rightarrow \omega_1;\omega_2$	*Äquivalenz*
(2)	$[\phi \wedge\ \mathsf{P}_1];\mathsf{P}_2$	$\Rightarrow \omega_1;\omega_2$	*Modus Ponens*
(3)	$[\phi \wedge\ \mathsf{P}_1]$	$\Rightarrow \Diamond[\bigcirc F \wedge \mathsf{pre}]$	
(4)	$[\phi \wedge\ \mathsf{P}_1 \wedge \Diamond[\bigcirc F \wedge \mathsf{pre}]];\mathsf{P}_2$	$\Rightarrow \omega_1;\omega_2$	*Äquivalenz*
(5)	$[[\phi \wedge\ \mathsf{P}_1];[\bigcirc F \wedge \mathsf{pre}]];\mathsf{P}_2$	$\Rightarrow \omega_1;\omega_2$	*Äquivalenz*
(6)	$[\phi \wedge\ \mathsf{P}_1];[\mathsf{pre} \wedge \mathsf{P}_2]$	$\Rightarrow \omega_1;\omega_2$	*temp. Folgerung*
(7)	$[\phi \wedge\ \mathsf{P}_1];[\mathsf{pre} \wedge \mathsf{P}_2]$	$\Rightarrow [\omega_1;\mathsf{pre} \wedge \bigcirc F];\omega_2$	*;-Komposition*
(8)	$\phi \wedge\ \mathsf{P}_1$	$\Rightarrow \omega_1;\mathsf{pre} \wedge \bigcirc F$	*Äquivalenz*
(9)	$\mathsf{pre} \wedge \mathsf{P}_2$	$\Rightarrow \omega_2$	*Äquivalenz*
(10)	$\phi \wedge\ \mathsf{P}_1$	$\Rightarrow \omega_1 \wedge \Diamond[\mathsf{pre} \wedge \bigcirc F]$	

Es wird vorausgesetzt, daß pre eine modalfreie Formel bezeichnet.

Die Beweisführung der Planinterpretation folgt nun dem in Abbildung 4.29 skizzierten Prinzip: Die Interpretation von Plänen, die in notwendigen Teilen nur Information aus dem assoziierten Beschreibungsintervall enthalten (vgl. den Abschnitt 4.2.7.7), kann bei einer inkrementellen Interpretation zu Problemen führen, da das Finden eines Aktionsintervalles im allgemeinen nicht allein auf der Basis der Dekomposition des vorhandenen Planes geschehen kann. Eine typische Spezifikation sei gegeben als:

$$init,\ \mathsf{CPlan} \Rightarrow \circ g_1 \wedge \odot\bigcirc F;\circ g_2 \wedge \odot\bigcirc F.$$

Interpretation bedeutet hier nun eher Planen, denn Verfeinerungen durch Aktionsetablierungen müssen noch vorgenommen werden. Obgleich gesichert ist, daß ein Aktionsintervall existiert, kann z.B. eine Entscheidung, mit welcher Aktion das Teilziel g_1 etabliert werden soll, nur getroffen werden, wenn auch die Unbedenklichkeit ihrer Auswirkungen auf die Etablierung von g_2 gesichert ist.

Die durch den Adaptionspunkt $AP_{plantrans}$ (vgl. Abschnitt 3.3.1) markierten Methoden zur Planverfeinerung stellen eine abgeschwächte Form der Planinterpretation dar. Da sie Transformationen eines Planes (nach dessen Fertigstellung) darstellen, gehen sie ebenfalls von der gesicherten Modellexistenz eines Planungsproblemes aus. Hingegen findet die Interpretation auf abstraktem Niveau statt und fordert demnach nicht die eindeutige Charakterisierung eines einzigen Aktionsintervalles als Ergebnis der Planinterpretation. Ausgehend von einer Problemspezifikation $pre \wedge \mathsf{Plan} \Rightarrow goal$ wurde etwa eine Instantiierung P für Plan in Form eines unterbrechbaren nichtlinearen Planes als Lösung des Planungsproblemes gefunden. Wenn P dann etwa in einen dichten nichtlinearen Plan transformiert werden soll, heißt dies, eine Problemspezifikation $pre \wedge \mathsf{Q} \Rightarrow P$ durch entsprechende Methoden zu verfeinern. Die Interpretation ist hier notwendiger Teil der Plangenerierung.

Eine weitere Anwendung abstrakter Interpretation ergibt sich bei der Realisierung eines Optimierungsverfahrens durch die Kombination von regressiver Lösungsfindung und

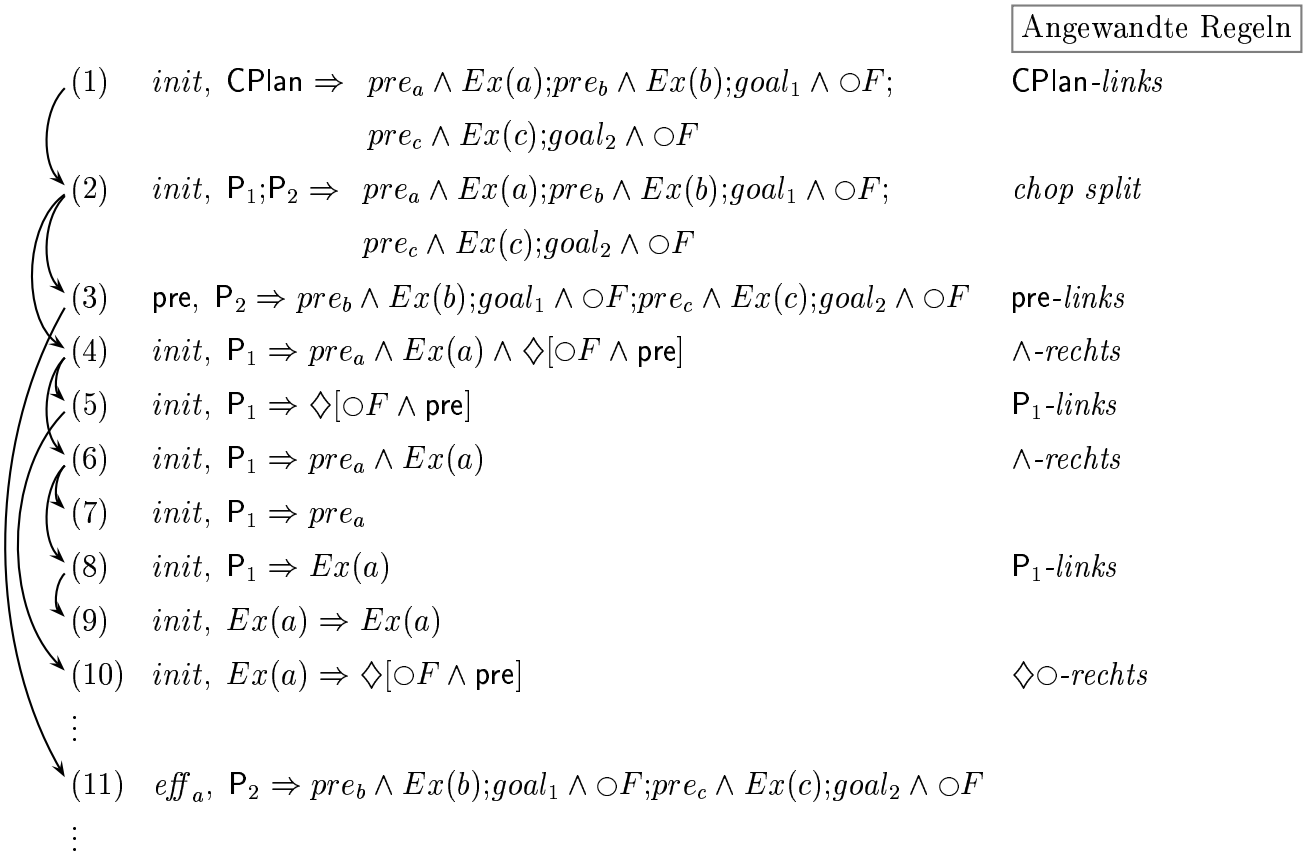


Abbildung 4.29: Beweisschritte einer Planinterpretation

progressiver Lösungsverbesserung wie auf Seite 159 beschrieben. Gegeben ein Planungsproblem $pre, \mathsf{P} \Rightarrow goal$ so erfolgt in einer ersten Phase die Lösung eines Problemes $pre, \mathsf{P}' \Rightarrow goal$ durch die auf Seite 159 beschriebene Methode zur Erstellung eines Zielplanes. Durch ein regressives Vorgehen bestimmt man dann eine Instantiierung $plan'$ für P' von der Form

$$\ldots\ ;\ \Diamond[g_i \wedge \bigcirc F] \wedge [\bigcirc F \vee \odot\bigcirc F]\ ;\ \ldots$$ [77]

Als progressiv durchzuführende, abstrakte Planinterpretationsaufgabe spezifiziert man nun das Problem $pre, \mathsf{P} \Rightarrow plan'$. Durch die vollständige Progression des Anfangszustandes und die Ausnutzung aller Effekte der der Progression zugrundeliegenden Aktionen ist es möglich, eine Verfeinerung $plan$ des Problemes zu finden, die die durch $plan'$ gegebene Maximallänge relevanter Aktionsintervalle reduzieren kann. Der Plan $plan$ kann dann, wenn er z.B. als Zielplan generiert wird, Teilpläne der Art

$$\ldots\ ;\ g_j \wedge \bigcirc F\ ;\ \ldots$$

[77] Es gebe für n Zwischenziele g_i analoge Teilpläne, d.h. man weiß, daß es als Lösung des Planungsproblemes einen dichten Plan bestehend aus n Aktionen gibt.

oder nochmals vereinfacht

$$\ldots \; ; \Diamond[g_{j-1} \wedge g_j \wedge \bigcirc F] \wedge [\bigcirc F \vee \odot\bigcirc F] \; ; \Diamond[g_{j+1} \wedge \bigcirc F] \wedge [\bigcirc F \vee \odot\bigcirc F] \; ; \ldots$$

enthalten. Damit haben sich Aktionen, die bei der Erstellung von $plan'$ als notwendig erachtet wurden, als redundant erwiesen.

4.2.9 Planverifikation und -validierung

Planverifikation bzw. eine ihrer Teilaufgaben, die Planvalidierung, sind ebenfalls Schlußfolgerungsprozesse, die mit den Vorgängen des Planens und Planinterpretierens eng zusammenhängen. Einer Planverifikationsaufgabe stehen als Voraussetzungen eine Planformel P_{Plan} und eine Planspezifikation $pre \wedge \mathsf{Plan} \rightarrow goal$ zur Verfügung. Das Ziel ist nun, zu beweisen, daß P_{Plan} eine Instantiierung für Plan ist, so daß die Planspezifikation erfüllt ist, d.h. P_{Plan} muß unter den Voraussetzungen pre ausführbar sein – die Validierung von P_{Plan} – und er muß das Ziel $goal$ erreichen.
Beschränken wir uns auf den Fall, daß $goal$ einer Formel $\Diamond[goal_0 \wedge \bigcirc F]$ entspricht, und $goal_0$ und pre modalfreie Formeln beschreiben. Planverifikation kann auf zwei prinzipielle Weisen betrieben werden, durch progressives und durch regressives Vorgehen (und durch Kombinationen davon). Bei einem progressiven Vorgehen kann man zur Lösung die beiden folgenden Beweisaufgaben formulieren:[78]

$$(1) \quad pre \wedge P_{\mathsf{Plan}} \Rightarrow \Diamond[\mathsf{end} \wedge \bigcirc F]$$
$$(2) \quad \mathsf{end} \Rightarrow goal_0$$

Beginnend mit Sequenz (1) versucht man die effektiven stärksten Nachbedingungen von pre bzgl. P_{Plan} als Instantiierung der Metavariablen end zu bestimmen. Wenn man aus ihnen das spezifizierte Ziel $goal_0$ folgern kann (Sequenz (2)), dann ist Plan P_{Plan} verifiziert. In entgegengesetzter Weise stellt ein regressives Vorgehen die beiden folgenden Aufgaben auf:[79]

$$(1) \quad \mathsf{init} \wedge P_{\mathsf{Plan}} \Rightarrow goal$$
$$(2) \quad pre \Rightarrow \mathsf{init}$$

Mit dem Beweis von Sequenz (1) versucht man die effektiven schwächsten Vorbedingungen von $goal_0$ bzgl. P_{Plan} als Instantiierung der Metavariablen init zu konstruieren. Wenn sie dann aus dem Ausgangszustand pre herleitbar sind (Sequenz 2), ist Plan P_{Plan} verifiziert. Möchte man Plan P_{Plan} nur validieren, d.h. seine Ausführbarkeit in der aktuellen Situation pre testen, so führt man im progressiven Fall den Beweis

$$pre \wedge P_{\mathsf{Plan}} \Rightarrow \Diamond[T \wedge \bigcirc F]$$

durch, d.h. man überprüft, ob das Planende durch die Anwendung progressiver Rahmenaxiome erreichbar ist. Im regressiven Fall beweist man entsprechend

$$(1) \quad \mathsf{init} \wedge P_{\mathsf{Plan}} \Rightarrow \Diamond[T \wedge \bigcirc F]$$
$$(2) \quad pre \Rightarrow \mathsf{init}$$

[78] Die direkte Aufteilung in zwei Aufgaben dient nur dem besseren Verständnis; sie sind implizit in der Beweisaufgabe $pre \wedge P_{\mathsf{Plan}} \Rightarrow \Diamond[goal_0 \wedge \bigcirc F]$ enthalten.

[79] Auch hier genügt eigentlich die Formulierung der Aufgabe $pre \wedge P_{\mathsf{Plan}} \Rightarrow \Diamond[goal_0 \wedge \bigcirc F]$.

d.h., gelten die schwächsten Vorbedingungen von T bzgl. P_{Plan} im aktuellen Zustand.
An einem einfachen Beispielplan wird nun diskutiert, wie das zur Verifikation notwendige Beweisvorgehen mit dem beim Planen angewandten Verfahren eng zusammenhängt. Sei der Plan P_{Plan} gegeben als $Ex(a);Ex(b)$ und unterstellen wir das regressive Vorgehen. Zu beweisen ist also zunächst die Sequenz

$$\mathsf{init} \wedge Ex(a);Ex(b) \Rightarrow \Diamond[goal_0 \wedge \bigcirc F].$$

Zur Beweisführung verwendet man im wesentlichen die Regel 22, die sich, wie auch schon Regel 21 zur Planinterpretation, als eine Anpassung von Regel 1 an veränderte Voraussetzungen der gleichen Problematik darstellt; die Korrektheit ergibt sich hier auch unmittelbar.

Regel 22 (goal verify)

$$\frac{\mathsf{pre}, P \Rightarrow \Diamond[\mathsf{pre}' \wedge \bigcirc F] \qquad \mathsf{pre}', Q \Rightarrow \Diamond[\phi \wedge \bigcirc F]}{\mathsf{pre}, P;Q \Rightarrow \Diamond[\phi \wedge \bigcirc F]}$$

pre *und* pre′ *sind hierbei Metavariablen und bezeichnen modalfreie Formeln,* P, Q *und* ϕ *Formeln aus LLP.*

□

Nach Anwendung von Regel 22 wird sinnvollerweise zunächst die rechte Antezedenssequenz weiter bearbeitet. Abbildung 4.30 zeigt wesentliche Schritte der Beweisführung:

Ohne die Themen der Planverifikation und der im letzten Abschnitt diskutierten Planinterpretation nun weiter zu vertiefen, was auch nicht Gegenstand der vorliegenden Arbeit ist, haben die kurzen Einführungen in die Problematik dennoch gezeigt, daß die für das Planen eingesetzten Verfeinerungstechniken durch geringfügige Anpassungen auch problemübergreifend eingesetzt werden können. Dies liegt nicht zuletzt an dem grundlegenden Prinzip, Aktionen, Spezifikationen und Pläne als syntaktisch und semantisch konforme Objekte – LLP-Formeln – zu repräsentieren.

4.2.10 Zusammenfassung

Im Verlauf der vorhergehenden Abschnitte wurden aufbauend auf dem Konzept des Planens als Modellverfeinerung auf der Grundlage eines verallgemeinerten Planbegriffes eine Vielzahl von Methoden eingeführt, den Planungsprozeß zu kontrollieren und Pläne mit spezifischen Merkmalen zu versehen. In Termini des semantischen Modelles kann der Verfeinerungsprozeß folgendermaßen charakterisiert werden:

Ausgangspunkt: Eine Planspezifikation in Form einer Menge von Randbedingungen an eine potentielle Menge von Aktionsintervallen; es sind Aussagen über bestimmte Zustände und Teilintervalle spezifiziert.

Ziel: Zum einen der Nachweis, daß solch eine Menge von Aktionsintervallen existiert und zum anderen eine im allgemeinen verfeinerte Charakterisierung dieser Menge durch zusätzliche Randbedingungen, interpretiert als Plan.

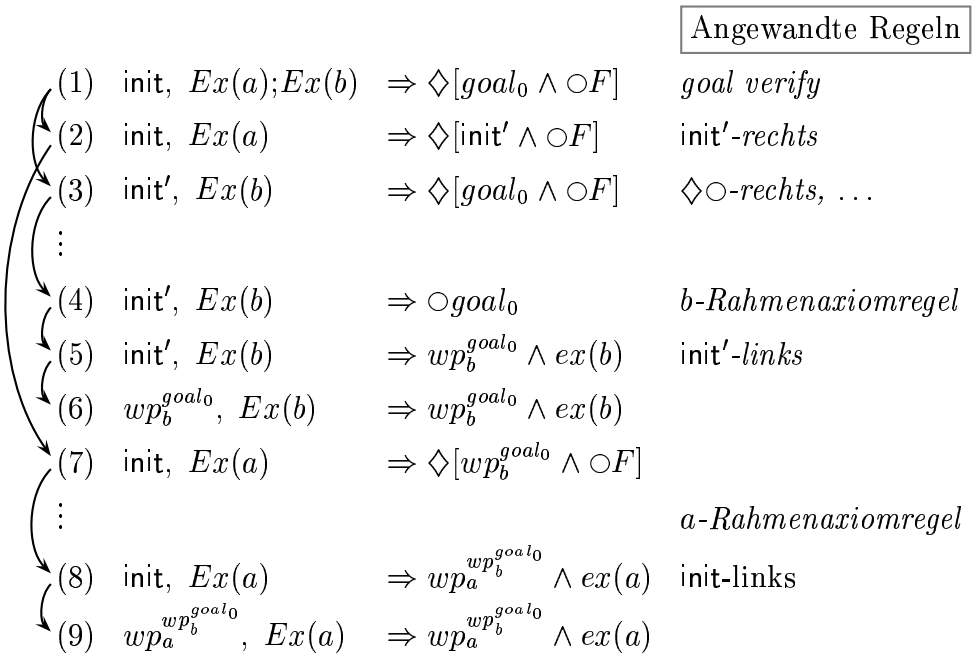


Abbildung 4.30: Skizzierung einfacher Planverifikationsschritte

Nebeneffekt: Es gibt ein Verfahren, um eine Aktionsintervallinstanz zu extrahieren.

Die folgende Übersicht faßt die verwendbaren Verfeinerungsarten zur Aktionsintervallcharakterisierung nochmals zusammen:

- Sequentialisierung von Teilintervallen
- Nichtdeterminismus zwischen Teilintervallen beliebiger Länge
- Einführung von Konditionalen
- Berücksichtigung partieller Information
- Berücksichtigung und Erzeugung von Sicherheitsbedingungen
- Nebenläufige (nichtlineare) Verfeinerung von Teilintervallen
- Berücksichtigung konditionaler Effekte
- Hierarchische Verfeinerung durch Aufgabenreduktion

Bei der freien Kombination dieser Verfeinerungsmöglichkeiten hat man nun noch jeweils verschiedene Optionen zur konkreten Ausprägung der Intervallcharakterisierung und zur Absicherung von Zustandsübergängen. Man kann wählen zwischen einer intensionalen Beschreibung von Teilintervallen ohne konkrete Benennung von Aktionen und einer extensionalen, bei der die intendierten Aktionen benannt werden; Mischformen sind ebenfalls

zulässig. Die semantische Fundierung von Zustandsübergängen, d.h. die durch axiomatisierte Aktionen bewirkte Veränderung einer Zustandsbeschreibung, kann entweder unter Einsatz von Progression oder durch Regression geschehen, jenachdem in welche zeitliche Richtung die Propagierung der semantischen Absicherung vorgenommen werden soll. Im gesamten Planungsprozess können progressives und regressives Vorgehen nahezu beliebig kombiniert werden.
Da der Planungsansatz fließende Übergänge zwischen Plänen und Spezifikationen erlaubt, können auch mehrstufige und inkrementelle Verfeinerungsprozesse beschrieben werden, z.B. zeitversetzte Plangenerierungs- und Interpretationszyklen. Die Homogenität des Ansatzes erlaubt es ebenfalls, die mit der Planung verwandten Prozesse der Planverifikation und Planvalidierung unmittelbar als spezielle Verfeinerungsprozesse zu modellieren und mit den Planungsmethoden zu bearbeiten.

Kapitel 5

Ein konfigurierbarer deduktiver Planer im IAS-Kontext

In Kapitel 4 wurde auf der Basis der Planungslogik LLP gezeigt, wie Planungsmechanismen als deduktive Prozesse in einem Sequenzenkalkül für LLP realisiert werden können. Der Durchführung einer planungsspezifischen Aufgabe entspricht dann die Führung eines speziellen Beweises. Es wird nun gezeigt, wie dieses Vorgehen mit den in Kapitel 3 eingeführten Werkzeugen zur Kontrollstrategiekonfiguration umgesetzt werden kann. Dazu betrachten wir zunächst die prinzipielle Abbildung des Deduktionsvorganges auf die in Kapitel 3 beschriebenen Kontrollstrategien.

5.1 Kontrollstrategien zur taktischen Beweissteuerung

Die in Abschnitt 3.3.2 eingeführte algebraische Spezifikation *Basic* führte die Basistypen und -funktionen zum Aufbau von Kontrollstrategien ein. Bildet man das deduktive Vorgehen als Instanz der Kontrollstrategiemechanismen ab, so ergibt sich für die Sorte *spec* die Korrespondenz mit einer Sequenz. Diese spezifiziert ein aktuelles Beweisproblem. Da bis zum Abschluß eines Beweises in der Regel Formeln aus LLP* verarbeitet werden, wird zur vollständigen Beschreibung eines Beweisproblems eine Sequenz mit einer Erweiterung um Information über verwendete Metavariablen und den assoziierten Substitutionen charakterisiert. Eine Sequenz beschreibt demnach einen Planungszustand im Sinne von Definition 3.2.1. Die Sorte *speclist* korrespondiert entsprechend mit einer geordneten Menge von Sequenzen.
Die durch $basic_i$ symbolisierten Basistransformationen repräsentieren die Anwendung von Sequenzenregeln, die Basisregeln des Kalküls darstellen und ebenso auch Theoreme und nichtlogische Axiome. Die Anwendung der Regel *goal refine* z.B. auf eine Sequenz S,[1] die zu einer Menge MS von zwei Sequenzen als neue Beweisprobleme führt, wird dann repräsentiert durch die Funktion $basic_{goal\ refine}(S, MS)$ oder kurz

$$\texttt{goal\_refine}(\texttt{S}, \texttt{MS}).$$

Gemäß der in Abschnitt 3.4.2 dargestellten Metaprogrammrealisierung von Kontrollstrategien bedeutet dies letztendlich auch, daß eine zweiargumentige PROLOG-Prozedur

[1] Von der konkreten Darstellung einer Sequenz abstrahieren wir hier zunächst.

`rel_goal_refine` zu definieren ist, die eine Transformation von S nach MS tatsächlich vornimmt.
Die durch die Spezifikation Tac und $Tacdef$ (vgl. auch Abschnitt 3.3.2) eingeführten Kontrollmechanismen erlauben die *Programmierung* des Einsatzes von Basistransformationen. Dies entspricht angewandt auf den Einsatz von Sequenzenregeln der Idee taktischer Theorembeweiser (vgl. Abschnitt 2.2.1). Die Hintereinanderausführung zweier Sequenzenregeln, z.B. $\Diamond\bigcirc$-*rechts* gefolgt von $\wedge$-*rechts*, auf eine Sequenz S resultierend in zwei Sequenzen MS ist repräsentiert durch den Term

$$\texttt{then}(\texttt{cons}(\texttt{sometimes\_next\_rechts}, \texttt{cons}(\texttt{and\_rechts}, \texttt{empty})), \texttt{S}, \texttt{MS}).$$

Wir geben nun beispielhaft Taktiken an, die eine einfache Strategie zur Erzeugung linearer Pläne unter der Verwendung eines rein progressiven Vorgehens wie auf den Seiten 140ff. beschrieben realisieren. Beschrieben sind sie in der für die Durchführung der Berechnung eines Taktikmoduls notwendigen Form.

```
rel_tacdef(simple_refine,
    orelse(cons(gentac(empty_goal(A),B),
           cons(selectapply_simple_progressive_goal(calltac(refine_goal,A,Y),
                                                    calltac(simple_refine,Y,B),A,B),
           empty)),A,B))

rel_tacdef(refine_goal,
    orelse(cons(calltac(empty_plan_sufficient,A,B),
           cons(calltac(single_action_sufficient,A,B),
           cons(calltac(insert_goal_split,A,B),
           empty))),A,B))


rel_tacdef(empty_plan_sufficient,
    then(cons(empty_plan_links(A,B),
         cons(sometimes_rechts(B,C),
         cons(repeat(and_rechts(C,D),C,D),
         cons(maptac(orelse(cons(ist_log_axiom(D,E),
                            cons(then(cons(repeat(orelse(
                                                  cons(and_links(I,J),
                                                  cons(and_rechts(I,J),
                                                  empty)),I,J),I,J),
                                      cons(maptac(ist_log_axiom(K,L),K,L),
                                      empty)),D,E),
                            cons(then(cons(meta_ax_inst_rechts(M,N),
                                      cons(ist_log_axiom(G,H),
                                      empty)),D,E),
                            empty))),D,E),D,E),
         empty)))),A,E))
```

Wir definieren dazu eine Taktik `simple_refine`, die die Hauptkontrolle übernimmt.
`empty_goal` ist formal eine Entscheidungsfunktion, d.h. eine Methode, um Taktiken zu bestimmen, die die im Parameter `A` gebundene Liste von Sequenzen gemäß einer gefällten Entscheidung weiterverarbeiten. Die Entscheidung ist hier ein Test, ob das *leere Ziel*, d.h. eine leere Liste, als Parameter erscheint. Im positiven Fall stellt die assoziierte PROLOG-Prozedur `rel_empty_goal` die Taktik `true` sonst die Taktik `false` als Entscheidungsergebnis zur Verfügung. Im `orelse`-Taktikkonstrukt wirkt der positive Test dann als Beendigung der rekursiven Verarbeitung von `simple_refine`.
Sind nun noch offene Ziele vorhanden, so wird mit Hilfe der Selektionsfunktion `simple_progressive_goal` die an `A` gebundene Liste von Sequenzen so umgeordnet, daß eine Problemspezifikation, die durch ein progressives Vorgehen verfeinert werden kann (vgl. Abbildung 4.14), am weitesten links in der Liste angeordnet wird. Das Taktikkonstrukt `selectapply` sorgt dafür, daß auf diese Sequenz die Taktikprozedur `refine_goal` zur Verfeinerung des aktuellen Problemes angewandt wird. Deren Ergebnis zusammen mit den verbleibenden nicht bearbeiteten Sequenzen bilden die Eingabe für die rekursive Interpretation von `simple_refine`.
Der Taktik `refine_goal` stehen drei wesentliche Methoden zur Verfeinerung einer progressiven Problemspezifikation zur Verfügung (vgl. die Abbildungen 4.15-4.17). Es sind dies die Einführung des leeren Planes, wenn die spezifizierten Ziele schon unmittelbar aus den Vorbedingungen ableitbar sind (die Taktik `empty_plan_sufficient`), die Einführung einer Aktion, deren Ausführung die spezifizierten Ziele direkt erreicht (die Taktik `single_action_sufficient`) und die Einführung eines weiteren Zwischenzieles zur Problemreduktion durch die Taktik `insert_goal_split`.
Exemplarisch ist die Taktik `empty_plan_sufficient` detaillierter beschrieben. Sie definiert eine Folge von Transformationsanweisungen, die alle erfolgreich sein müssen, damit die Taktik erfolgreich interpretiert werden kann. Sie beginnt z.B. mit der Anweisung zur Einführung des leeren Planes als Instantiierung der vorhandenen Planmetavariable (vgl. Abbildung 4.17 Sequenz (6*a*)). Die Basistransformation `ist_log_axiom` entspricht der Sequenzenregel zum Test, ob ein logisches Axiom vorliegt – die Möglichkeit zum Abschluß einer Beweisaufgabe im Sequenzenkalkül. Als Ergebnis stellt diese Transformation bei Erfolg eine leere Liste zur Verfügung; abgeschlossene Sequenzen stehen also nicht mehr weiter zur Verfügung.[2] Führt man nun gemäß Definition 3.3.3 eine Berechnung eines Taktikmodules $\mathbf{TM}_{\texttt{simple\_refine}}$ bzgl. der Parameter `S` und `?SL` durch, so ergibt sich bei erfolgreicher Berechnung und abgeschlossenem Beweis des in `S` enthaltenen Beweisproblemes eine Instantiierung `empty` für `?SL`.
Um Definition 3.3.4 gerecht zu werden, insbesondere um die entstandene Lösung eines Planungsproblemes auch nach *außen* verfügbar zu machen, ist es notwendig, den Beweisbaum nach Abschluß der Berechnung im Zugriff zu haben. Der Beweisbaum ist definiert durch die erfolgreiche Anwendung von Basistransformationen, also Sequenzenregeln, auf die initiale Problemspezifikation, die Wurzel des Baumes. Assoziiert man die Knoten des Baumes mit Sequenzen, so entspricht eine gerichtete markierte Kante $s \xrightarrow{r} s'$ der Beziehung, daß Sequenz s' aus Sequenz s durch die Anwendung von Regel r entstanden ist. Die Verantwortung zum Aufbau und der Aktualisierung des Beweisbaumes obliegt dementsprechend während der Berechnung eines Taktikmodules den die Basistransformationen

[2]Dies ist lediglich eine aus praktischen Erwägungen getroffene Designentscheidung.

ausführenden `rel_*`-Prozeduren. Sie realisieren dabei eine Funktion *update_tree* gemäß folgender Anforderung:
Ein Beweisbaum $\mathcal{T}$ sei gegeben als

$$\mathcal{T}(S, R) = \langle I, f, st_I, mv \rangle$$

S sei die Menge aller möglichen Sequenzen, R sei die Menge aller Sequenzenregeln. I ist eine Menge von Identifikatoren; die Funktion $st_I : I \rightarrow S$ realisiert eine Symboltabelle, die einer Sequenz einen eindeutigen Identifikator zuordnet. Die partielle Funktion $f : I \times I \rightarrow R$ definiere einen Baum über I mit gerichteten Kanten und Markierungen aus R. Die Menge mv enthalte alle Substitutionen für Metavariablen, die in Sequenzen aus dem Bildbereich von st_I enthalten sind.
Gegeben sei nun eine Sequenz s und eine Sequenzenregel r, s sei Konklusion von r. ms enthalte die Menge der Prämissen von r.[3] mi sei die Menge von durch r vorgenommenen Substitutionen für Metavariablen aus s. Die Funktion *update_tree* zur Aktualisierung eines Beweisbaumes kann nun wie folgt spezifiziert werden:

function $update\_tree(\mathcal{T}, s, r, ms, mi) \leftarrow$

if es gibt (i, s) im Graph von st_I

then $f' :=$ der Graph von f reduziert um den Teilbaum t mit Wurzel i;
$mv' := mv$ reduziert um alle Substitutionen für Metavariablen, die in durch t indizierten Sequenzen vorkommen;
$st'_I :=$ erweitere den Graph von st_I um Indizierungen für alle Sequenzen in ms[4]

else $st'_I :=$ erweitere den Graph von st_I um Indizierungen für alle Sequenzen in ms und s;
$f' := f$; $mv' := mv$

$mv'' := mv' \cup mi$;

$f'' := f' \cup \{(st_I^{-1}(m), st_I^{-1}(ms_k), r) \mid \forall ms_k \in ms\}$;

return $\langle I, f'', st'_I, mv'' \rangle$

Die Lösung für ein Planungsproblem kann nun nach erfolgreicher Berechnung einer Planungstaktik aus dem dann aktuellen Zustand von $\mathcal{T}$ extrahiert werden. Der Beweisbaum steht auch entsprechenden Visualisierungswerkzeugen zur Inspektion zur Verfügung (vgl. Abschnitt 5.5).

Die formale Spezifikation von Kontrollstrategien und die Spezifikation aller `rel_*`-Funktionen als logische Programme unterstützen den Einsatz formaler Programmverifikationsmethoden zur Überprüfung von Korrektheits- und Terminierungseigenschaften des implementierten Planungssystems.[5] Die verfügbaren Taktikkonstrukte sind alle so definiert, daß Fehlschläge bei der Interpretation einer Strategie automatisch dazu führen, zum letzten

[3] Hinweis: Sequenzenregeln werden zur Beweisführung rückwärts angewandt.
[4] Wir identifizieren hier eine Funktion mit ihrem Graphen.
[5] Dies ist jedoch nicht Gegenstand dieser Arbeit.

möglichen Auswahlpunkt alternativer Strategien zurückzusetzen; dort muß mindestens eine der Alternativen erfolgreich beendet werden, um mit der Berechnung fortfahren zu können. Unter der Annahme, daß alle Basistransformationen korrekt implementiert sind, kann die für das Planen wichtige Eigenschaft, daß durch die Berechnung einer Planungstaktik ein abgeschlossener Beweis resultiert, durch einen sehr einfachen Test über dem aktuellen Zustand von $\mathcal{T}$ geschehen: man muß nur überprüfen, ob alle Blattknoten durch die Anwendung der Regel `ist_log_axiom` entstanden sind.

5.2 Realisierung von Entscheidungsfunktionen

In Abschnitt 3.1.4 wurde das Konzept der Entscheidungsfunktionen zur Integration spezifischer Kontrollinformation in den Planungsprozeß zunächst abstrakt unabhängig von einem konkreten Planungsmechanismus eingeführt. Unter Berücksichtigung des in Kapitel 4 beschriebenen auf Modellverfeinerung basierenden deduktiven Planungsansatzes wird nun die Konkretisierung des Konzeptes, wie es durch die Taktikkonstrukte *gentac* und *multigentac* (vgl. Abschnitt 3.3.2) operationalisiert ist, näher erläutert. Entscheidungsfunktionen dienen in der in Abschnitt 3.1.4 vorgenommenen Definition dazu, eine bestehende Planverfeinerung weiter zu verfeinern oder zumindest eine folgende Verfeinerung gezielt vorzubereiten (wie z.B. im Falle der Umordnungen konjunktiver Ziele). Sie sind damit unmittelbar mit dem konkretisierten Modellverfeinerungskonzept vereinbar. Ihre allgemeine Philosophie besteht darin, gegeben ein bestimmtes Ziel und mehr oder weniger detaillierte Randbedingungen, nun auf eine Präferenzklasse von Plänen abzubilden. Diese Pläne können in der angenommenen allgemeinen Plansicht auch lediglich aus Zielen bestehen. Für die aktuelle Anwendung einer Entscheidungsfunktion innerhalb des Planungsprozesses setzen wir voraus, daß der durch das deduktive Vorgehen implizierte Beweisbaum (vgl. Abschnitt 5.1) verfügbar ist und insbesondere eine aktuell offene, primär planungsrelevante[6] Sequenz FS fokussiert ist. Charakterisieren wir diese Sequenz durch die Teilplanspezifikation $pre \wedge \mathsf{P} \rightarrow current\_goal$ und die initiale Planspezifikation RS durch $precond \wedge \mathsf{Plan} \rightarrow goal$. Der Graph einer planspezifischen Entscheidungsfunktion ef wird nun beschrieben durch eine Menge von Tupeln der Art

$$\langle goal_k, \phi_k, plan\_list_k \rangle$$

wobei wir annehmen, daß eine totale Ordnung zwischen den einzelnen Planalternativen in $plan\_list_k$ besteht und somit eine Liste als Repräsentation der geordneten Menge gewählt wird. $goal_k$ und ϕ_k sind bzw. $plan\_list_k$ enthält mit Variablen versehene Muster, wobei in $plan\_list_k$ vorkommende Variablen entweder in $goal_k$ oder in ϕ_k vorkommen müssen.
Betrachten wir die angenommenen verschiedenen Anwendbarkeitsbedingungen einer Entscheidungsfunktion bzgl. der Sequenz FS. Der Test auf Anwendbarkeit ist zweigeteilt:

1. *Ein verallgemeinerter Mustervergleich zwischen den relevanten Zielformeln*
   Hierzu muß folgende Zusicherung gelten:

$$goal'_k \rightarrow current\_goal, \quad \text{mit } goal'_k := \sigma(goal_k)$$

[6] Im Gegensatz etwa zu einer domänenunabhängigen temporallogischen Zusicherung.

σ beschreibt eine Substitution für in *goal_k* vorkommende freie Variablen; $goal'_k$ sei dann entsprechend die Formel, die sich durch Anwendung der Substitution auf $goal_k$ ergibt. Die Form der zu beweisenden Zusicherung deckt sowohl temporallogische als auch zustandsbezogene Schlußfolgerungen ab; beispielsweise hat man auf diese Weise die sich durch Kommutativität und Assoziativität logischer Operatoren ergebenden strukturellen Unterschiede äquivalenter Formeln abgedeckt.

2. *Verschiedene Alternativen zur Evaluierung der Randbedingung*
Die Randbedingung ϕ_k kann durch die Verfügbarkeit des aktuellen (partiellen) Beweisbaumes auf unterschiedliche Weise getestet werden. Wir beschränken uns dabei beispielhaft auf die Berücksichtigung der Sequenzen FS und RS und formulieren entsprechend folgende Varianten:

   (a) Evaluierung bzgl. der aktuellen Vorbedingungen in FS

   $$pre \rightarrow \phi''_k, \quad \text{mit } \phi''_k := \sigma_2(\sigma(\phi_k))$$

   σ_2 beschreibt hierin Substitutionen, die über die Instantiierungen in σ hinausgehen. Die Spezifikation in dem Entscheidungsfunktionengraph zur Kennzeichnung, daß die Evaluierung auf diese Weise erfolgen soll, geschieht über eine *Metatypisierung* von ϕ_k als

   $$\phi_k : \mathsf{current\_precondition}$$

   (b) Evaluierung bzgl. der aktuellen, globalen Vorbedingungen in RS:

   $$precond \rightarrow \phi''_k, \quad \text{mit } \phi''_k := \sigma_2(\sigma(\phi_k))$$

   Typisierung: ϕ_k:global_precondition

   (c) Evaluierung bzgl. des aktuellen Zieles in FS:

   $$current\_goal \rightarrow \phi''_k, \quad \text{mit } \phi''_k := \sigma_2(\sigma(\phi_k))$$

   Typisierung: ϕ_k:current_goal

   (d) Evaluierung bzgl. des globalen Zieles in RS:

   $$goal \rightarrow \phi''_k, \quad \text{mit } \phi''_k := \sigma_2(\sigma(\phi_k))$$

   Typisierung: ϕ_k:global_goal

   (e) Evaluierung bzgl. des aktuellen (partiellen) Planes:[7]

   $$\mathsf{Plan} \rightarrow \phi''_k, \quad \text{mit } \phi''_k := \sigma_2(\sigma(\phi_k))$$

   Typisierung: ϕ_k:current_global_plan

   Über die elementaren Randbedingungen (a)–(e) hinausgehend ist es durch die Anwendung boolscher Operatoren *und* und *oder* leicht möglich komplexere Randbedingungen zu formulieren.

[7] Die Durchführung weiterer Instantiierungen von möglicherweise vorhandenen Teilplanmetavariablen ist hier nicht erlaubt.

Ist die Anwendbarkeit gewährleistet, so enthält $plan\_list_k$ eine Liste von Verfeinerungen als vorgeschlagene Reaktionen auf die evaluierten Bedingungen. Für ein $m_k^i \in plan\_list_k$ wird die Reaktion durch die folgende Sequenzenregel beschrieben:

$$\frac{pre \wedge \mathsf{P} \Rightarrow m_k^{i\prime}}{pre \wedge \mathsf{P} \Rightarrow current\_goal} \text{ mit } m_k^{i\prime} := \sigma_2(\sigma(m_k^i)).$$

Damit die Reaktionen in den Taktikkonstrukten *gentac* und *multigentac* verwendet werden können, muß eine prozedurale Abstraktion vorgenommen werden. Dies bedeutet, daß Funktionen $basic_{k^i}$[8] gemäß der Spezifikation *Basic* (vgl. Abschnitt 3.3.2) im Sinne von Basistransformationen generiert werden. Die Operationalisierung $decfunc_{ef}$ der Entscheidungsfunktion ef zur Verwendung innerhalb eines Taktikkonstruktes liefert demnach als Ergebnis ihrer Auswertung eine Liste mit Elementen $basic_{k^i}(s, sl)$ analog den Planverfeinerungen $m_k^i \in plan\_list_k$.

5.3 Realisierung von Adaptionspunkten

In Abschnitt 3.3.1 wurden allgemein verschiedene Adaptionspunkte einer Planungsstrategie eingeführt, deren jeweilige Instantiierung zu einem unterschiedlichen Verhalten der Strategie führen kann und somit auch die Möglichkeit zu einer differenzierten Lösung des jeweils gleichen Problemes eröffnet. Abschnitt 4.2.4 beschrieb verschiedene Verfeinerungskategorien, durch deren flexible Kombination sich ein sehr hoher Freiheitsgrad für den Vorgang der Planerstellung ergibt. Wir diskutieren nun zunächst, welche der verfügbaren Modellverfeinerungen den vorgeschlagenen Adaptionspunkten zugeordnet werden können.

AP_{split} $\Leftrightarrow$
- Aufsplittung
- Disjunktion
- Nebenläufigkeit

Die genannten Verfeinerungsarten teilen alle eine bestehende Modellverfeinerung auf in mehrere neue Verfeinerungen.

AP_{foc} $\Leftrightarrow$ Selektion einer bestehenden Verfeinerung

Wenn mehrere (unterspezifizierte) Verfeinerungen im aktuellen Planungsprozeß bestehen, so erfolgt durch eine spezifische Methode die Selektion einer dieser Verfeinerungen mit dem Ziel, sie als nächstes weiter zu verfeinern.

Die folgenden Adaptionspunkte schränken jeweils bereits bestehende Modellverfeinerungen weiter ein.

[8] Inklusive ihrer prozeduralen Realisierung $rel\_basic_{k^i}$.

$$AP_{imp} \Leftrightarrow \left| \begin{array}{l} \text{Längeneinführung} \\ \text{Sicherheitsbedingungseinführung} \\ \text{Aufgabenreduktion} \\ \text{Korrelation} \\ \text{Determinierung} \end{array} \right.$$

Methoden, die eine Teilzielrealisierung unmittelbar vorbereiten oder herbeiführen, können den genannten Verfeinerungsarten zugeordnet werden, z.B. die Einführung einer Intervallängenbeschränkung, die Korrelation von Intervallen bei der Erstellung nichtlinearer Pläne, oder die Realisierung eines Problems durch Wiederverwendung einer erstellten Problemlösung.

$$AP_{probtrans} \Leftrightarrow \left| \begin{array}{l} \text{Aufgabenreduktion} \end{array} \right.$$

Die intendierten Transformationen und Umformulierungen von Teilproblemen können alle als vom Typ Aufgabenreduktion eingeordnet werden.

$$AP_{act} \Leftrightarrow \left| \begin{array}{l} \text{Aktionsetablierung} \end{array} \right.$$

Es handelt sich hier um verschiedene Methoden, eine Aktionsetablierung situationsbedingt umzusetzen.

$$AP_{plantrans} \Leftrightarrow \left| \begin{array}{l} \text{Längeneinführung} \\ \text{Korrelation} \\ \text{Sicherheitsbedingungseinführung} \\ \text{Determinierung} \end{array} \right.$$

Hier handelt es sich um die Nachbehandlung eines Planes, indem die ihn charakterisierenden Freiheitsgrade eingeschränkt oder teilweise eliminiert werden.

5.3.1 Die Merkmalszuordnung

Als nächstes ordnen wir nun den einzelnen Adaptionspunkten exemplarisch bestimmte, sie konkretisierende Merkmale zu, d.h. wir definieren die Mengen $features(AP_i)$ (vgl. Seite 78). Es wird ebenfalls beschrieben, welche der in Abschnitt 4.2 eingeführten deduktiven Planungsmechanismen jeweils bei einer Realisierung der Merkmale Verwendung finden und welche Einflüsse die Realisierung auf die Realisierung anderer Adaptionspunkte hat.
Beginnen wir mit der Auflistung relevanter Merkmale für den Adaptionspunkt AP_{split} (vgl. Abbildung 5.1).
Die ersten Drei indizieren die prinzipiellen Methoden zur Aufteilung einer Verfeinerung, durch Sequenzialisierung von Teilverfeinerungen (*sequential*), durch die Einführung nebenläufiger Verfeinerungen (*concurrent*) und durch die Etablierung alternativer Verfeinerungen (*plan_nondeterminism*). Von letzterer Methode sind der Aktionsnichtdeterminis-

AP_{split} **Problemaufteilung**
sequential
concurrent
plan_nondeterminism
action_nondeterminism
conditional_introduction
first_goal_single_state
complete_goal_single_state
optional_subgoal
weak_nonlinear_goal_determination
necessary_safety

Abbildung 5.1: Merkmale des Adaptionspunktes AP_{split}

mus (*action_nondeterminism*) durch seine lokale Beschränkung auf Intervalle der Länge 1 und die Einführung von Konditionalen (*conditional_introduction*) durch ihre nichtdeterministische Beschreibungsintervallspezifikation lediglich Spezialfälle. Die übrigen Merkmale sind nun lediglich Modifikatoren der Grundmethoden zur Aufsplittung.
first_goal_single_state bezeichnet den Fall, daß eine Zielaufspaltung so vorgenommen wird, daß das erste Konjunktionsglied einer konjunktiven Zielzustandsbeschreibung als Zielspezifikation einer neuen Verfeinerung auftritt (vgl. etwa Seite 137 die Regeln 2 und 3).
complete_goal_single_state symbolisiert Zielaufspaltungen im Sinne der Regeln 4 und 5, d.h. die Aufsplittung erfolgt an bereits in der Problemspezifikation markierten Zwischenzuständen[9].
Das Merkmal *optional_subgoal* bezeichnet eine Aufspaltung, wie sie in den Regeln 7 und 8 vorgenommen wird (vgl. Seite 150ff). Die aufgrund von Heuristiken bestimmten, nicht notwendigen Zwischenziele müssen hier nur optional erreicht werden, d.h. wenn sie sich während des Planens als überflüssig herausstellen (vgl. Abbildung 4.19), kann die für sie vorgesehene Teilintervallverfeinerung, ohne Probleme bei der Verfeinerungskohärenz zu erzeugen, entfallen.
Das Merkmal *weak_nonlinear_goal_determination* drückt die durch die Regel 6 beschriebene Methode zum Umgang mit nichtlinearen Zielen aus. Es wird gemäß der Regel rein syntaktisch entschieden, welches der Ziele fokussiert wird.
Schließlich beschreibt das Merkmal *necessary_safety* die notwendigen Anpassungen aller Aufsplittungsverfeinerungen zur Berücksichtigung notwendiger Sicherheitsbedingungen wie sie auf Seite 174 am Beispiel einer sequentiellen Aufsplittung beschrieben wurden.

Mit der Aufteilung eines Problems ist bei sequentieller Verarbeitung stets auch die Fo-

[9]Diese können als *Sollbruchstellen* angesehen werden.

kussierung eines der offenen Probleme zur weiteren Verarbeitung involviert. Die Basismerkmale des Adaptionsmerkmales AP_{foc} (siehe Abbildung 5.2) sind aus der vertretenen Planungssicht *progression* bzw. *regression* zum Ausdruck der progressiven bzw. regressiven Verfeinerungsstrategie.

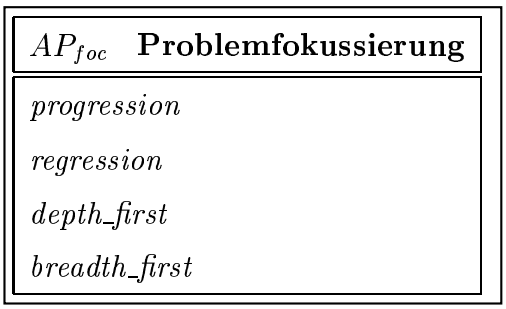


Abbildung 5.2: Merkmale des Adaptionspunktes AP_{foc}

Wie die damit assoziierten Auswahlmechanismen definiert sind, wurde in Abschnitt 4.2 detailliert beschrieben. Als spezialisierende Merkmale, die bei der Erzeugung nebenläufiger Pläne eine Rolle spielen, sind die Merkmale *depth_first* bzw. *breadth_first* anzusehen. Sie repräsentieren eine Teilproblemauswahl gemäß einer Tiefensuche- bzw. Breitensuchestrategie.
Wenn ein Teilproblem zur Bearbeitung ausgewählt ist, so besteht die Möglichkeit, es einer direkten Realisierung zuzuführen. Der Adaptionspunkt AP_{imp} ist durch eine Reihe von Merkmalen näher spezifizierbar, die alle zum Ziel haben, eine Teilzielrealisierung direkt vorzubereiten oder zu ermöglichen.
Das Merkmal *abstract_actions* repräsentiert hierbei das Standardvorgehen, die Erreichung eines Zieles mithilfe einer Aktion vorzubereiten. Das aktuelle Problem wird dabei geeignet in Zusicherungen und Kernprobleme zerlegt, so daß eine Aktionsetablierung möglich sein könnte (vgl. etwa Abbildung 4.15). Die Strategien zum Beweis der Zusicherungen sind hier ebenfalls angesiedelt. Die Realisierung dieses Merkmales muß den Adaptionspunkt AP_{act} enthalten.
action_verification repräsentiert die Strategie, eine vorgegebene Aktion bzgl. des spezifizierten Zieles zu verifizieren (vgl. auch Abschnitt 4.2.9). Dies kann auch während des Planens notwendig werden, wenn eine Teilzieltransformation durch eine entsprechende Aufgabenreduktionsverfeinerung Aktionen direkt eingeführt hat.
Das Merkmal *object_determination* spricht eine Methode zur Substitution von Objektmetavariablen an, die als Folge der Verwendung einer unterspezifizierten Aktion in Teilzielen oder Effekten aufgetreten sind (vgl. Abschnitt 4.2.5.2). Handelt es sich um unterspezifizierte Effekte, so muß hier auch der Beweis der mit den Metavariablen assoziierten noch nicht abgeschlossenen Zusicherungen aktiviert werden.
Das Merkmal *empty_plan* beschreibt die einfachste Möglichkeit ein Teilziel zu realisieren: dann, wenn dieses Ziel schon aus den Voraussetzungen unmittelbar ableitbar ist, genügt es, den leeren Plan zu etablieren (vgl. etwa Abbildung 4.17). Die drei folgenden Merkmale betreffen als Modifikationen von *empty_plan* die unterschiedlichen Methoden Sicherheitsbedingungen einzuführen (siehe entsprechend Abschnitt 4.2.7.4). Das Merkmal

AP_{imp} **Teilzielrealisierung**
abstract_actions
action_verification
object_determination
empty_plan
weak_safety
strong_safety
weak_safety_repair
explicit_action_context
explicit_abstract_goal_context
only_goal_plan
benevolent_only_goal_plan
reuse_library_solution
cache_solution
reuse_cached_solution
necessary_safety
subgoal_loop_check
single_action_goal

Abbildung 5.3: Merkmale des Adaptionspunktes AP_{imp}

weak_safety_repair verlangt in seiner Realisierung nach der Generierung eines Reparaturplanes, die entweder mit der allgemeinen Planungsstrategie vollzogen werden kann, oder aber auch mit einer speziellen Reparaturplanmethode[10]. Neben der Möglichkeit, eine fixe Methode zu verwenden, ist es ebenso möglich, für die Generierung von Reparaturplänen gesonderte Adaptionspunkte vorzusehen, um eine individuelle Reparaturplanerstellung zu gewährleisten.
Das Merkmal *explicit_action_context* sorgt gemäß dem in Abschnitt 4.2.7.6 beschriebenen Vorgehen dafür, daß der primäre, aktionsrelevante Kontext einer Aktion zusätzlich im Plan erscheint; es handelt sich dabei im allgemeinen um die schwächsten Vorbedingungen der Aktion bzgl. eines ihrer unmittelbaren Effekte.
Im Gegensatz zur Integration von Vorbedingungen beschreibt das Merkmal *explicit_abstract_goal_context* die Integration von Zielen, insbesondere abstrakten Zielen gemäß Aufgabenreduktionsverfeinerungen, in einen Plan (vgl. Abschnitt 4.2.7.7 Seite 197ff.).
Die konsequente Weiterführung der beiden letztgenannten Merkmale wird durch das Merkmal *only_goal_plan* erfaßt; seine Realisierung führt zur Beschreibung eines Planes alleine durch Ziele (vgl. ebenfalls das entsprechende Vorgehen auf Seite 197ff.). Eine Abwandlung, die zeitliche Flexibilität in die Abfolge dieser Ziele einführt (vgl. Seite 198), ist mit dem Merkmal *benevolent_only_goal_plan* assoziiert.
Auf Seite 199 wurde die Wiederverwendung von bereits erstellten Problemlösungen als eine Form der Aufgabenreduktionsverfeinerung eingeführt. Das Merkmal *reuse_library_solution* repräsentiert den Einsatz eines solchen Vorgehens in den Planungsprozeß. Seine Realisierung bildet dabei die Schnittstelle zu einem externen wie auch immer gearteten System zur Verwaltung und zum Wiederauffinden geeigneter Lösungen zu einem aktuellen Problem. Zu den geringsten Anpassungsproblemen führt hier natürlich ein Ansatz, der direkt kompatibel ist mit dem verwendeten logischen Problemrepräsentationsformalismus.
Eine spezielle Variante eines Systems zur Problemlösungswiederverwendung stellt ein *Caching*-System als Erweiterung der verwendeten Beweiskalkülimplementierung dar (vgl. etwa den *Caching*-Ansatz im Planungssystem ALPS [Calistri-Yeh & Segre 96]). Ein *Cache* stellt in gewissem Sinne eine integrierte *online*-Problemlösungsbibliothek von limitierter Größe dar. Ein *Cache* dient vorallem dazu, Lösungen zu wiederauftretenden gleichen Problemstellungen anzubieten, d.h. es gibt im Gegensatz zu einer allgemeinen Bibliothekslösung einen einfachen, aber schnellen Indizierungsmechanismus. Die Realisierung des Merkmales *cache_solution* ergänzt die gesamte Realisierung des Adaptionspunktes AP_{imp} um den Mechanismus, eine erfolgreiche Teilzielrealisierung dem *Caching*-System zuzuführen. Entsprechend realisiert das Merkmal *reuse_cached_solution* die Methode, im *Cache* eine Lösung des aktuell spezifizierten Problems zu finden und sie in eine aktuelle Problemlösung umzuwandeln.
Das Vorhandensein des Merkmales *necessary_safety* in einer aktuellen Adaptionspunktspezifikation beschreibt die notwendigen Anpassungen der übrigen Merkmale zur Berücksichtigung möglicherweise vorhandener Sicherheitsbedingungen als Teil der aktuellen Problemspezifikation.
Das ergänzende Merkmal *subgoal_loop_check* beschreibt eine spezielle Methode, die der Realisierung aller anderen Merkmale vorangestellt wird. Sie integriert die Funktion einer Schleifenerkennung bei der Realisierung eines Teilzieles, d.h. wenn das aktuelle Problem

[10] Die z.B. auf verschiedene Abstraktionen verzichtet.

als Teilproblem seiner früheren, noch nicht abgeschlossenen Bearbeitung selbst noch einmal auftritt, dann führt dies zum Scheitern der aktuellen Teilzielrealisierung.
Mit dem Merkmal *single_action_goal* erreicht man die explizite Einführung einer Aktion in den Plan nur unter einer bestimmten Bedingung: es wird gefordert, daß zur Erreichung des spezifizierten Zieles nur ein einziges Aktionsschema effektiv[11] verwendet werden kann. Ist diese Voraussetzung nicht erfüllt, so scheitert die mit dem Merkmal assoziierte Strategie.

Mit dem Adaptionspunkt $AP_{probtrans}$ sind nun Transformationen eines aktuellen Teilproblemes mit dem Ziel der Umformung oder Spezialisierung des Problemes intendiert. Ihre Realisierungen stellen spezifische Heuristiken dar, die teilweise den verfügbaren Strategien anderer Adaptionspunkte angepaßt sind. Abbildung 5.4 faßt mögliche Merkmale zusammen.

$AP_{probtrans}$ **Teilzieltransformation**
no_transformation
avoid_interact_goal_ordering
current_context_goal_ordering
common_producer_goal_ordering
static_dynamic_goal_ordering
plan_behavior_goal_ordering
plan_behavior_goal_extension
context_dependency
domain_task_reduction
nonlinear_goal_determination

Abbildung 5.4: Merkmale des Adaptionspunktes $AP_{probtrans}$

Das erste Merkmal *no_transformation* markiert lediglich das *Default*-Vorgehen, keine Transformationen vorzunehmen.
Durch das Merkmal *avoid_interact_goal_ordering* wird eine spezielle Entscheidungsfunktion zur Umsetzung planspezifischer Präferenzen beschrieben (vgl. Abschnitt 3.1.4.3). Sie setzt eine Methode zur Umordnung spezifizierter Teilziele um, die kompiliertes Aktionswissen gemäß der Funktion $ef_{erschwert}$ in Abschnitt 3.1.7 ausnutzt.
Die beiden Merkmale *current_context_goal_ordering* und *common_producer_goal_ordering* markieren ebenfalls denkbare Teilzielordnungsheuristiken, die im ersten Fall, die Ziele so anordnen, daß Ziele, die in der aktuellen Situation *leicht* erreichbar sind, zuerst ausgewählt werden können (vgl. die Entscheidungsfunktion $ef_{current\_context}$ in Abschnitt 3.1.4.3 auf Seite 57ff.). Das zweite der beiden Merkmale beschreibt eine Methode zur Umordnung,

[11] Die Aktion muß das Ziel als eigenen Effekt erzeugen können.

die ebenfalls auf kompiliertes Aktionswissen zurückgreift, hier assoziiert mit der Entscheidungsfunktion $ef_{synergy}$ aus Abschnitt 3.1.7. Das Merkmal *static_dynamic_goal_ordering* bezeichnet eine Zielordnungsheuristik, die domänenspezifisch *statische* Eigenschaften gegenüber *dynamischen* Eigenschaften präferiert. Durch sie erreicht man, daß durch Aktionen nicht beeinflußbare Fakten frühzeitig verwendet werden können, um den weiteren Verfeinerungsprozeß domänenspezifisch einzuschränken.
Das Merkmal *plan_behavior_goal_ordering* beschreibt nun eine Teilzielumordnung, die auf benutzerspezifischem Verhalten beruht und nicht wie in den drei vorhergehenden Merkmalen alleine aus der Domänenbeschreibung heraus ableitbar ist. Die assoziierte Entscheidungsfunktion bildet eine im Sinne von Abschnitt 3.1.4.3 planspezifische Präferenz ab, die die Integration von speziellem Benutzermodellwissen darstellt.
Das Merkmal *plan_behavior_goal_extension* schließt sich hier an und beschreibt eine Entscheidungsfunktion im Sinne von $ef_{planver}$, die eine über zeitliche Randbedingungen hinausgehende Verfeinerung des Zieles vornimmt. Diese Zielverfeinerung entspringt hier ebenfalls in einem Benutzermodell verwaltetem, typischen Verhalten.
Eine ebenfalls benutzerorientierte aber nicht notwendigerweise durch dessen bisheriges Verhalten ableitbare Transformation bzw. Verfeinerung eines Zieles wird durch das Merkmal *context_dependency* beschrieben (vgl. dazu die Ausführungen in Abschnitt 3.1.6). Die Erweiterung des Zieles berücksichtigt hierbei die gegenüber dem Planungssystem im allgemeinen lediglich partielle, dem Benutzer verfügbare Zustandsinformation. Die Erweiterung erfolgt so, daß Aktionen eingeführt werden, die dieses Defizit ausgleichen.
Durch die Realisierung des Merkmales *domain_task_reduction* geschieht die Integration von domänenspezifischen Aufgabenreduktionsverfeinerungen (vgl. Abschnitt 4.2.7.7) in den Planungsprozeß.
Mit dem Merkmal *nonlinear_goal_determination* ist eine Entscheidungsfunktion beschrieben, die eine Ordnung zwischen im Original ungeordneten Zielzuständen einführt, damit eine Struktur entsteht, die mit anderen bestehenden Verfeinerungsmethoden weiterverarbeitet werden kann (vgl. auch Abschnitt 139). Die eingeführte Ordnung kann z.B. unter Berücksichtigung des aktuellen Kontextes analog zum Merkmal *current_context_goal_ordering* geschehen.

Der Adaptionspunkt AP_{act} (vgl. Abbildung 5.5) beschreibt verschiedene aktionsspezifische Entscheidungsfunktionen (vgl. Abschnitt 3.1.4.1) und Modifikatoren, die eine Aktionsetablierung triggern.
Die ersten fünf Merkmale symbolisieren unterschiedliche, lokal operierende Spezialfälle der Entscheidungsfunktion ef_{akt}. *action_index* beschreibt eine allgemeine Indizierung von Einzelzielen durch alle sie produzierende Aktionen. *user_action_index* beschreibt eine Teilmenge dieser Indizierung, die einer benutzerspezifischen, einem Benutzermodell extrahierten allgemeinen Präferenzstruktur hinsichtlich der Zuordnung von Zielen zu Aktionen entspricht. Die Merkmale *class_common_goals* und *user_class_common_goals* beschreiben entsprechende Varianten, in denen die Präferenzordnung zwischen einzelnen Aktionen durch die Quantität ihrer Effekte bestimmt ist. Das Merkmal *class_execution_costs* schließlich ordnet Aktionen qualitativ unter Berücksichtigung ihrer spezifischen Ausführungskosten an.
Das Merkmal *global_new_action_bound* realisiert eine Protokollierung der Verwendung von

[12] n ist dabei eine natürliche Zahl.

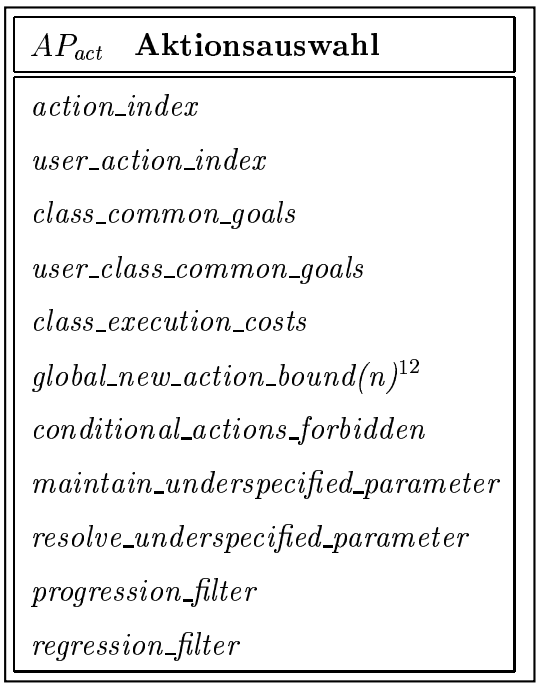


Abbildung 5.5: Merkmale des Adaptionspunktes AP_{act}

Aktionen[13], die gemäß der benutzerspezifisch indizierten Entscheidungsfunktionen nicht verfügbar wären, also im Benutzermodell als dem Benutzer nicht bekannte oder von ihm nicht verwendete Aktionen gekennzeichnet sind. Durch eine globale Beschränkung der Anzahl dieser *neuen* Aktionen kann ein Plan nach tutoriellen Gesichtspunkten erstellt werden; es kann verhindert werden, daß er durch dem Benutzer nicht geläufige Aktionen überladen ist.
Durch das Merkmal *conditional_actions_forbidden* kann ein zusätzlicher Filter über den für den Planungsprozeß verfügbaren Aktionen eingeschaltet werden, der nur Aktionen ohne konditionale Effekte passieren läßt. Die Erlaubnis, diese Aktionen verwenden zu können, kann sich durchaus negativ auf die Transparenz des erstellten Planes auswirken (vgl. die Ausführungen in Abschnitt 4.2.7.6).
Die beiden darauffolgenden Merkmale beeinflussen die Art und Weise, wie bei der Bildung eines Aktionsaxioms mit der Instantiierung von Variablen umgegangen wird, für die aus der aktuellen Problemspezifikation kein geeigneter Substitutionspartner bestimmt werden kann. *maintain_underspecified_parameter* beschreibt hierbei das Vorgehen, die Instantiierungsentscheidung bis zu einem geeigneten Zeitpunkt aufzuschieben.
resolve_underspecified_parameter beschreibt die Alternative, durch eine entsprechende Heuristik eine Instantiierung direkt festzulegen (vgl. dazu auch Abschnitt 4.2.5 Seite 129ff.).
Das Merkmal *progression_filter* filtert unter Aktionen, die ein bestimmtes Ziel erreichen, diejenige heraus, die bzgl. des in der aktuellen Progressionsphase verfügbaren Anfangszustandes möglichst wenige offene Vorbedingungen entstehen läßt. Analog dazu filtert die Realisierung des Merkmales *regression_filter* die Aktion heraus, die bzgl. der aktuellen

[13]Abstrahierend von der jeweiligen Instantiierung.

Zielspezifikation möglichst viele offene Ziele explizit erreichen kann.
Wie auf Seite 202 beschrieben sind die durch $AP_{plantrans}$ gruppierten Merkmale (siehe Abbildung 5.6) spezielle Formen der Planinterpretation. Das Merkmal *dense_plan* beschreibt eine Methode zur Transformation eines längenvariablen Planes in einen bzgl. notwendiger Aktionen dichten Plan. Der längenvariable Plan charakterisiert implizit Aktionen, die nicht notwendiger Bestandteil des Planes sind.

$AP_{plantrans}$ **Plantransformation**
dense_plan
redundant_action_elimination
weak_to_strong_safety
safety_considering_reaction_on_event
safety_explicit_disturbing_actions
linearisation
necessary_safety
explicit_necessary_safety

Abbildung 5.6: Merkmale des Adaptionspunktes $AP_{plantrans}$

Durch das Merkmal *redundant_action_elimination* wird die auf Seite 202ff. beschriebene Kombinationsmethode zur Längenoptimierung eines Planes beschrieben.
Die drei folgenden Merkmale beschreiben Methoden, die sich mit Plänen, angereichert durch optionale Sicherheitsbedingungen, befassen. Das Merkmal *weak_to_strong_safety* beschreibt die triviale Umformung eines Planes mit schwachen Sicherheitsbedingungen in einen mit starken Sicherheitsbedingungen. *safety_considering_reaction_on_event* bezeichnet eine Interpretationsmethode zur Anreicherung eines Planes, der "einfache" Sicherheitsbedingungen enthält, durch Reparaturpläne für möglicherweise auftretende Verletzungen dieser Bedingungen (vgl. dazu das mit Regel 11 assoziierte Vorgehen, s. Seite 173). Das Merkmal *safety_explicit_disturbing_actions* beschreibt die Transformation einer starken Sicherheitsbedingung in eine Form, die alle Aktionen, die diese Bedingung verletzen könnten, explizit macht (vgl. die Ausführungen auf Seite 173 über erweiterte Sicherheitsbedingungen).
Das Merkmal *linearisation* beschreibt die abstrakte Interpretation eines nichtlinearen Planes zu einem rein sequentiellen Plan. Die Strategie zur Erstellung eines nichtlinearen Planes wurde dann offensichtlich nur als unterstützende Methode verwendet, um letztendlich zu einem linearen Plan zu gelangen.
Mit dem Merkmal *necessary_safety* werden wie schon zuvor bei anderen Adaptionspunkten Anpassungen der spezifizierten Realisierungen an das Vorkommen notwendiger Sicherheitsbedingungen gekennzeichnet.
Das Merkmal *explicit_necessary_safety* beschreibt den einfachen Vorgang einer äquivalenten Umformung, die eine notwendige Sicherheitsbedingung als solche nochmals in den

Plan *kopiert*.
Zu dem Adaptionspunkt $AP_{plantrans}$ gehören auch alle Methoden, die syntaktische Vereinfachungen von Plänen beschreiben, z.B. die Elimination oder Zusammenfassung von Sicherheitsbedingungen in nichtlinearen Plänen, wenn sie dort keine semantische Funktion mehr erfüllen (vgl. die Erläuterungen auf Seite 189).

Problemaufteilung

	sequential	concurrent	plan_nondeterminism	action_nondeterminism	conditional_introduction	first_goal_single_state	complete_goal_single_state	optional_subgoal	weak_nonlinear_goal_determination	necessary_safety
sequential		$\rightsquigarrow 1$	$\rightsquigarrow 1$	< 1	< 1	$\wedge 1$	$\wedge 1$	$\wedge 1$	$\wedge 1$	$\wedge 1$
concurrent			$\rightsquigarrow 1$							$\wedge 1$
plan_nondeterminism	< 1 ;2 $\rightsquigarrow 1$					$\wedge 1$	$\wedge 1$			$\wedge 1$
action_nondeterminism	< 1				< 1					$\wedge 1$
conditional_introduction	< 1 ;1									$\wedge 1$
first_goal_single_state	$\wedge 2$		$\wedge 2$				< 1	$\wedge 1$		
complete_goal_single_state	$\wedge 2$		$\wedge 2$			< 1 ;1		$\wedge 1$		
optional_subgoal	$\wedge 2$					$\wedge 2$	$\wedge 2$			
weak_nonlinear_goal_determination	$\wedge 2$									
necessary_safety	$\wedge 2$	$\wedge 2$	$\wedge 2$	$\wedge 2$	$\wedge 2$					

Tabelle 5.1: Zulässige Merkmalskombinationen für AP_{split}

5.3.2 Intendierte Merkmalskombinationen

In der Spezifikation *Adapt* in Abschnitt 3.4.1 sind eine Reihe von Operatoren genannt, mit deren Hilfe Kombinationen von Adaptionsmerkmalen gebildet werden können. Da nicht alle Merkmale mit allen Operatoren sinnvoll kombinierbar sind, legt die Funktion *valid_combination* fest, welche Merkmale innerhalb eines Adaptionspunktes mithilfe der Operatoren verknüpft werden können und wie das Ergebnis weiter verwendet werden kann. Wir strukturieren den Graphen der Funktion *valid_combination*, indem wir für jeden Adaptionspunkt A eine Matrix m^A anlegen, die die Funktion partiell wie folgt beschreibt. Jede Spalte und jede Zeile von m^A werden durch ein Adaptionspunktmerkmal eindeutig indiziert. Ist nun $(op(m_i, m_j), m_i)$ bzw. $(op(m_i, m_j), m_j)$, m_i, m_j Merkmale von A, Element des Graphen von *valid_combination*, so enthält das Feld $m^A(m_i, m_j)$ einen Eintrag $op1$ bzw. $op2$. In den nachfolgenden Tabellen werden zur besseren Übersicht Abkürzungen für die Kombinationsoperatoren von Merkmalen verwendet:

$$
\begin{array}{lcl}
< & : & \mathit{prefer\_to} \\
; & : & \mathit{before} \\
\vee & : & \mathit{some\_of} \\
\wedge & : & \mathit{and} \\
\rightsquigarrow & : & \mathit{adapt\_to\_when}
\end{array}
$$

Tabelle 5.1 enthält die Matrix für den Adaptionspunkt AP_{split}. Mit dem Operator *adapt_to_when* wird die Veränderung von Merkmalen in Abhängigkeit von dem Gelten einer bestimmten Bedingung beschrieben. Für die definierbaren Bedingungen wird gefordert, daß sie bei einer gegebenen aktuellen Verfeinerung entschieden werden können; wir definieren die drei folgenden einfach zu entscheidenden Bedingungen:

- *initial_plan_length(n)*: Diese Bedingung ist erfüllt, wenn aus der Vervollständigung des aktuell zu verfeinernden Teilintervalles σ_a die Existenz eines notwendigen initialen Aktionsintervalles von mindestens der Länge n unmittelbar folgt. Wenn σ das Gesamtintervall des zu erstellenden Planes charakterisiert, so ist σ_a als Teilintervall von σ durch sequentielle, nebenläufige oder disjunktive Aufsplittung entstanden. Als Vervollständigung von σ_a sehen wir nun die Intervallcharakterisierung σ' an, die sich von σ nur dadurch unterscheidet, daß eine mögliche, σ_a unmittelbar umgebende, disjunktive Verfeinerung auf das Disjunkt reduziert wird, das σ_a als Teilintervall enthält.

- *terminal_plan_length(n)*: Hier wird analog die Existenz eines notwendigen terminalen Aktionsintervalles von mindestens der Länge n gefordert.

- *complete_plan_length(n)*: Hier wird eine bisher bestehende notwendige Gesamtlänge von n gefordert.

Als Beispiel einer syntaktisch zulässigen Spezifikation von AP_{split} ergibt sich gemäß Tabelle 5.1 etwa:

adapt_to_when(*and*(sequential,*prefer_to*(complete_goal_single_state,
first_goal_single_state)),
and(concurrent,*and*(complete_goal_single_state,
optional_subgoal)),
initial_plan_length(5))

Hiermit legt man fest, daß solange eine sequentielle Problemaufteilung stattfindet, bis sich ein initiales notwendiges Aktionsintervall mit der Länge 5 ergeben hat; dannach werden nur noch nebenläufige Aufteilungen vorgenommen. Die sequentielle Verfeinerung präferiert eine globale vor einer lokalen Zielstrukturierung. Die nebenläufige Verfeinerung führt eine globale Zielstrukturierung durch und behandelt neu eingeführte Zwischenziele als optional und nicht notwendig zu erreichen.

Tabelle 5.2 beschreibt eine mögliche Kombinationsmatrix für den Adaptionspunkt AP_{foc}. Als Beispiel einer Kombination sei spezifiziert:

Problemfokussierung

	progression	regression	depth_first	breadth_first
progression		$\rightsquigarrow 1$ < 1	$\wedge 1$	$\wedge 1$
regression	$\rightsquigarrow 1$ < 1	$\rightsquigarrow$	$\wedge 1$	$\wedge 1$
depth_first	$\wedge 2$	$\wedge 2$		
breadth_first	$\wedge 2$	$\wedge 2$		

Tabelle 5.2: Zulässige Merkmalskombinationen für AP_{foc}

adapt_to_when(progression,
and(regression,depth_first),
initial_plan_length(5))

Diese Spezifikation harmoniert mit der obigen Beispielspezifikation von AP_{split} und beschreibt nun zusammengenommen, daß der Anfang des Planes bis zu einer notwendigen Länge von 5 progressiv geplant wird und danach der Rest des Planes regressiv; die Auswahl des Problems, für das eine nebenläufige Verfeinerung vorgenommen wird, geschieht dabei gemäß der *Tiefezuerst*-Suchstrategie.

Teilzielrealisierung

	abstract_actions	action_verification	object_determination	empty_plan	weak_safety	strong_safety	weak_safety_repair	explicit_action_context	explicit_abstract_goal_context	only_goal_plan	benevolent_only_goal_plan	reuse_library_solution	cache_solution	reuse_cached_solution	necessary_safety	subgoal_loop_check	single_action_goal
abstract_actions			;1					∧1					;1		∧1	∧1	
action_verification	<1														∧1		
object_determination	;2											;2		;2			
empty_plan	<2									<2	<2	<2		<2		∧1	<2
weak_safety															∧1		
strong_safety															∧1		∧2
weak_safety_repair				<1	<1	<1									∧1		
explicit_action_context	<2									∧1	∧1						
explicit_abstract_goal_context	<2							∧1									
only_goal_plan	;2							∧1					;1		∧1	∧1	
benevolent_only_goal_plan	;2							∧1					;1		∧1	∧1	
reuse_library_solution	<1									<1	<1		;1		∧1	∧1	
cache_solution																	
reuse_cached_solution	<1									<1	<1	<1			∧1	∧1	<1
necessary_safety	∧2	∧2			∧2	∧2	∧2			∧2	∧2	∧2		∧2			∧2
subgoal_loop_check	;2 ∧2			;2 ∧2						;2 ∧2	;2 ∧2	;2 ∧2		;2 ∧2			;2 ∧2
single_action_goal	<2		;1			∧1		∧1		<2	<2		;1		∧1	∧1	

Tabelle 5.3: Zulässige Merkmalskombinationen für AP_{imp}

Als Beispiel, einer in Tabelle 5.3 beispielhaft beschriebenen Kombinationsmatrix für den Adaptionspunkt AP_{imp} betrachten wir:

before(subgoal_loop_check,*prefer_to*(empty_plan,
prefer_to(reuse_library_solution,
and(abstract_actions,
explicit_action_context))))

Hiermit spezifiziert man, daß dem Versuch einer Teilzielrealisierung, zunächst ein Test auf zyklisches Teilzielvorkommen vorausgeht. Die eigentliche Realisierung präferiert die Realisierung des Problems durch Einführung des leeren Planes[14] gegenüber der Wiederverwendung einer möglicherweise füher einmal erstellten Lösung und gegenüber dem Versuch, das Problem durch die Anwendung einer Aktion zu reduzieren. Kommt die Aktionsreduktion zum Zuge, so macht sie den aktuellen Aktionskontext im Plan explizit.

Tabelle 5.4 enthält das Beispiel einer Matrix zum Adaptionspunkt $AP_{probtrans}$. Als typisches Beispiel seiner Spezifikation betrachten wir:

prefer_to(*before*(current_context_goal_ordering,
avoid_interact_goal_ordering),
no_transformation)

Beschrieben wird die Präferenz einer kombinierten Zielordnungsstrategie gegenüber dem Nichtdurchführen einer Transformation. Damit verhindert man das Scheitern der mit $AP_{probtrans}$ verbundenen Strategie, wenn sich kein geeigneter Kandidat durch die Ordnungsverfahren ergibt.
Tabelle 5.5 enthält Beispiele zur Kombination von Merkmalen des Adaptionspunktes AP_{act}. Eine mögliche Spezifikation ist gegeben als:

and(*prefer_to*(class_common_goals, action_index),
resolve_underspecified_parameter)

Es ist eine Präferenz gegenüber zwei Indizierungen der verfügbaren Aktionen spezifiziert; die Aktionsetablierung geschieht aber in beiden Fällen so, daß durch den aktuellen Kontext unterspezifizierte Variablen dennoch sofort instantiiert werden.

Schließlich beschreibt Tabelle 5.6 Merkmalskombinationen bzgl. einer nachfolgenden Plantransformation.

5.3.3 Die Umsetzung in Taktiken

Nachdem die zulässigen Merkmale und ihre Kombinationsmöglichkeiten definiert sind, muß nun gemäß der auf Seite 80ff. beschriebenen Klassifikation spezifiziert werden, wie ihre Umsetzung in Taktiken bzw. Anpassungen von Adaptionspunkten geschehen soll; die Klassifizierung erfolgt in die unterschiedlichen Ersetzungsregeln 1(a)–2(f). Die Gesamtheit der Regelinstanzen bildet dann das System $ETeS_{Adapt}$ (vgl. Seite 84), das die Basis der operationalen Taktikkonfigurierung bildet.

[14] Damit eine unnötige Verfeinerung vermieden wird.

Teilzieltransformation										
	no_transformation	avoid_interact_goal_ordering	current_context_goal_ordering	common_producer_goal_ordering	static_dynamic_goal_ordering	plan_behavior_goal_ordering	plan_behavior_goal_extension	context_dependency	domain_task_reduction	nonlinear_goal_determination
no_transformation		< 2	< 2	< 2	< 2	< 2			< 2	
avoid_interact_goal_ordering	< 1		< 1	< 1	< 1	< 1	< 1			$\wedge 1$
current_context_goal_ordering	< 1	;2 < 1		;2 < 1	;2 < 1		;2	;2		$\wedge 1$
common_producer_goal_ordering	< 1	< 1	$\wedge 1$ < 1		< 1	< 1				$\wedge 1$
static_dynamic_goal_ordering	< 1	< 1	< 1	< 1		< 1	< 1			$\wedge 1$
plan_behavior_goal_ordering	< 1	< 1	< 1	< 1	< 1		< 1			$\wedge 1$
plan_behavior_goal_extension	< 1							< 1		$\wedge 1$
context_dependency	< 1						;1 < 1		< 1	$\wedge 1$
domain_task_reduction	< 1						< 1	< 1		$\wedge 1$
nonlinear_goal_determination	< 1	;2	;2	;2	;2	;2	;2	;2	;2	

Tabelle 5.4: Zulässige Merkmalskombinationen für $AP_{probtrans}$

Aktionsauswahl

	action_index	user_action_index	class_common_goals	user_class_common_goals	class_execution_costs	global_new_action_bound(n)	conditional_actions_forbidden	maintain_underspecified_parameter	resolve_underspecified_parameter	progression_filter	regression_filter
action_index	$\rightsquigarrow 1$ $\wedge 1$;1	$\rightsquigarrow 1$		$\rightsquigarrow 1$		$\wedge 1$	$\wedge 1$	$\wedge 1$	$\wedge 1$	$\wedge 1$	$\wedge 1$
user_action_index	< 1 $\rightsquigarrow 1$	$\rightsquigarrow 1$ $\wedge 1$;1	< 1 ;1	<1	< 1 ;1		$\wedge 1$	$\wedge 1$	$\wedge 1$	$\wedge 1$	$\wedge 1$
class_common_goals	<1	<1	$\rightsquigarrow 1$ $\wedge 1$;1		< 1 ;1	$\wedge 1$	$\wedge 1$	$\wedge 1$	$\wedge 1$	$\wedge 1$	$\wedge 1$
user_class_common_goals	< 1 $\rightsquigarrow 1$	<1	<1	$\rightsquigarrow 1$ $\wedge 1$;1	< 1 ;1		$\wedge 1$	$\wedge 1$	$\wedge 1$	$\wedge 1$	$\wedge 1$
class_execution_costs	<1	< 1 ;1	< 1 ;1	< 1 ;1	$\rightsquigarrow 1$ $\wedge 1$;1	$\wedge 1$	$\wedge 1$	$\wedge 1$	$\wedge 1$	$\wedge 1$	$\wedge 1$
global_new_action_bound(n)	$\wedge 2$		$\wedge 2$		$\wedge 2$					$\wedge 1$	$\wedge 1$
conditional_actions_forbidden	$\wedge 2$	$\wedge 2$	$\wedge 2$	$\wedge 2$	$\wedge 2$					$\wedge 1$	$\wedge 1$
maintain_underspecified_parameter	$\wedge 2$	$\wedge 2$	$\wedge 2$	$\wedge 2$	$\wedge 2$					$\wedge 1$	$\wedge 1$
resolve_underspecified_parameter	$\wedge 2$	$\wedge 2$	$\wedge 2$	$\wedge 2$	$\wedge 2$			<1		$\wedge 1$	$\wedge 1$
progression_filter $\wedge 2$	$\wedge 2$	$\wedge 2$	$\wedge 2$	$\wedge 2$	$\wedge 1$	$\wedge 1$	$\wedge 1$	$\wedge 1$;1	$\wedge 1$;1	$\wedge 1$	$\wedge 1$
regression_filter $\wedge 2$	$\wedge 2$	$\wedge 2$	$\wedge 2$	$\wedge 2$	$\wedge 1$	$\wedge 1$	$\wedge 1$	$\wedge 1$;1	$\wedge 1$;1	$\wedge 1$	$\wedge 1$

Tabelle 5.5: Zulässige Merkmalskombinationen für AP_{act}

Plantransformation								
	dense_plan	redundant_action_elimination	weak_to_strong_safety	safety_considering_reaction_on_event	safety_explicit_disturbing_actions	linearisation	necessary_safety	explicit_necessary_safety
dense_plan							∧1	;1
redundant_action_elimination							∧1	;1
weak_to_strong_safety					;1		∧1	;1
safety_considering_reaction_on_event							∧1	;1
safety_explicit_disturbing_actions							∧1	;1
linearisation	;2	;2		;2	;2		∧1	;1
necessary_safety	∧1	∧1	∧1	∧1	∧1	∧1		
explicit_necessary_safety	∧2	∧2	∧2	∧2	∧2	∧2		

Tabelle 5.6: Zulässige Merkmalskombinationen für $AP_{plantrans}$

Im folgenden werden nun einige Regelinstanzen gemäß den in den Abbildungen 5.1–5.6 definierten Merkmalen beschrieben.
Die Umsetzung von Basismerkmalen geschieht zunächst im wesentlichen durch die Spezifikation von Taktiken, die die in Abschnitt 4.2.7 eingeführten Strategien umsetzen. In Abschnitt 5.1 auf Seite 209 wurde beispielsweise eine Taktik eingeführt, die in vereinfachter Form überprüft, ob eine Problemspezifikation alleine durch den *leeren* Plan schon erfüllt werden kann. Ein solche Taktik kann dann als Realisierung des Merkmales *empty_plan* für den Adaptionspunkt AP_{imp} (vgl. Abbildung 5.3) dienen. Eine Regelinstanz gemäß dem Schema 1(a) ergibt sich dann als

$$feature\_tac(empty\_plan) \rightarrow calltac(empty\_plan\_sufficient, A, B)$$

Instanzen des Schemas 1(a) existieren u.a. für Merkmale der Menge $\{empty_i | i \in AP\_NEC\}$, d.h. zu jedem Adaptionspunkt existiert ein *leeres* Merkmal, das die Irrelevanz dieses Adaptionspunktes für eine aktuelle Konfigurierung ausdrückt. Die Realisierung erfolgt dann gemäß:

$$feature\_tac(empty_i) \rightarrow true$$

Die zweite Variante, ein Basismerkmal zu realisieren, geschieht durch die Angabe einer konditionalen Ersetzungsregel gemäß Schema 1(b). Ein Beispiel für eine Instantiierung ergibt sich bei der Realisierung des Merkmales *abstract_actions* im Adaptionspunkt AP_{imp}. Umgesetzt wird dieses Merkmal durch zwei leicht unterschiedliche Strategien (vgl. die Abbildungen 4.15 und 4.23) in Abhängigkeit davon, ob die Verfeinerung durch Progression oder Regression geschieht.
Regelinstanzen ergeben sich dann in der Form:

$$\begin{aligned}
&current_features(AP_{foc}) \rightarrow M \Rightarrow \\
&\quad nec_subterm(M) \rightarrow progression \Rightarrow \\
&\qquad feature_tac(abstract_actions) \rightarrow calltac(abstract_action_progression, A, B) \\
&current_features(AP_{foc}) \rightarrow M \Rightarrow \\
&\quad nec_subterm(M) \rightarrow regression \Rightarrow \\
&\qquad feature_tac(abstract_actions) \rightarrow calltac(abstract_action_regression, A, B)
\end{aligned}$$

Betrachtet man die Umsetzungen von Merkmalskombinationen, wie sie in den Tabellen 5.1–5.6 aufgeführt sind, ergeben sich beispielsweise folgende Regelinstanzen. Bzgl. des Adaptionspunktes AP_{imp} ergibt sich z.B. gemäß Schema 2(a) die Instanz:

$$\begin{aligned}
&feature_tac(prefer_to(action_verification, abstract_actions)) \rightarrow \\
&\quad orelse(cons(feature_tac(action_verification), feature_tac(abstract_actions)), \\
&\qquad A, B)
\end{aligned}$$

Für den Adaptionspunkt AP_{split} kann die Kombination

$$and(necessary\_safety, sequential)$$

spezifiziert sein. Ihre Umsetzung kann ebenfalls durch eine Instanz von Schema 2(a) erreicht werden:

$$feature\_tac(and(necessary\_safety, sequential)) \rightarrow$$
$$then(cons(feature\_tac(necessary\_safety), feature\_tac(sequential)),$$
$$A, B)$$

Die Realisierung der Merkmalspezifikation $feature\_tac(necessary\_safety)$ kann dann durch eine Regelinstanz des Schemas 1(a), die auch **extend**-Anweisungen – also eine Propagierung einer Adaptionspunktanpassung – miteinbezieht, gegeben sein:

$$feature\_tac(necessary\_safety) \rightarrow \quad calltac(safety\_distribution, A, B) \text{ und}$$
$$(\mathbf{extend}(AP_{imp}, necessary\_safety),$$
$$\mathbf{extend}(AP_{plantrans}, necessary\_safety))$$

Durch diese Form der Realisierung kann man erreichen, daß alle Adaptionspunkte, die durch das Vorhandensein notwendiger Sicherheitsbedingungen beeinflußt werden, automatisch daran angepaßt werden.
Ein weiteres Beispiel des Adaptionspunktes AP_{split} zeigt eine Instanz gemäß Schema 2(b):

$$feature\_tac(and(sequential, optional\_subgoal)) \rightarrow$$
$$feature\_tac(sequential\_os)$$

Die Eigenschaft *optional_subgoal* dient hier lediglich zur weiteren Spezialisierung der sequentiellen Aufsplittung (vgl. auch Abbildung 4.19) und die *and*-Verknüpfung verweist in diesem Sinne auf ein spezialisiertes Merkmal.
Mit dem Regelschema 2(c) hat man die Möglichkeit, eine Merkmalskombination aufgrund zwingender Erfordernisse automatisch umzudefinieren. Bezüglich des Adaptionspunktes AP_{imp} ergibt sich z.B.:

$$feature\_tac(and(abstract\_actions, subgoal\_loop\_check)) \rightarrow$$
$$feature\_tac(before(subgoal\_loop\_check, abstract\_actions))$$

Um den mit dem Merkmal *subgoal_loop_check* intendierten Test zur Erkennung eines zyklischen Verhaltens in der Teilzielrealisierung effektiv einzusetzen, ist es notwendig, ihn vor der weiteren Ausführung einer Teilzielverfeinerung durchzuführen.

Regelinstanzen des Schemas 2(d) ergeben sich, wenn man z.B. den Adaptionspunkt AP_{act} betrachtet (vgl. Tabelle 5.5). Die meisten der dort definierten Merkmale sind assoziiert mit einer bestimmten Entscheidungsfunktion. In der einfachsten Art ist solch eine Entscheidungsfunktion z.B. eingebunden in ein parametrisierbares Taktikkonstrukt *multigentac* (vgl. die algebraische Spezifikation *Tac*, Seite 68ff.), etwa in der Form:

$$multigentac(decfunc_{user\_action\_index}(A), B, true)$$

Eine *and*-Kombination der Merkmale *user_action_index* und *progression_filter* kann dann gemäß Schema 2(d) realisiert werden:

$$feature\_tac(and(user\_action\_index, progression\_filter)) \rightarrow$$
$$multigentac(intersection(decfunc_{user\_action\_index}(A), decfunc_{progression\_filter}(A)),$$
$$B, true)$$

Betrachten wir ein Beispiel für Schema 2(e), das eine Verallgemeinerung der Schemata 2(a) und 2(b) darstellt. Für den Adaptionspunkt AP_{act} kann folgendes spezifiziert sein:

$$and(global\_new\_action\_bound(5), action\_index)$$

Eine Realisierung ergibt sich z.B. durch

$$feature\_tac(and(global\_new\_action\_bound(5), action\_index)) \rightarrow$$
$$iftac(new\_action\_bound\_5\_not\_reached(A, A),$$
$$then(cons(feature\_tac(action\_index), new\_action\_logging(B, B)), A, B),$$
$$false)$$

Hierin seien *new_action_bound_5_not_reached* und *new_action_logging* spezielle Basisfunktionen der Spezifikation *Basic*. Die letztere der beiden Funktionen protokolliert die erfolgreiche Verwendung einer neuen Aktion in einem zusätzlichen, die Basisrepräsentation erweiternden, globalen Speicher des aufgebauten Beweisbaumes (vgl. Seite 211).[15] Die Funktion *new_action_bound_5_not_reached* testet den speziellen Speichereintrag auf die Einhaltung der globalen Beschränkung, hier auf die Einhaltung des Maximalwertes von 5.

Das Regelschema 2(f) findet bei selbstadaptiven Adaptionspunktspezifikationen Anwendung. Betrachten wir den Adaptionspunkt AP_{imp} mit einer Spezifikation:

adapt_to_when(abstract_actions,
adapt_to_when(*and*(abstract_actions,weak_safety),
before(only_goal_plan,abstract_actions),
terminal_plan_length(5)),
initial_plan_length(5))

Die Intention dieser Spezifikation ist, eine mit fortschreitender Dauer des Planes ansteigende Abstraktion zu erreichen. Die ersten fünf Aktionen des Planes sollen "normale" abstrakte Aktionen sein, dannach folgen Aktionen, die durch schwache Sicherheitsbedingungen unterbrochen sind und schließlich erscheinen die letzten fünf Schritte im Plan nur noch in Form von Zielen[16] Um nun die Bedingung *initial_plan_length(5)* auswerten zu können, ist es erforderlich, daß der Plan zu Beginn *progressiv* verfeinert wird. Analog folgt bzgl. der Bedingung *terminal_plan_length(5)*, daß am Planende eine *regressive*

[15]Bei *Backtracking* ist die Funktion auch für die Wiederherstellung des Originalzustandes verantwortlich.

[16]Vorausgesetzt, daß der Plan überhaupt die Länge 10 erreicht.

Verfeinerung vorgenommen werden muß. Dies bedeutet, daß die jeweilige Realisierung des Adaptionspunktes AP_{imp} eine bestimmte Realisierung des Adaptionspunktes AP_{foc} verlangt. Mit den verfügbaren Regelschemata kann man dies folgendermaßen erreichen. Unter Verwendung des Schemas 2(f) definiert man Regeln

$$\begin{aligned}
&cond_type(C) \rightarrow initial_plan_length \Rightarrow \\
&\quad feature_tac(adapt_to_when(abstract_actions, F, C)) \rightarrow \\
&\qquad applytac(iftac(cond_real(C), \\
&\qquad\qquad feature_tac(F), \\
&\qquad\qquad feature_tac(feature_mod(abstract_actions, initial_plan_length))), \\
&\qquad\quad A, B)
\end{aligned}$$

$$\begin{aligned}
&cond_type(C) \rightarrow terminal_plan_length \Rightarrow \\
&\quad feature_tac(adapt_to_when(F_1, F_2, C) \rightarrow \\
&\qquad applytac(iftac(cond_real(C), \\
&\qquad\qquad feature_tac(F_1), \\
&\qquad\qquad feature_tac(feature_mod(F_2, terminal_plan_length))), A, B)
\end{aligned}$$

Für den Graph der Funktion $feature\_mod$ ergeben sich die Tupel:

$$\langle abstract\_actions, initial\_plan\_length, abstract\_actions\_ipl \rangle$$

$$\langle before(A, B), terminal\_plan\_length, before(feature\_mod(A, terminal\_plan\_length), feature\_mod(B, terminal\_plan\_length)) \rangle$$

$$\langle only\_goal\_plan, terminal\_plan\_length, only\_goal\_plan\_tpl \rangle$$

$$\langle abstract\_actions, terminal\_plan\_length, abstract\_actions\_tpl \rangle$$

Das Merkmal $abstract\_actions\_ipl$ stellt ein abstraktes Basismerkmal dar, das gemäß Schema 1(a) realisiert wird in der Form:

$$\begin{aligned}
feature_tac(abstract_actions_ipl) \rightarrow \quad & feature_tac(abstract_actions) \text{ und} \\
& (\mathbf{extend}(AP_{foc}, progression))
\end{aligned}$$

Die Realisierung erfolgt hierin durch die Abbildung des abstrakten Merkmales auf ein konkretes Basismerkmal und zusätzlich durch die Propagierung einer notwendigen Anpassung des Adaptionspunktes AP_{foc}.

Zur Vervollständigung der Voraussetzungen für die dynamische Konfigurierung einer Planungstaktik fehlt noch die Angabe einer totalen Ordnung $<_{AP}$, die Prioritäten zwischen den vorgesehenen Adaptionspunkten definiert (vgl. auch Seite 85). Eine beispielhafte Ordnung ist gegeben durch:

$$AP_{plantrans} <_{AP} AP_{act} <_{AP} AP_{probtrans} <_{AP} AP_{imp} <_{AP} AP_{foc} <_{AP} AP_{split}$$

Diese Anordnung verleiht z.B. dem Adaptionspunkt AP_{split} die höchste Priorität.

Um eine Taktikkonfiguration durchführen zu können, benötigt man zum Abschluß noch ein Taktikmodul, das Adaptionspunkte enthält. In Anlehnung an das in Abbildung 3.4 dargestellte allgemeine Planerschema kann die übergeordnete Planungstaktik kompakt auch wie folgt aufgebaut werden (vgl. auch Seite 95):

```
rel_tacdef(planner,
    then(cons(iftac(calltac(no_open_goals,A,B),
                    true,
                    then(cons(choicetac(APfoc),
                         cons(orelse(cons(choicetac(APsplit),
                                     cons(choicetac(APprobtrans),
                                     cons(choicetac(APimp),
                                     empty))),B,C),
                         cons(calltac(planner,C,D),
                         empty))),B,D)),
         cons(choicetac(APplantrans),
         empty)),A,E))
```

Realisierungen für den Adaptionspunkt AP_{imp} werden je nach Situation zu Taktiken führen, die den Adaptionspunkt AP_{act} enthalten, also prinzipiell in folgender Form erscheinen:

```
rel_tacdef(tacx,
    then( ...
          choicetac(APact),
          ...
    ))
```

5.4 Beispiele von Planervarianten

Wir gehen wie in Abschnitt 1.1.2 motiviert davon aus, daß das Planungssystem eine Komponente eines intelligenten Assistenzsystemes ist. Als ein erstes Beispiel betrachten wir die Situation, daß für die interaktive Anwendung "*Kalendermanagement*" gemäß einer bestimmten Planspezifikation ein Plan generiert werden soll, der als Eingabe für ein Planerkennungssystem dienen soll; dort übernimmt er die Rolle einer Hypothese bzgl. des möglichen Verhaltens des Benutzers zur Erreichung des prognostizierten Zieles. Die Kopplung von Plangenerierungs- und erkennungsprozessen auf einer gemeinsamen logischen Basis zur Forcierung von intelligenter Assistenzleistung eines Hilfesystemes wurde erfolgreich im Projekt PHI (*Planbasierte Hilfesysteme*) [Bauer et al. 93] demonstriert. Die Verwendung insbesondere abstrahierter Pläne zur Unterstützung und Ansteuerung eines Planerkennungsprozesses ist in [Dengler 95] beschrieben.
Die prinzipiellen Voraussetzungen des Szenarios sind:

- alle Aktionen der Anwendungsdomäne sind axiomatisiert;
- es existiert eine vollständige Repräsentation des aktuellen Zustandes des Anwendungssystemes.

Betrachten wir zunächst zwei Aspekte, die einen benutzerorientierten Plan von einem rein zielorientierten Plan in der angenommenen interaktiven Software-Domäne unterscheiden.

Der Benutzer verwendet "Sensor"-Aktionen

Für das Planungssystem stellt sich die Problematik wie folgt dar. Eine Planspezifikation sei abstrakt durch folgende Formel beschrieben:

$$\forall x \;\; pre(x) \wedge \mathsf{P} \rightarrow \Diamond goal(x)$$

Der Beweis der Formel liefert als abstrakten Plan eine Instantiierung $\{plan(x)/\mathsf{P}\}$. Abstrakte Pläne dienen dazu, eine Klasse von konkreten Plänen kompakt zu beschreiben. Wenn der Planer einen konkreten Plan erzeugen würde, einen, der in der aktuellen Situation auch ausführbar wäre, würde er folgendermaßen vorgehen.
Es wäre notwendig, zwei Formeln zu beweisen, erstens eine abgewandelte Planspezifikation

$$\exists x \;\; pre(x) \wedge \mathsf{P} \rightarrow \Diamond goal(x)$$

und zum zweiten, daß der gefundene Plan in der aktuellen Situation anwendbar ist

$$current\_state \rightarrow pre(t)$$

für die Substitution $\sigma = \{t/x\}$ aus dem Beweis der Planspezifikation

Das Finden einer Substitution σ beim Beweis der Planspezifikation kann gerade so gesteuert werden, daß der Beweis der Zusicherung bzgl. der Vorbedingung des Planes gelingt. Der Beweiser hat dabei Zugriff auf eine *vollständige* Beschreibung der aktuellen Situation und kann den eingeschränkten Wertebereich der Variable x "berechnen".
Der Plan soll nun allerdings das Verhalten eines Benutzers simulieren, damit er als erfolgversprechende Hypothese für die Beobachtung des Benutzers dienen kann, d.h. die Wertebereichseinschränkung der Variable x muß auch vom Benutzer nachvollziehbar sein. Da der Benutzer im allgemeinen kein vollständiges Wissen über die aktuelle Situation hat, muß er sich das notwendige Wissen durch zusätzliche *informative* Sensor-Aktionen verfügbar machen. Diesem Vorgehen muß auch der generierte Plan entsprechen. Bei dem verfolgten Beweisansatz zur Generierung gelingt dies nur, wenn an entsprechender Stelle erweiterte Teilplanspezifikationen bewiesen werden. Die Art und Weise, wie Spezifikationen zu erweitern sind, wird durch domänenspezifisches Wissen in Form von planspezifischen Entscheidungsfunktionen (vgl. Abschnitt 3.1.4) bereitgestellt.
Betrachten wir ein konkreteres Beispiel aus der Kalendermanagementdomäne und gehen von der informellen Zielbeschreibung

> *"Der Termin mit Firma X soll aktualisiert werden und für den 13.12.96 wird ein Termin mit automatischer Erinnerungsfunktion eingefügt."*

aus. Aus Systemsicht ist der Kalendereintrag mit *Firma X* – da der aktuelle Systemzustand vollständig beschrieben ist – automatisch auffindbar, z.B. als Referenz *appointment*27. Der Benutzer hat im allgemeinen nicht die Möglichkeit, diese Information direkt verfügbar zu haben, sondern führt eine Suche oder Browsing-Aktion im Sinne eines *Sensoring* aus. Üblicherweise erhält er dann visuelle Information, die ihn in die Lage versetzt, auf den gesuchten Eintrag zuzugreifen bzw. er erhält diesen Eintrag unmittelbar. Festzustellen ist, daß hier die Sensor-Aktionen nur aus Benutzersicht notwendige Aktionen sind.

Der Benutzer verwendet bestimmte typische Verhaltensmuster

Bleiben wir bei dem beschriebenen Szenario und nehmen an, daß der Benutzer typischerweise einen neueingetragen Termin, für den er schon eine automatische Erinnerungsfunktion eingerichtet hat, zusätzlich (zur Sicherheit) auch noch ausdruckt. Dies ist bzgl. der unterstellten Zielbeschreibung ebenfalls keine notwendige Aktion und wird aus Systemsicht bei einem rein zielgerichteten Vorgehen nicht als Teil eines Planes erscheinen. Der Planungsprozeß kann aber auch hier durch die Integration einer planspezifischen Entscheidungsfunktion zur Berücksichtigung der zusätzlichen Aktion angesteuert werden.

Betrachten wir nun zunächst eine Planspezifikationsformel, die die oben genannte Zielbeschreibung als Planungsproblem umsetzt; sie lautet beispielsweise:

$$
\begin{aligned}
&[\ name(cal) = my\_cal \wedge opened(cal) = 1 \wedge \\
&\ has\_subject(cal, a) = \text{“}FirmaX'' \wedge has\_date(cal, a) = \text{“}9.1.97'' \wedge \\
&\ visible(cal, a) = 0 \wedge \\
&\ \ldots \wedge \Box[\neg \odot^{11} \bigcirc F]\ ;\ \bigcirc F \wedge \\
&\ \mathsf{Plan}\ ] \\
&\rightarrow \Diamond[updated(cal, a) = 1 \wedge \exists x\ [alarm\_set(cal, x) = 1 \wedge has\_date(cal, x) = \text{“}13.12.96'']]
\end{aligned}
$$

Eine Instantiierung der Metavariablen Plan durch eine Planformel, die neben den eben genannten benutzerspezifischen Aspekten auch noch weitere, plankonsumentenbezogene Merkmale aufweist, ist wie folgt gegeben:

$$
\begin{aligned}
&\Diamond[Ex(search\_appointment(cal, subject(\text{“}Firma\ X''))) \ ; \\
&\quad \Box deleted(cal, a) = 0\ ;\ \bigcirc updated(cal, a) = 1] \wedge \\
&\Diamond[\bigcirc has\_date(cal, b) = \text{“}13.12.96''\ ;\ \Box deleted(cal, b) = 0\ ; \\
&\quad Ex(set\_alarm(cal, b))\ ;\ Ex(print(cal, b))] \wedge \\
&\Box[opened(cal) = 1 \wedge \neg \odot^{11} \bigcirc F]\ ;\ \bigcirc F
\end{aligned}
$$

Eine Interpretation im Sinne einer Planbeschreibung lautet:

> Suche nach einem Termin mit Firma X, sorge dafür, daß er nicht gelöscht wird, so daß eine Aktualisierung des Termines erreicht werden kann. Sorge dafür, daß für den 13.12.96 ein Termin eingetragen ist und schütze ihn, so daß er, nachdem eine Erinnerungsfunktion für ihn eingetragen ist, ausgedruckt werden kann. Der Kalender soll während der gesammten Interaktion, die aus nicht mehr als 10 Einzelschritten bestehen darf, nicht geschlossen werden.

Diese Planformel charakterisiert Aktionsmodelle durch eine nebenläufige Verfeinerung bestehend aus drei Teilplänen:

1. *Die Aktualisierung des Termines mit Firma X*
   Die erste Aktion dieses Teilplanes, $search\_appointment(cal, subject(\text{“}Firma\ X''))$, ist eine “Sensor”-Aktion, die einen Termin mit Gegenstand *Firma X* markiert (auf

dem Bildschirm) darstellt. Durch die Ausführung der Aktion erreicht man, daß der Benutzer eine Referenz zu dem entsprechenden Termin aufbauen kann, d.h. mit den Eigenschaften $has\_subject(cal, a) =$ "*Firma X*" und insbesondere $visible(cal, a) = 1$ ist eine Instantiierung des Objektes a vom Benutzer *konzeptionell* vollzogen worden. Die Eigenschaft *visible* ist hierbei gerade dafür verantwortlich, daß die Sensor-Aktion eingeführt wurde. Sie wurde durch die Verwendung einer kontextabhängigen Entscheidungsfunktion als Zwischenziel in die Planspezifikation eingeführt.
Für den Termin a wird durch die Angabe eines Zieles spezifiziert, daß er aktualisiert ist, $updated(cal, a) = 1$. Da die Aktualisierung durch verschiedene Aktionen hervorgerufen werden kann, ist es für den Plankonsumenten *Planerkennung* effektiver, wenn das zu beobachtende Verhalten des Benutzers in dieser Hinsicht nur durch das zu erreichende Ziel beschrieben ist. Im Falle der Aktion *search_appointment* nehmen wir an, daß nur ein Aktionsschema verfügbar ist, das die Eigenschaft *visible* erreichen kann. Der Teilplan enthält eine Sicherheitsbedingung $deleted(cal, a) = 0$, die die Ausführbarkeit einer Aktualisierungsaktion sichert – der Termin a darf nicht gelöscht werden.

2. *Die Etablierung eines neuen, mit Alarmierung verbundenen Termines*
   Dieser Teilplan enthält ebenfalls eine Sicherheitsbedingung und konkrete Aktionen nur dort, wo sie einer injektiven Zuordnung zum gegebenen Ziel entsprechen. Die Etablierung[17] wird durch das Ziel $has\_date(cal, b) =$ "*13.12.96*" für ein bestimmtes Objekt b markiert. Die beiden konkreten Aktionen in der Folge *set_alarm* vor *print* entspringen der Integration eines wie oben skizzierten, benutzertypischen Verhaltensmusters. Die Annahme der Spezifizierung ihres unmittelbaren Hintereinanderauftretens verhindert[18] die Einführung einer eingeschobenen Sicherheitsbedingung. Die Aktion *set_alarm* alleine wäre nur notwendig, um das spezifizierte Ziel zu erreichen.

3. *Die Etablierung einer globalen Sicherheitsbedingung und Intervallbeschränkung*
   Die Bedingung $opened(cal) = 1$ ist eine strenge Sicherheitsbedingung – der entsprechende Kalender ist durchgehend aktiviert –, die für den gesamten Plan gilt; durch eine geeignete Plantransformation ist ihr redundantes Vorkommen auf diese Weise vereinfacht worden. Die zweite Bedingung, die eine Intervallängenbeschränkung ausdrückt, ist eine notwendige Sicherheitsbedingung bzgl. der Anforderungen des Plankonsumenten *Planerkennung*. Sie verlangt als Heuristik, das ein zu berücksichtigendes Benuzterverhalten aus nicht mehr als zehn Aktionen zusammengesetzt ist.

Die nebenläufige Verfeinerung, d.h. in diesem Fall die Generierung eines nichtlinearen Planes, entspringt primär der Anforderung des Plankonsumenten Planerkennung. Diese benötigt eine effektive Abdeckung möglichen Benutzerverhaltens zur Erreichung eines angenommenen Zieles; die betrachtete Domäne läßt eine große Variablität in der Reihenfolge der ausführbaren Aktionen zu.

Betrachten wir nun eine Adaptionspunktspezifikation für den gesamten Planer, die nach ihrer Realisierung zu einer Planungsstrategie führt, die eine geeignete Planspezifikation

[17] Z.B. durch Kopieren oder vollständigen Neueintrag.
[18] Besser, macht sie irrelevant.

zu einem wie oben beschriebenen Plan verfeinern kann. Gemäß den in Abschnitt 5.3 eingeführten Merkmalen und ihren Kombinationen kann die in Abbildung 5.7 dargestellte Spezifikation vorgenommen werden:

$$
\begin{aligned}
\{\\
&(AP_{split}, and(necessary\_safety,\ prefer\_to(\ and(concurrent, first\_goal\_single\_state)\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad sequential)))\\
&(AP_{foc}, and(regression, depth\_first))\\
&(AP_{imp},\ prefer\_to(\ empty\_plan,\\
&\qquad\qquad\qquad prefer\_to(action\_verification,\\
&\qquad\qquad\qquad\qquad\qquad prefer\_to(\ single\_action\_goal,\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad before(only\_goal\_plan,\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad abstract\_action)))))\\
&(AP_{probtrans},\ prefer\_to(\ prefer\_to(before(context\_dependency,\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad plan\_behavior\_goal\_extension),\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad plan\_behavior\_goal\_extension)\\
&\qquad\qquad\qquad\qquad no\_transformation))\\
&(AP_{act}, and(action\_index, regression\_filter))\\
&(AP_{plantrans}, before(weak\_to\_strong\_safety, explicit\_necessary\_safety))\\
\}
\end{aligned}
$$

Abbildung 5.7: Adaptionspunktspezifikation Kalenderdomäne

Domänenspezifische Information wird hier nur durch die Merkmale *context_dependency* und *plan_behavior_goal_extension* integriert. Sie sind mit dem Einsatz entsprechender Entscheidungsfunktionen gekoppelt. Beispielhafte Einträge der entsprechenden Funktionsgraphen sind gemäß der Umsetzung aus Abschnitt 5.1 beschrieben als:

- für das Merkmal *context_dependency*

$$
\begin{aligned}
\langle\ \Diamond updated(?x, ?y) = 1,\ visible(?x, ?y) = 0 : &\ \text{current\_precondition}\ \underline{oder}\\
visible(?x, ?y) = 0 : &\ \text{global\_precondition},\\
(\Diamond[visible(?x, ?y) = 1 \wedge \Diamond updated(?x, ?y) = 1])\ \rangle&
\end{aligned}
$$

- für das Merkmal *plan_behavior_goal_extension*

$$\begin{aligned}\langle\ \Diamond[&alarm\_set(?x,?y)=1 \wedge has\_date(?x,?y)=?v \wedge \mathsf{meta\_goal}],\\ &\forall z \exists d[has\_date(?x,z)=d \rightarrow \neg(d=?v)] : \mathsf{global\_precond}\ \underline{und}\\ &\neg\Diamond Ex(set\_alarm(?x,?y)) : \mathsf{current\_global\_plan},\\ &(\Diamond[\mathsf{meta\_goal} \wedge has\_date(?x,?y)=?v]\ ;\ [alarm\_set(?x,?y)=1 \wedge\\ &\qquad\qquad \bigcirc Ex(print(?x,?y))])\ \rangle\end{aligned}$$

Zu dem letzteren der beiden Beispieleinträge ist folgende Bemerkung anzufügen: Das Symbol meta_goal fungiert als Metavariable für eine modalfreie Formel und wird während der Ausführung des Anwendbarkeitstestes der Entscheidungsfunktion durch alle *restlichen* vorkommenden Ziele instantiiert.[19] Das durch die Entscheidungsfunktion getriggerte Verhalten beschreibt, daß unmittelbar nach Erreichung des Zieles $alarm\_set(\ldots) = 1$ die Aktion *print* ausgeführt wird. Das beschriebene Planverhalten enthält also sowohl eine Zielstrukturierung als auch eine konkrete Handlungsanweisung. Die explizite Einführung einer Aktion macht es notwendig, bei der Spezifikation des Adaptionspunktes AP_{imp} (siehe oben) das Merkmal *action_verification* zu berücksichtigen.
Durch das Merkmal *explicit_necessary_safety* erreicht man, daß die Sicherheitsbedingung über die maximal zulässige Länge des Planes aus der Planspezifikation auch in den Plan selbst übernommen wird. In dem betrachteten Beispielszenario sind die expliziten Bedingungen vom Plankonsumenten gewünschte Merkmale und gleichzeitig nutzbar für die eigene Suchkontrolle.

Bemerkungen zu unterbrechbaren Plänen

Die auf Seite 238 betrachtete Planformel als Ergebnis eines adaptierten Planungsprozesses enthält lokale und globale Sicherheitsbedingungen und repräsentiert in dieser Hinsicht einen eingeschränkt unterbrechbaren Plan. Aus der Sicht des Plankonsumenten *Planerkennung* bedeutet *unterbrechbar*, daß Beobachtungen von Benutzeraktionen, die nicht auf die primären Ziele oder Aktionen des Planes abbildbar sind, nicht notwendigerweise direkt zum Verwerfen der Planhypothese führen, sondern Aktionen sein können, die den Plan und das damit verfolgte Ziel nur kurzzeitig *unterbrechen*. Die Etablierung zeitlicher Abstraktionen zwischen planrelevanten Aktionen bringt neben dem Vorteil einer größeren Robustheit bzgl. planabweichender Aktionen auch einen Nachteil mit sich: die Zahl der gleichzeitig zu betrachtenden Hypothesen kann sich u.U. sehr stark erhöhen, da nun im allgemeinen für keine der beobachteten Aktionen mehr eindeutig entschieden werden kann, ob sie zu einem Planfortschritt oder -abbruch führt. Neben einer globalen Beschränkung der maximalen Zahl erlaubter Beobachtungen bzgl. einer Planhypothese gibt es auch noch die Möglichkeit, unter Berücksichtigung des Benutzerverhaltens lokal Unterbrechungen auszuschließen. Aus der früheren Beobachtung des Benutzerverhaltens (akkumuliert in einem Benutzermodell) kann sich beispielweise ergeben, daß bestimmte Aktionsfolgen von ihm praktisch nie unterbrochen werden. In dem obigen Planbeispiel ist dies für die Aktionsfolge *set_alarm* ... *print* angenommen worden. Um die Etablierung

[19] Sind außer den explizit genannten Zielen keine weiteren Ziele aktuell vorhanden, ergibt sich als Instantiierung die Formel T (true).

einer zeitlichen Abstraktion während der Planverfeinerung lokal zu verhindern, muß lediglich vermieden werden, daß die Basisvoraussetzung ihrer Einführung erfüllt wird. Dieses syntaktische Kriterium verlangt als Trigger eine zeitlich abstrahierte Zielspezifikation der Form

$$pre,\ \mathsf{P} \Rightarrow \Diamond\gamma$$

Die Verwendung der obigen Entscheidungsfunktion zur Berücksichtigung des Merkmales *plan_behavior_goal_extension* führt lokal zu einer zeitlich determinierten Zielspezifikation ($alarm\_set(\ldots) = 1 \boxed{\wedge \bigcirc} Ex(print(\ldots))$) und verhindert auf diese Weise die Etablierung einer Planunterbrechung.

Eine veränderte Domäne stellt veränderte Anforderungen an einen Planer

Als zweites Beispiel betrachten wir ein Szenario, in dem der primäre Plankonsument ein Planausführungs- und überwachungssystem in einer Roboterdomäne ist.[20] Es handele sich um eine räumlich abgegrenzte Situation, in der ein Roboter logistische Aufgaben zu erfüllen hat. Die Umgebung ist *dynamisch* in dem Sinne, daß neben dem modellierten Roboter auch noch andere *Agenten* eine Weltveränderung bewirken können. Hier werden diese Agenten in einem zu erstellenden Plan jedoch nicht explizit berücksichtigt.
Die Beschreibung eines Weltzustandes enthält zweierlei Eigenschaften. *Statische* Eigenschaften beschreiben als unveränderlich anzusehende Fakten der Domäne, wie z.B. räumliche Zusammenhänge. *Dynamische* Eigenschaften charakterisieren Fakten, die einer (möglichen) zeitlichen Veränderung unterliegen, hervorgerufen durch einen der Akteure in der Domäne, z.B. sind damit Beschreibungen beweglicher Objekte oder Maschinendaten erfaßt. Da hier nur Pläne für einen einzigen Agenten berücksichtigt werden, gibt es aus dessen Sicht unter den dynamischen Eigenschaften auch eine bestimmte Teilmenge, die für die Zeit der Planung als statisch anzusehen ist; es handelt sich dabei z.B. um Beschreibungen von Objekten, die zu verändern nicht im Autoritätsbereich des fokussierten Roboters liegt; man kann in diesem Sinne von *"Hindernissen"* sprechen.
Für die Erstellung und Form eines Planes werden folgende Anforderungen formuliert:

- *Hindernisse* werden als explizite Sicherheitsbedingungen aufgefaßt und steuern dabei während der Plangenerierung effektiv die Suche nach Lösungen, die die Hindernisse als solche frühzeitig erkennen und akzeptieren.
- Ein Plan beschreibt ein Verhalten mit zunehmendem Abstraktionsgrad, um die dynamische Veränderng der Welt (und der Hindernisse) frühzeitig in der Planausführungsphase zu berücksichtigen, d.h. der Plan enthält nur für wenige erste Schritte konkrete Aktionsanweisungen.

Eine Adaptionspunktspezifikation für einen auf dieses Szenario adaptierten Planer kann, wie in Abbildung 5.8 zu sehen, vorgenommen werden:
Die spezifizierten Merkmale beschreiben folgende Strategie zur Plangenerierung:

Es wird ein sequentieller (partiell konkreter) Plan durch ein progressives Vorgehen erzeugt. Teilziele, die während der Suche nach einer anwendbaren Aktion als (nicht notwendige)

[20] Dem Benutzer falle hier lediglich die Rolle einer übergeordneten Kontrolle im Sinne einer Leitstandüberwachung und eines Entscheiders in unvorhergesehenen Situationen zu.

$$
\begin{array}{l}
\{ \\
\quad (AP_{split},\ and(necessary\_safety, \\
\qquad\qquad and(and(sequential, optional\_subgoal), \\
\qquad\qquad\quad prefer\_to(complete\_goal\_single\_state, \\
\qquad\qquad\qquad first\_goal\_single\_state)))) \\
\quad (AP_{foc}, progression) \\
\quad (AP_{imp},\ prefer\_to(empty\_plan, \\
\qquad\qquad adapt\_to\_when(abstract\_actions, \\
\qquad\qquad\qquad only\_goal\_plan, \\
\qquad\qquad\qquad initial\_plan\_length(5)))) \\
\quad (AP_{probtrans}, before(\ prefer\_to(domain\_task\_reduction, \\
\qquad\qquad\qquad no\_transformation), \\
\qquad\qquad prefer\_to(static\_dynamic\_goal\_ordering, \\
\qquad\qquad\qquad prefer\_to(current\_context\_goal\_ordering, \\
\qquad\qquad\qquad\qquad no\_transformation)))) \\
\quad (AP_{act},\ and(progression\_filter, \\
\qquad\qquad prefer\_to(and(resolve\_underspecified\_parameter, \\
\qquad\qquad\qquad class\_common\_goals), \\
\qquad\qquad action\_index))) \\
\quad (AP_{plantrans}, nil) \\
\}
\end{array}
$$

Abbildung 5.8: Adaptionspunktspezifikation Roboterdomäne

Zwischenziele eingefügt werden, sind als optional spezifiziert (Merkmal: *optional_subgoal*). Damit kann man Zwischenzielannahmen, die sich beim progressiven Fortschreiten der Planverfeinerung und damit der Aktualisierung des Weltzustandes als "Umwege"[21] herausstellen als überflüssig betrachten und notwendige Ziele möglicherweise direkt erreichen.
Die Zielaufspaltung präferiert eine globale Strukturierung gegenüber einer lokalen (Merkmal: *complete_goal_single_state*). Dies harmoniert insbesondere mit dem spezifizierten Vorgehen, primär domänspezifisches Wissen zur hierarchischen Aufgabenstrukturierung (Merkmal: *domain_task_reduction*) zu verwenden. In der betrachteten Domäne entspricht dies der üblichen Verwendung von prinzipiellen Arbeitsvorgangsbeschreibungen.
Die Detailplanung von Aktionen erfolgt entsprechend der jeweiligen Situation unter Berücksichtigung der folgenden Teilzielstrukturierung: von den während der Lösungssuche entstehenden offenen Teilproblemen werden zunächst die, die statische Eigenschaften charakterisieren, verfeinert[22] (Merkmal: *static_dynamic_goal_ordering*). Sind diese verfeinert, so wird eine Zielordnung vorgenommen, die die Ziele präferiert, die in der aktuellen Situation (wahrscheinlich) am ehesten zu erreichen sind (Merkmal: *current_context_goal_ordering*). Zum Beispiel, befindet sich der Roboter in Raum A und sind offene Ziele bzgl. der Räume B und A zu erfüllen, so bedeutet dies, daß er das Ziel für Raum A als erstes erfüllen sollte.
Die Aktionsauswahl geschieht durch eine kombinierte Indizierung, zum einen zur Ausnutzung aller Effekte einer Aktion (Merkmal: *class_common_goals*) und zum anderen zur Präferierung von Aktionen, die möglichst wenig neue offene Teilziele erzeugen (Merkmal: *progression_filter*).
Die Teilzielrealisierung ist als ein adaptiver Prozeß beschrieben. Sobald durch das progressive Vorgehen eine initiale Planlänge von 5 Aktionen erreicht ist, wechselt die Planverfeinerung von der Einführung konkreter Aktionen (Merkmal: *abstract_actions*) hin zur Einführung von Zielen (Merkmal: *only_goal_plan*), die aufgrund der Spezifikation von $AP_{probtrans}$ Zielen der Aufgabenhierarchie (falls vorhanden) entsprechen. Die oben erwähnte primäre Verwendung einer Aufgabenhierarchie zur Zielaufspaltung unterstützt das progressive Vorgehen in der Hinsicht, daß möglichst schnell ein erfolgversprechendes initiales Planstück gefunden werden kann.
Die Berücksichtigung der konkret spezifizierten Sicherheitsbedingungen erfolgt durch das Merkmal *necessary_safety*. Bzgl. des Auftretens der Sicherheitsbedingungen als zusätzliches Constraint im Plan bestehen zwei Optionen:

- Sie erscheinen wie hier spezifiziert nicht im Plan
  Der Plankonsument ist dadurch nicht gezwungen, mögliche Veränderungen aller "Hindernisse" ständig zu kontrollieren, sondern erst im Fall, daß eine Aktion nicht ausführbar ist, zu reagieren.

- Sie werden als Teil des Planes aufgenommen[23]
  Hier wird der Plankonsument zu einer ständigen Beobachtung der "Hindernisse" gezwungen und kann entsprechend reagieren, bevor eine Aktion nicht mehr ausfürbar ist. Dieses strenge frühzeitige Verhalten ist meist jedoch nur bei sicherheitskritischen oder besonders wichtigen Eigenschaften notwendig.

[21] In der betrachteten Anwendung durchaus im räumlichen Sinne.
[22] Um dadurch u.a. geeignete Objektinstanzen zur Verfeinerung dynamischer Eigenschaften zu bestimmen.
[23] Bewirkt durch die Spezifikation $(AP_{plantrans}, explicit\_necessary\_safety)$.

Konkretisieren wir das betrachtete abstrakte Szenario beispielsweise zu einem Lebensmittellager, in dem mehrere Roboter logistische Aufgaben erfüllen. Abbildung 5.9 zeigt eine mögliche Skizze eines solchen Szenarios.

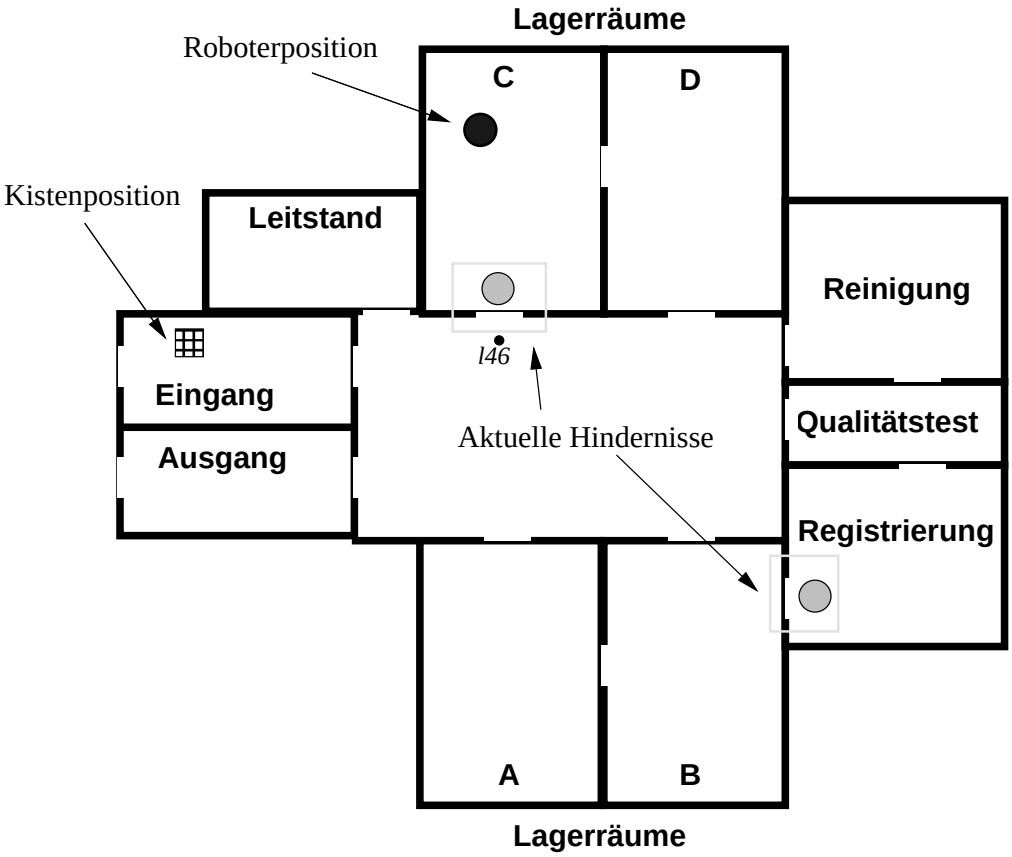


Abbildung 5.9: Skizze eines Roboterbeispielszenarios

Für die Aufgabe des Einlagerns einer Apfelkiste $kisteA$ ergibt sich z.B. folgende Planspezifikation:

$$location(kisteA) = l27 \wedge in\_area(kisteA) = Eingang \wedge$$
$$in\_area(robo) = C \wedge registered(kisteA) = 0 \wedge current\_job(robo) = kisteA \wedge \ldots \wedge$$
$$\Box[occupied(l51) = robo2 \wedge in\_area(robo2) = C \wedge$$
$$occupied(l34) = robo3 \wedge in\_area(robo3) = Registrierung] \wedge$$
$$\mathsf{Plan} \rightarrow$$
$$\Diamond stored\_in(kisteA) = B$$

Betrachten wir nun einige Aspekte der Lösung dieser Planungsaufgabe unter Berücksichtigung der obigen Adaptionspunktspezifikation. Es ergeben sich z.B. Einträge des Funktionsgraphen der mit dem Merkmal *domain_task_reduction* assoziierten Entscheidungsfunk-

tion von der folgenden Art:

$$\begin{aligned}
&\langle\ \Diamond stored_in(?x) = ?y,\\
&\ registered(?x) = 0 : \mathsf{global_precond},\\
&\ (\Diamond[cleaned(?x) = 1 \wedge analysed(?x) = 1 \wedge\\
&\qquad \Diamond[registered(?x) = 1 \wedge \Diamond[stored_in(?x) = ?y]]])\ \rangle
\end{aligned}$$

$$\begin{aligned}
&\langle\ \Diamond[cleaned(?x) = 1 \wedge \mathsf{meta_goal}],\\
&\ [in_area(?x) = Eingang \wedge current_job(?r) = ?x] : \mathsf{global_precond}\\
&\ (\Diamond[in_area(?r) = Eingang \wedge carries(?r) = ?x \wedge\\
&\qquad \Diamond[in_area(?r) = Reinigung \wedge cleaned(?x) = 1 \wedge \mathsf{meta_goal}]])\ \rangle
\end{aligned}$$

Als Plan entsteht durch angepaßte Planverfeinerung bei Definition entsprechender Aktionen die folgende Formel:

$$\begin{aligned}
&Ex(move(robo, door_position(C, D)))\ ;\\
&Ex(passdoor(robo, C, D))\ ;\\
&Ex(move(robo, door_position(D, Flur)))\ ;\\
&Ex(passdoor(robo, D, Flur))\ ;\\
&Ex(move(robo, door_position(Flur, Eingang)))\ ;\\
&\Diamond[in_area(robo) = Eingang \wedge carries(robo) = kisteA \wedge\\
&\qquad \Diamond[in_area(robo) = Reinigung \wedge cleaned(kisteA) = 1 \wedge\\
&\qquad\quad analysed(kisteA) = 1 \wedge\\
&\qquad\quad \Diamond[registered(kisteA) = 1 \wedge \Diamond[stored_in(kisteA) = B]]]]
\end{aligned}$$

Die Spezifikation von "Hindernissen" als Sicherheitsbedingungen ermöglichte auf folgende Weise eine effektive Berücksichtigung des Hindernisses in Raum C. Als Teilproblem, den Roboter aus Raum A in den Raum Eingang zu bewegen, entsteht unter Verwendung der modifizierten Zielaufspaltung gemäß des Merkmales *optional_subgoal* eine Spezifikation S:

$$\begin{aligned}
&in_area(robo) = C \wedge current_job(robo) = kisteA \wedge \ldots \wedge\\
&\Box[occupied(l51) = robo2 \wedge in_area(robo2) = C \wedge\\
&\quad occupied(l34) = robo3 \wedge in_area(robo3) = Registrierung] \wedge\\
&\mathsf{P} \Rightarrow\\
&\qquad \Diamond[\bigcirc F \wedge occupied(l46) = robo \wedge \mathsf{pre}_x],\\
&\qquad \bigcirc F \wedge in_area(robo) = Flur \wedge \mathsf{pre}_x
\end{aligned}$$

Die progressive Planungsstrategie entscheidet sich, zur weiteren Zielverfeinerung die Aktion $passdoor(robo, C, Flur)$ zu verwenden, weil durch sie nur *ein* neues atomares Teil-

problem, $occupied(l51) = robo$[24], entsteht. Bevor nun eine weitere Planverfeinerung vorgenommen wird, verlangen die spezifizierten Sicherheitsbedingungen bereits eine Überprüfung der Konsistenz des partiell beschriebenen Zwischenzielzustandes mit den in ihnen fixierten Randbedingungen. Es ergibt sich dabei unmittelbar ein Widerspruch aus den beiden Formeln $occupied(l51) = robo2$ und $occupied(l51) = robo$ wegen der globalen Voraussetzung $robo \neq robo2$. Damit scheitert die Zielverfeinerung frühzeitig und ermöglicht eine alternative Problemlösung. Die besteht im Beispielszenario nur aus der Möglichkeit, über den Raum D den Raum C zu verlassen. Hat der Roboter dann durch die progressive Anwendung der Aktion $passdoor(robo, D, Flur)$ den Raum Flur betreten, steht immer noch die obige Problemspezifikation S zur Lösung an. Das als *notwendig* spezifizierte Ziel $in\_area(robo) = Flur$ kann nun *opportunistisch* unmittelbar durch eine Planverfeinerung mit dem leeren Plan erreicht werden und somit ist die Erreichung des früher eingeführten Zwischenzieles $occupied(l46) = robo$ überflüssig geworden; die weitere Verfeinerung kann damit "ohne Umwege" fortschreiten.

Durch die syntaktische Struktur eines entstehenden Planes wird nun im Zusammenhang mit der Kopplung der Plangenerierung an eine Planausführungskomponente, die Methoden der Planinterpretation (vgl. Abschnitt 4.2.8) verwendet, ein inkrementeller Plangenerierungs- und ausführungsprozeß getriggert. Spätestens dann, wenn die Interpretation der Planformel ein abstraktes Ziel erreicht, beginnt eine erneute Generierungsphase, jetzt unter Berücksichtigung neuer Gegebenheiten. In [Dengler 96b] sind Planverfeinerungsprozesse im Zusammenhang mit einem Plangenerierungs- und ausführungsszenario für eine dynamische Roboterumgebung detailliert beschrieben (siehe auch die Erläuterungen in Abschnitt 5.5).

5.5 Beispiele im implementierten System

Es werden hier nun Ausschnitte aus dem implementierten Basissystem zur deduktiven Generierung von Plänen gezeigt. Umgesetzt sind hier die wesentlichen Prinzipien des Planens durch Modellverfeinerung, wie sie in Abschnitt 4.2.7 erläutert wurden. Die Steuerung des Planungsprozesses wird wie in Abschnitt 5.1 beschrieben durch *Taktiken* vorgenommen. Eine Implementierung des Taktikkonfigurationsmechanismus aus Abschnitt 3.4 liegt zur Zeit nicht vor.
Der implementierte Planer[25] ist Teil einer logikbasierten Entwicklungsumgebung für Planungssysteme, genannt RAP (vom engl. *Reasoning About Plans*, "Schlußfolgern über Pläne") [Bauer et al. 96]. RAP bietet zum einen Systemunterstützung bei der Modellierung von Planungswissen und zum anderen eine Palette effizienter, bereichsunabhängiger Inferenzsysteme für Pläne. Damit können Planungssysteme zur Lösung unterschiedlichster Aufgaben im Bereich interaktiver Softwaresysteme, logistischer Anwendungen oder spezieller sicherheitskritischer Anwendungen entwickelt werden, beispielsweise zur Planerstellung, Planerkennung, Planüberwachung, sicheren Planausführung und flexiblen Kombinationen aus diesen Prozessen (vgl. dazu auch [Feibel 97]). Abbildung 5.10 zeigt die prinzipielle Systemarchitektur von RAP. Ausgehend von einer konkreten Anwendungsdomäne wird ein Modell dieser Domäne in Form einer logischen Axiomatisierung mithilfe

[24] Eine Vorbedingung der Aktion.
[25] Detaillierte Beschreibungen von Teilkomponenten finden sich in [Geib 96, Fedeler 97].

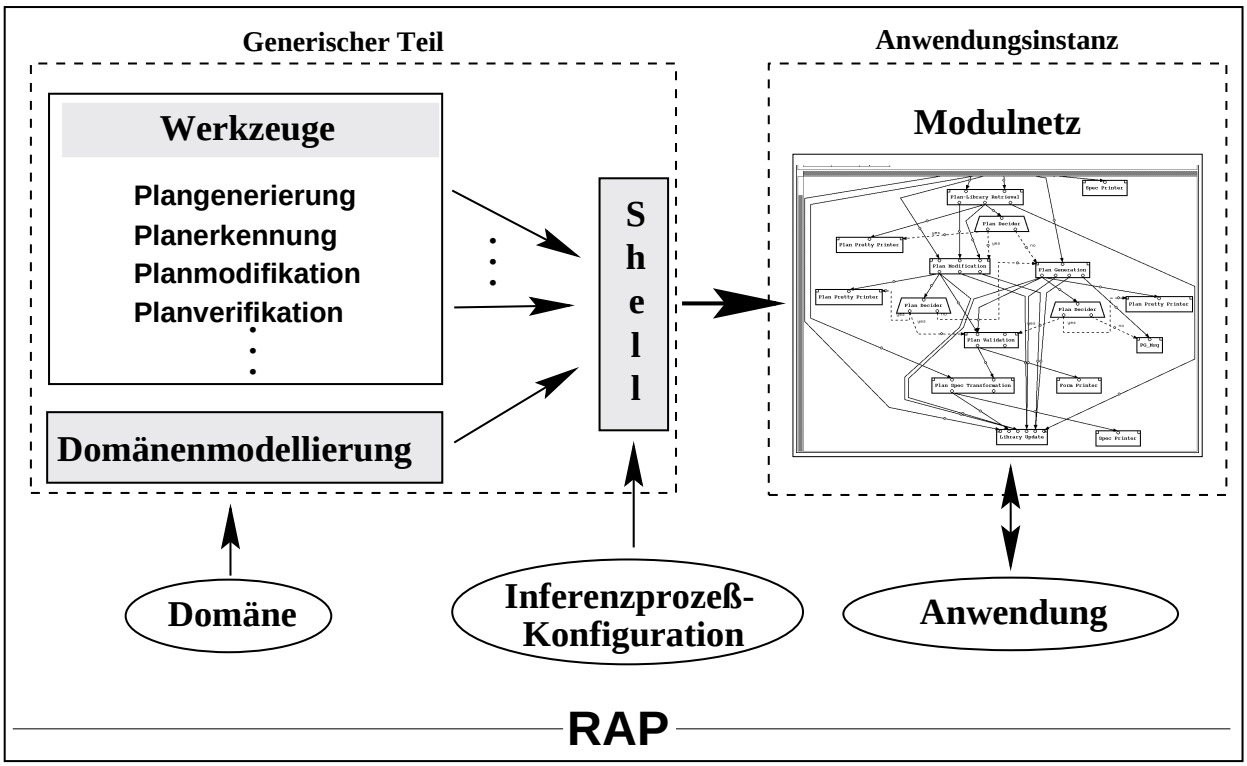


Abbildung 5.10: Architektur des generischen RAP-Systems

des *Domänenmodellierungswerkzeuges* definiert. Innerhalb des *Shell*-Systems erfolgt dann eine Konfiguration eines komplexen Planinferenzsystems aus den verfügbaren Werkzeugen zur Planerzeugung und -verarbeitung. Als Resultat dieses Prozesses entsteht ein sogenanntes *Modulnetz*, das die Ablaufsteuerung des Planinferenzsystems übernimmt.

Die Basis für den konkreten Planungsprozeß wird durch eine logische Beschreibung der jeweiligen Planungsdomäne gebildet. Anhand einer einfachen *Blockswelt*domäne werden zunächst die Prozesse der Domänendefinition demonstriert. Es existiert hierzu ein graphisches Werkzeug, das eine Definition auch ohne Kenntnis der konkreten Repräsentation für den späteren Planungsablauf erlaubt.

> Die betrachtete Blocksweltdomäne ist charakterisiert durch Objekte wie Blöcke und einen Tisch und durch zwei Aktionen `stack` und `unstack`. Sie beschreiben die Bewegung eines Blockes vom Tisch auf einen anderen Block, bzw. die Bewegung eines Blockes auf den Tisch.

Gemäß der in Abschnitt 4.2.1 beschriebenen objektzentrierten Repräsentation von Domäneneigenschaften spezifizieren wir einen strukturierten Typ `blocks_world`, so daß über Objekte diesen Typs jeweils alle Eigenschaften einer konkreten Blockswelt zugreifbar und veränderbar sind. Abbildung 5.11 zeigt die aktuelle Struktur der Sorte `blocks_world` innerhalb des graphischen Werkzeuges zur Domänenbeschreibung. Den Aufbau der dort verwendeten Sorte `block` zeigt Abbildung 5.12. Ein Block wird demnach durch seinen

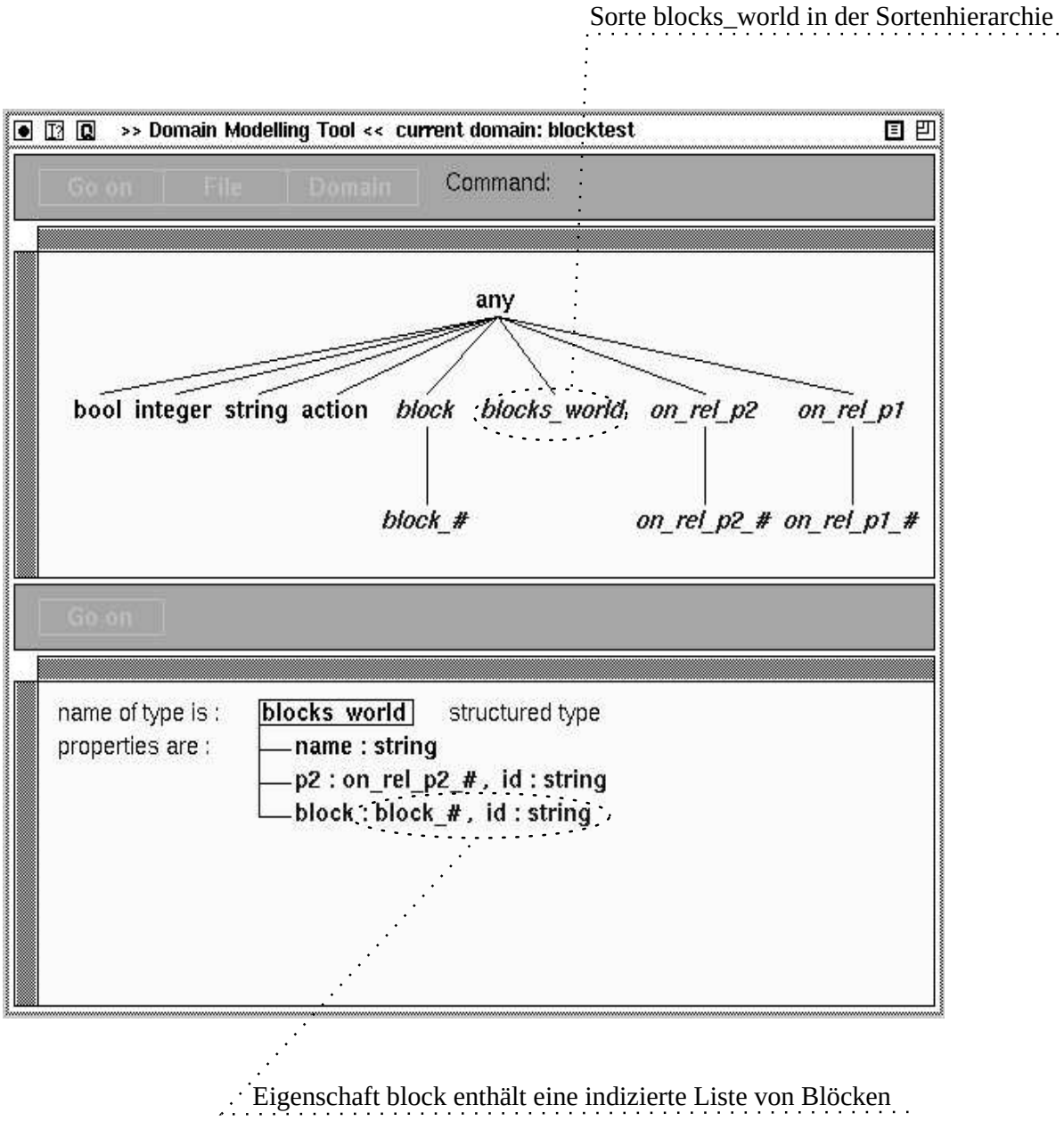


Abbildung 5.11: Strukturierter Typ `blocks_world`

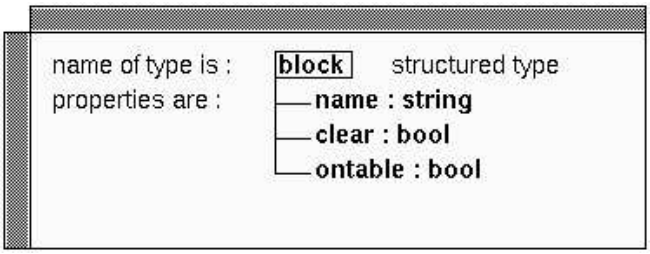


Abbildung 5.12: Strukturierter Typ `block`

Namen und die Eigenschaften, ob etwas auf ihm steht bzw. ob er auf dem Tisch steht, charakterisiert. Die Eigenschaft `p2` der Sorte `blocks_world` enthält eine doppelt geschachtelte indizierte Liste, mit deren Hilfe man eine zweistellige Relation `on`[26] repräsentieren kann.

Den nächsten Schritt einer Domänencharakterisierung bildet die Definition von Aktionen und ihrem Verhalten. Abbildung 5.13 zeigt den ersten Schritt der Spezifikation der Aktion `stack`. Man bestimmt zunächst abstrakt, welche Objekte bei der Definition der Aktion betroffen sind. Da die Sorte `blocks_world` eine komplexe Struktur darstellt, über die alle relevanten Eigenschaften der Domäne zugreifbar sind, ist zur Spezifikation sowohl der Vorbedingungen als auch der Effekte der Aktion jeweils nur ein Objekt der Sorte `blocks_world` notwendig. In dem folgenden Verfeinerungsschritt können dann die Veränderungen zwischen dem `blocks_world`-Objekt in den Vorbedingungen und dem in den Effekten der Aktion `stack` detailliert spezifiziert werden. Abbildung 5.14 zeigt eine resultierende Spezifikation. Um Beziehungen zwischen spezifischen Eigenschaften des `blocks_world`-Objektes aufzubauen oder bestimmte Eigenschaften mit Werten zu belegen,[27] kann man die strukturierten Objekte je nach Bedarf expandieren und mit den internen Relationen `equals`, `unequals` und `is_arg` Beziehungen aufbauen. Mit der obersten `equals`-Beziehung in Abbildung 5.14 spezifiziert man z.B., daß sich auf Vorbedingungs- und Effektseite das gleiche `blocks_world`-Objekt befindet. Die oberste `is_arg`-Beziehung legt den Identifikator des Blockes, der vom Tisch wegbewegt werden soll, als ersten Parameter der Aktion `stack` fest. Aus der graphisch erstellten Aktionsspezifikation wird automatisch die interne Repräsentation eines Aktionsschemas gemäß Abschnitt 4.2.5 erzeugt. Für die beiden Aktionen

[26] Zur Darstellung der Eigenschaft, daß ein Block auf einem bestimmten anderen Block steht.
[27] Boolsche Werte werden hier z.B. durch `0` (falsch) bzw. `1` (wahr) dargestellt.

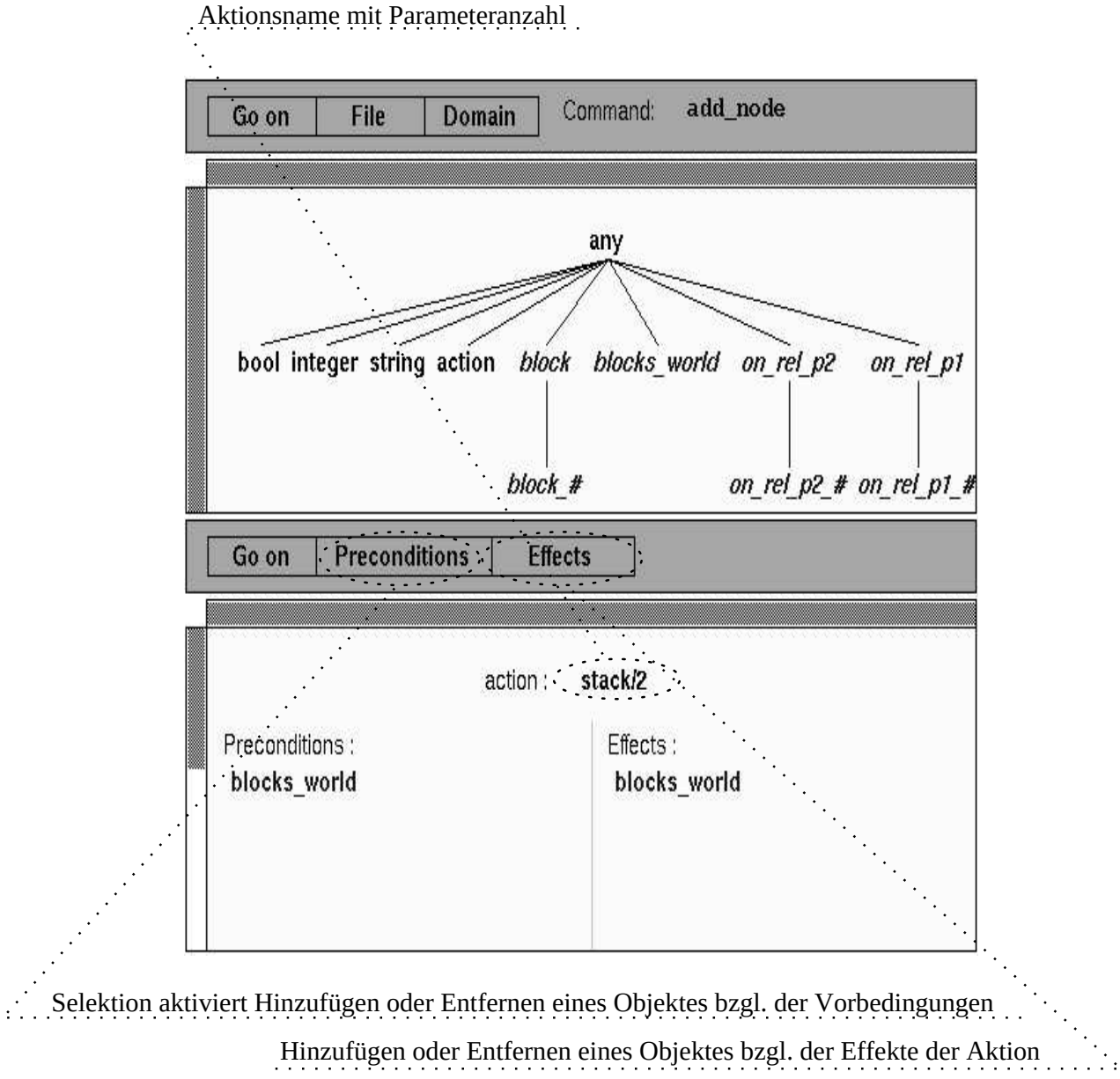


Abbildung 5.13: Abstrakte Aktionscharakterisierung

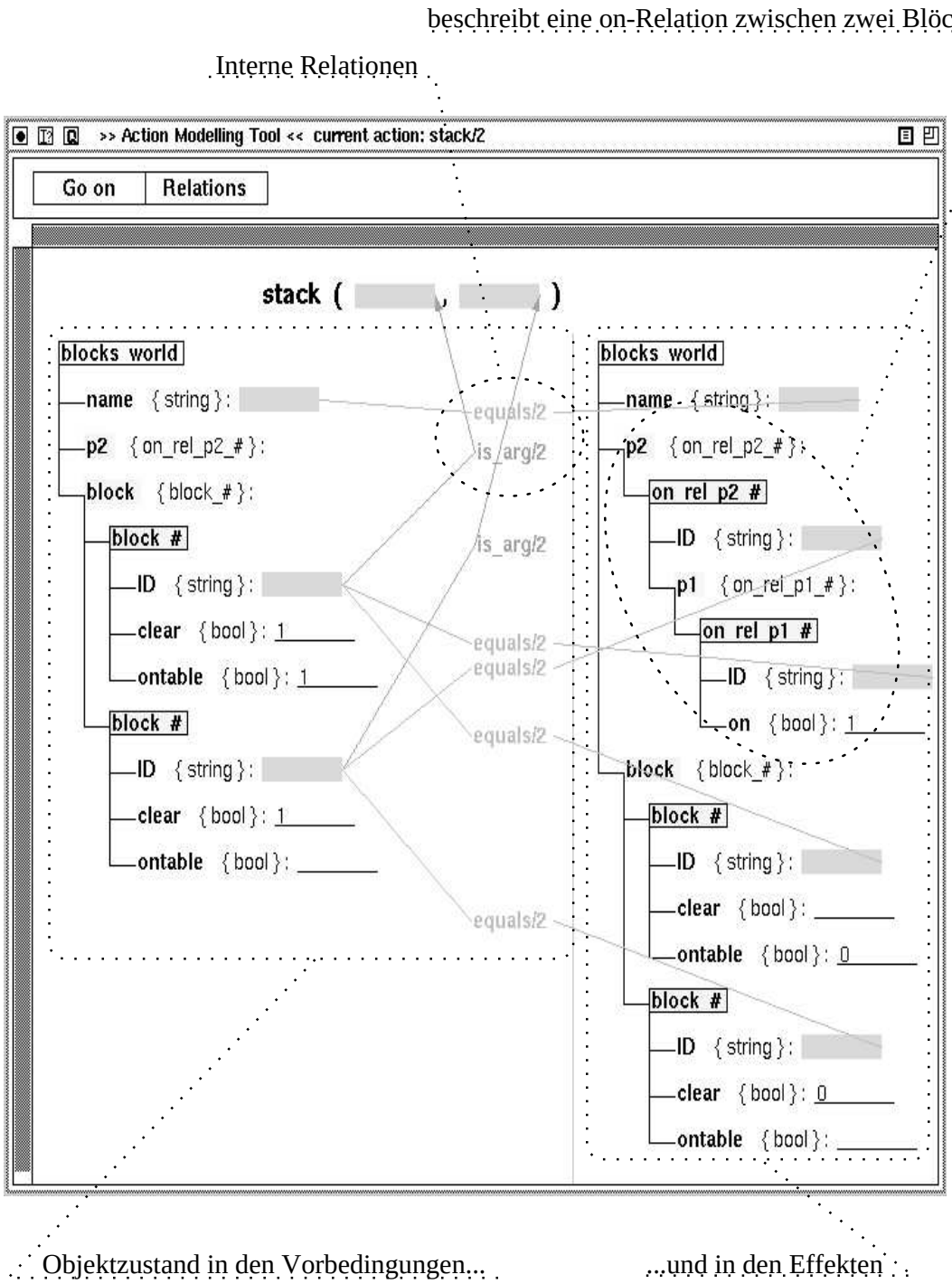
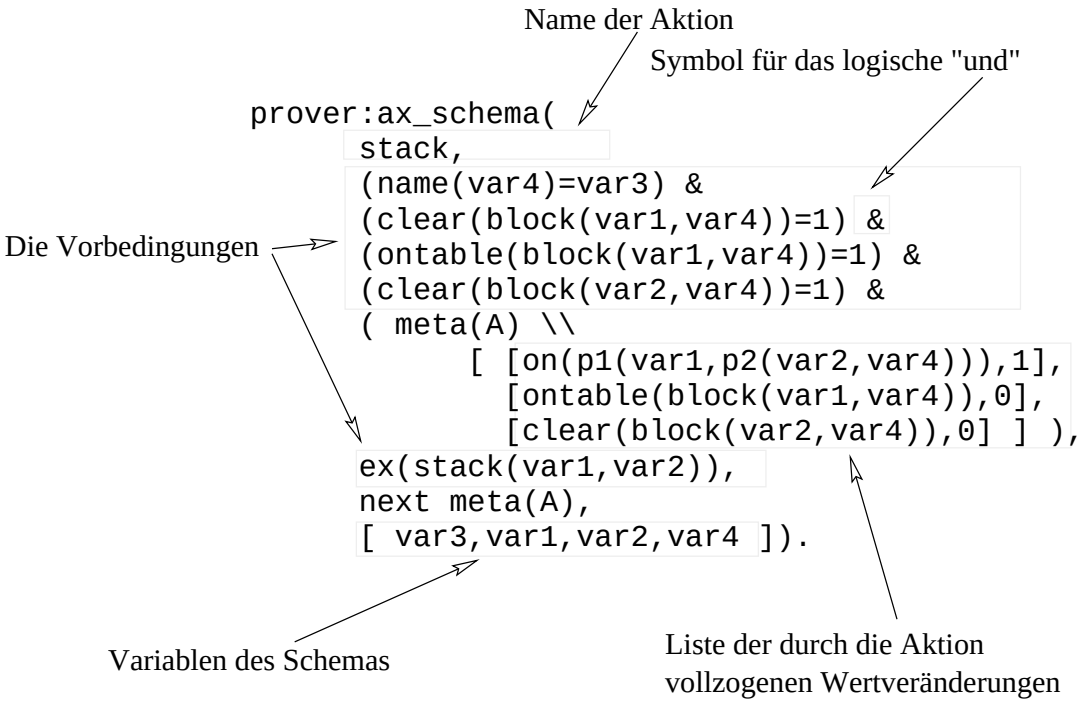


Abbildung 5.14: Detaillierte Aktionsspezifikation

`stack` bzw. `unstack` ergeben sich Aktionenschemata der Art:




```
prover:ax_schema(
      stack,
      (name(var4)=var3) &
      (clear(block(var1,var4))=1) &
      (ontable(block(var1,var4))=1) &
      (clear(block(var2,var4))=1) &
      ( meta(A) \\
            [ [on(p1(var1,p2(var2,var4))),1],
              [ontable(block(var1,var4)),0],
              [clear(block(var2,var4)),0] ] ),
      ex(stack(var1,var2)),
      next meta(A),
      [ var3,var1,var2,var4 ]).
```

und

```
prover:ax_schema(
      unstack,
      (name(var4)=var3) &
      (on(p1(var2,p2(var1,var4)))=1) &
      (clear(block(var2,var4))=1) &
      ( meta(A) \\
            [ [on(p1(var1,p2(var2,var4))),0],
              [ontable(block(var2,var4)),1],
              [clear(block(var1,var4)),1] ] ),
      ex(unstack(var1,var2)),
      next meta(A),
      [ var3,var1,var2,var4 ]).
```

Nach Definition der Aktionen zur Modellierung einer Planungsdomäne folgt als Vorbereitung der Durchführung eines Planungsprozesses die Definition von Planungsproblemen, in unserem Fall Planspezifikationen. Dies geschieht wiederum graphisch auf der Grundlage des Vorgehens der Aktionsdefinition. Zunächst legt man hier auf der abstrakten Ebene die temporale Struktur der zur Spezifikation des Planungsproblemes relevanten Zustände fest und bestimmt, welche Objekte in jedem einzelnen Zustand von Bedeutung sind. Abbildung 5.15 zeigt die Festlegung einer Problemstruktur bestehend aus dem Startzustand und einem Zielzustand. Abbildung 5.16 zeigt die darauf folgende detaillierte Definition einer Werteveränderungsbeziehung zwischen jeweils zwei Zuständen – in unserem Fall der zwei einzigsten Zustände. Modelliert wurde hierbei die durch die folgende Graphik beschriebene Situation:

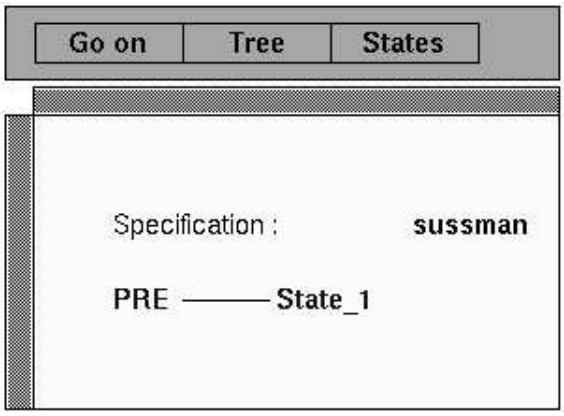
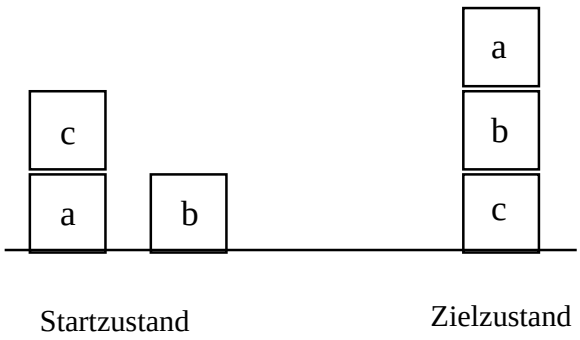


Abbildung 5.15: Abstrakte Planspezifikationsdefinition



Die charakterisierende Formel ist in ihrer internen Repräsentation gegeben durch einen PROLOG-Fakt

```
prover:plan_specification( sussman,
      [ local_var(var11,blocks_world), global_var(var10,string),
        constant(a,string), constant(c,string), constant(b,string),
        constant(1,bool), constant(0,bool) ],
      ( meta(_4747) &
        (name(var11)=var10) &
        (on(p1(c,p2(a,var11)))=1) & (clear(block(a,var11))=0) &
        (ontable(block(a,var11))=1) & (clear(block(b,var11))=1) &
        (ontable(block(b,var11))=1) & (clear(block(c,var11))=1) &
        (ontable(block(c,var11))=0) )
      ~~>
      ( sometimes( (on(p1(a,p2(b,var11)))=1) &
                   (on(p1(b,p2(c,var11)))=1) ) )).
```

Frei vorkommende globale Variablen, d.h. konventionelle logische Variablen, werden implizit als allquantifiziert angenommen. Tabelle 5.7 beschreibt zur Ergänzung die Umsetzung von logischen LLP-Operatoren in ihre systemintern verwendeten Pendants.

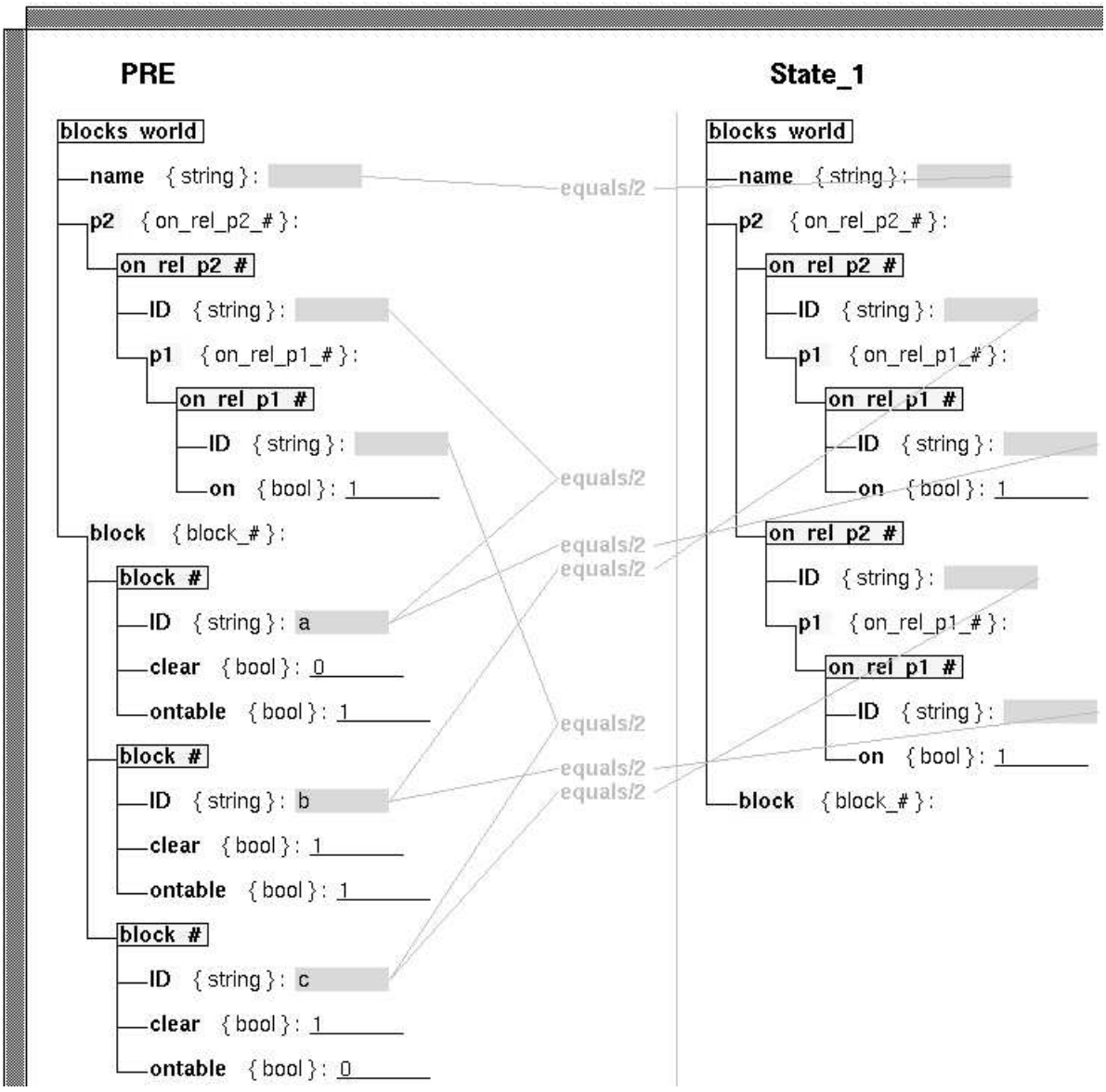


Abbildung 5.16: Detaillierte Modellierung einer Planspezifikation

LLP-Operator	**PROLOG-Repräsentation**
$\forall x\ p(x)$	`forall x :` p`(x)`
$\exists x\ p(x)$	`exists x :` p`(x)`
$p \rightarrow q$	p `~~>` q
$p \wedge q$	p `&` q
$p \vee q$	p `or` q
$p;q$	p `chop` q
$\Box p$	`always` p
$\Diamond p$	`sometimes` p
$\bigcirc p$	`next` p
$\odot p$	`strongnext` p
$\neg p$	`~` p

Tabelle 5.7: Übersetzung logischer Operatoren

Betrachten wir nun, mit welchen Werkzeugen Planungsprobleme gelöst werden. Abbildung 5.17 zeigt das Arbeiten mit der Benutzeroberfläche der deduktiven Planungsumgebung. Über die obere Menueleiste sind hierin verschiedene *Pull-down*-Menues aktivierbar, die Methoden zum Aufbau, der Verwaltung und Veränderung von Planungsdomänen,[28] Methoden zur Überwachung, Visualisierung und dem *Debugging* von Planungsstrategien und letztendlich auch verschiedene Basismethoden zur Plangenerierung bereitstellen. In Abbildung 5.17 ist folgende Situation beschrieben. Über den Menuepunkt `Load` wurde zunächst die Anwendung `blocks_world` – wie oben beschrieben – geladen. Dies bedeutet, daß gemäß einer entsprechenden Konfigurationsdatei verschiedene Schritte durchgeführt werden:

- Das Planungssystem, d.h. das taktikgesteuerte Beweissystem, wird um alle aktuell enthaltenen domänenspezifischen Eigenschaften, Indizierungen und Voreinstellungen bereinigt.
- Die Signatur der Planungslogikinstantiierung `blocks_world` wird dem Beweiskalkül verfügbar gemacht.
- Aktionsaxiomenschemata und Planspezifikationsformeln der neuen Domäne werden etabliert.
- Domänenspezifische Kontrollstrategien werden eingebunden, indem Standardstrategien/-taktiken durch erweiterte Strategien/Taktiken ersetzt werden; im Beispiel gehen wir nur von einer Standardstrategie aus.

[28] Neben den zuvor vorgestellten graphischen Werkzeugen existieren hier auch textbasierte Methoden.

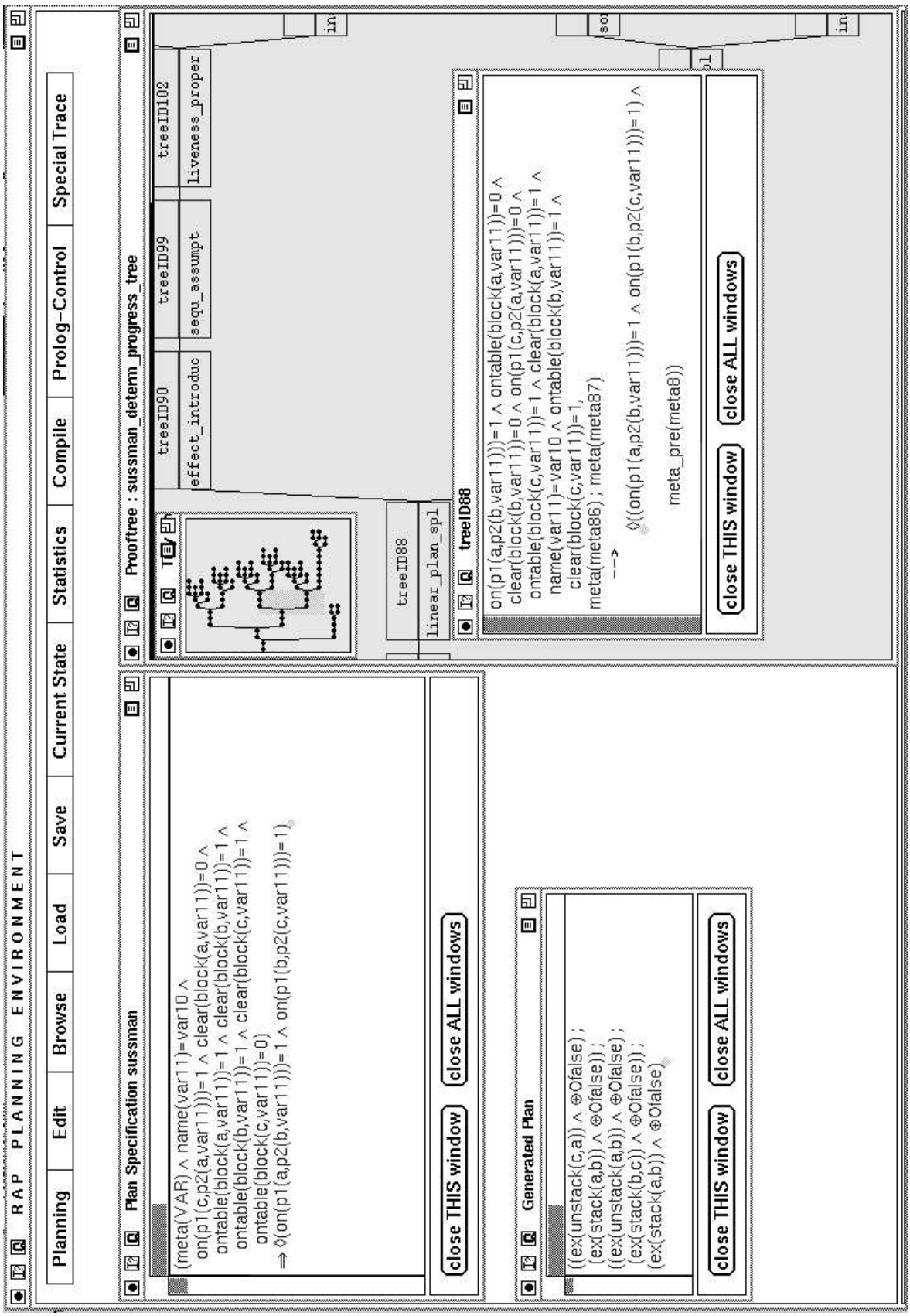
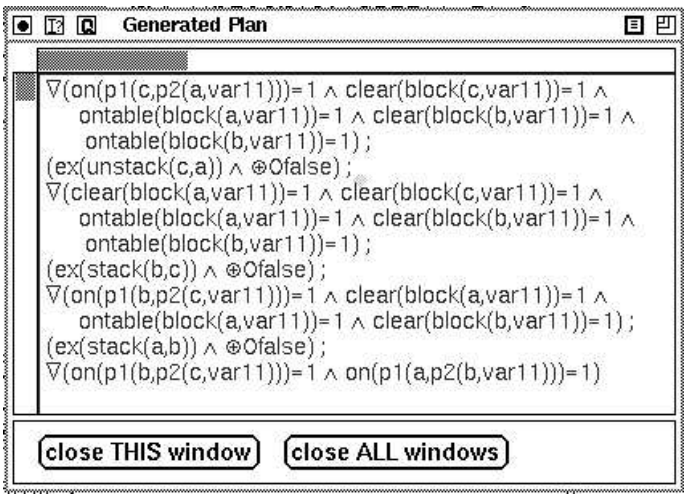


Abbildung 5.17: Oberfläche der Planungsumgebung

- Die Realisierung von Entscheidungsfunktionen (vgl. Abschnitt 3.1.4) wird hinzugefügt, durch die statische Einbindung bereits definierter Funktionsgraphen oder durch ihre automatische Erzeugung; wir gehen nur davon aus, daß die minimal notwendige Entscheidungsfunktion, die atomare Ziele mit den sie erreichenden Aktionen in Verbindung bringt, realisiert wird (wird vollautomatisch durchgeführt).

Nach dem Laden der Domäne, wird eine Planspezifikation als aktuelles Planungsproblem festgelegt. Das linke obere Teilfenster in Abbildung 5.17 zeigt eine Spezifikation "sussman", die der oben im Rahmen der Domänenmodellierung eingeführten Planspezifikation entspricht. Die Planungsstrategie, die zum Beweis dieser Spezifikationsformel benutzt wird, entspricht der in Abschnitt 4.2.7.1 eingeführten Strategie zur progressiven Erzeugung linearer Pläne. Als resultierende Planformel und damit Instantiierung der in der Spezifikation enthaltenen Planmetavariable ergibt sich die im linken unteren Teilfenster sichtbare Formel.[29] Führt man also Aktionen in der Reihenfolge `unstack(c,a)` gefolgt von `stack(a,b)`, `unstack(a,b)`, `stack(b,c)` und `stack(a,b)` aus, so erreicht man ausgehend von dem beschriebenen initialen Zustand einen Zustand, in dem die geforderte Zielspezifikation gilt. Das rechte Teilfenster in Abbildung 5.17 zeigt einen Ausschnitt aus dem Visualisierungstool für Beweisbäume. Die Verwendung eines Sequenzenkalküls als Basismaschine zur Beweisführung erzeugt Beweiszustände, die durch einen Baum charakterisiert sind. Die Knoten enthalten *Sequenzen*, d.h. Beweisprobleme; die Kanten sind markiert durch den Namen einer Regel, die auf die näher zur Wurzel liegende Sequenz angewandt wurde, so daß die über die Kante erreichbare Sequenz entstanden ist. Knoten im Baum können *inspiziert* werden, um die sie charakterisierenden Sequenzen detailliert zu betrachten. Im Beispiel ist ein Knoten gewählt, der eine Verfeinerung beschreibt, in der so eben eine partielle Planstrukturinstantiierung als Vorbereitung einer Aufsplittungsverfeinerung vorgenommen wurde.[30]

Verwendet man nun zum Beweis der gleichen Planspezifikation eine Strategie zur Erzeugung nichtlinearer Pläne mit strengen Sicherheitsbedingungen gemäß Abschnitt 4.2.7.5 so ergibt sich ein wie zu erwarten effizienterer Plan. In der Systemausgabe erscheint dieser in der Form[31]



[29] Das Symbol $\oplus$ wird zur Darstellung der Modalität $\odot$ verwendet.
[30] Wir befinden uns dabei in einem Zustand, in dem alle drei Blöcke auf dem Tisch stehen.
[31] Der Modaloperator $\Box$ wird hier aus technischen Gründen durch das Symbol ∇ dargestellt.

Abbildung 5.18 zeigt eine Visualisierung des nichtlinearen Planes, wie sie vom System zusätzlich zur Formeldarstellung bereitgestellt wird. Quadrate symbolisieren hier Sicher-

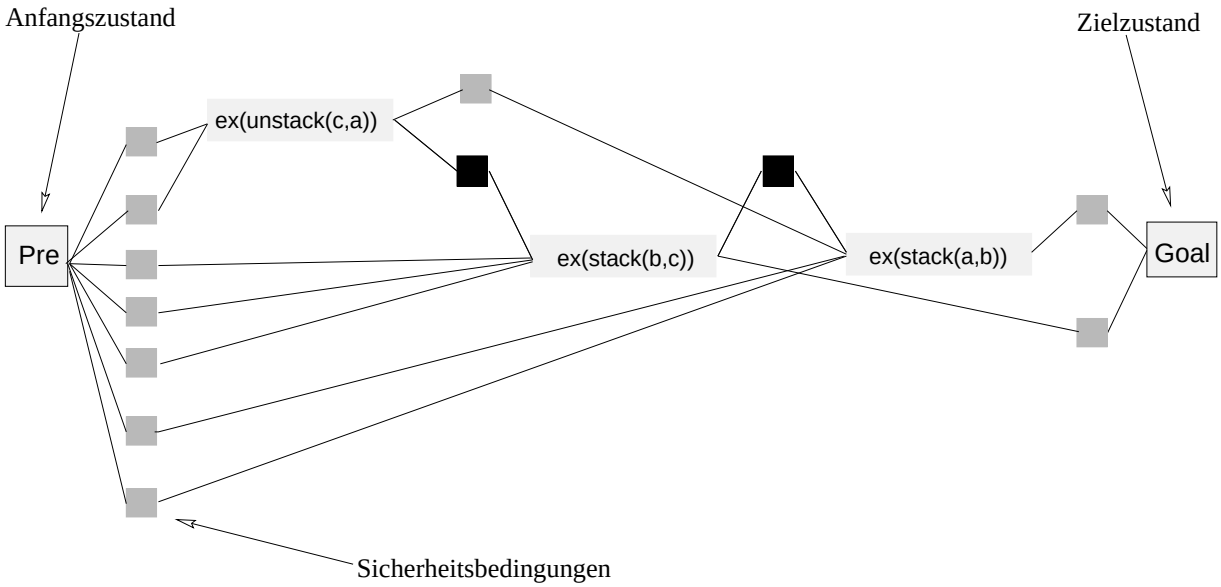


Abbildung 5.18: Visualisierung eines nichtlinearen Planes

heitsbedingungen, wobei ein Quadrat jeweils eine atomare Formel im Skopus eines $\Box$-Operators darstellt. Die schwarzen Quadrate repräsentieren Bedingungen der Form $\Box T$, die nur dazu dienen, eine notwendige Ordnung zwischen zwei Aktionen einzufügen, es besteht dann keine kausale Abhängigkeit im Sinne einer Produzent–Konsument-Beziehung zwischen den beteiligten Aktionen. Eine detaillierte Erläuterung der Implementierung der Strategie zur nichtlinearen Planung insbesondere unter Berücksichtigung ihrer Interaktionsfähigkeit bzgl. Zielfokussierung und Aktionsselektion findet sich in [Fedeler 97].

In der Planungsumgebung sind prototypisch neben der Blockswelt auch andere Domänen modelliert, z.B. *Electronic Mail*, einfache *Texteditor*vorgänge und eine Robotersimulationsumgebung. Ein Planungsbeispiel aus der *Mail*-Domäne ist in Abbildung 5.19 zu sehen.

Die zugrundeliegende Planspezifikation besagt, daß im initialen Zustand eine *Mailbox* mit Namen mymail existiert, die geöffnet ist: (open_flag(...)=1). Im Zielzustand soll eine Nachricht in dieser Mailbox gelesen und abgespeichert sein. Der Plan, der diese Spezifikation erfüllt, muß mit unvollständiger Information (vgl. Abschnitt 4.2.7.3) umgehen und beginnt aus diesem Grund mit einer Fallunterscheidung bzgl. der fehlenden Information über den aktuellen Verfügbarkeitsstatus der spezifizierten Nachricht. Wenn diese als gelöscht markiert ist, muß sie zunächst durch die Aktion undelete wieder reaktiviert werden. Ist sie nicht als gelöscht markiert, so kann sofort mit den Aktionen type (zum Lesen der Nachricht) und save (zum Abspeichern) fortgefahren werden.

Die oben bereits angesprochene Robotersimulationsumgebung ist aus verschiedenen Basismodulen konfiguriert als ein RAP-System zur sicheren Planausführung. Die Anwendung ist ein Lager für toxische Stoffe, die zur Wiederverwendung bzw. Entsorgung vorbereitet

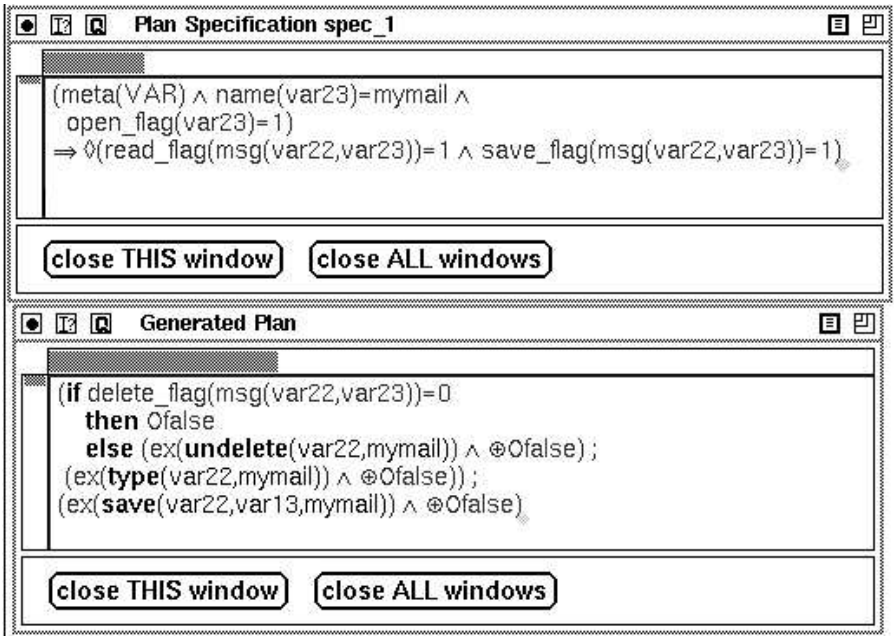


Abbildung 5.19: Beispiel aus der *Mail*-Domäne

werden. Die Ausführung von Handlungen in diesem Szenario unterliegt der Einhaltung strenger Sicherheitsvorschriften. In der Simulation wird die Aufgabe betrachtet, angelieferte Fässer von einer Analysestation in ein geeignetes Lager zu bringen, wobei bestimmte Sicherheitsbedingungen einzuhalten sind; Abbildung 5.20 zeigt einen Überblick der räumlichen Gegebenheiten des Szenarios in Form einer 3D-Graphiksimulation, die dem RAP-System als Visualisierungswerkzeug dient. Abbildung 5.21 zeigt einen Schnappschuß, der den Roboter nach Ausführung einer Aktion zeigt.
Ausgehend von einer formalen Problemspezifikation wird ein korrekter Plan zur Lösung der beschriebenen Aufgabe erzeugt. Um jedoch dynamischen Änderungen in der Umgebung während der Ausführungszeit Rechnung zu tragen, realisiert das in RAP konfigurierte Kontrollsystem eine sichere Planausführung, indem es vor jedem Planschritt dessen Durchführbarkeit und die aktuellen Sicherheitsbedingungen überprüft. Ist der Test negativ, so veranlaßt das System automatisch eine korrekte Neuplanung unter Berücksichtigung aller aktuellen Gegebenheiten. Abbildung 5.22 zeigt das graphisch konfigurierte Kontrollsystem als spezielles RAP-Modulnetz.
Die Anforderungen an die Plangenerierung sind hier vielschichtig und betreffen u.a.:

- den Umgang mit komplexeren Aktions- und Zustandsbeschreibungen,
- die Integration auch domänenspezifischer Heuristiken und
- die Generierung auch von partiell-abstrakten Aktions-/Zielplänen.

Zur Modellierung der Domäne wurden die folgenden Aktionen axiomatisiert:load_device, move, pickup_container, putdown_pallet, putdown_container, putdown_on_pallet, pass_door,

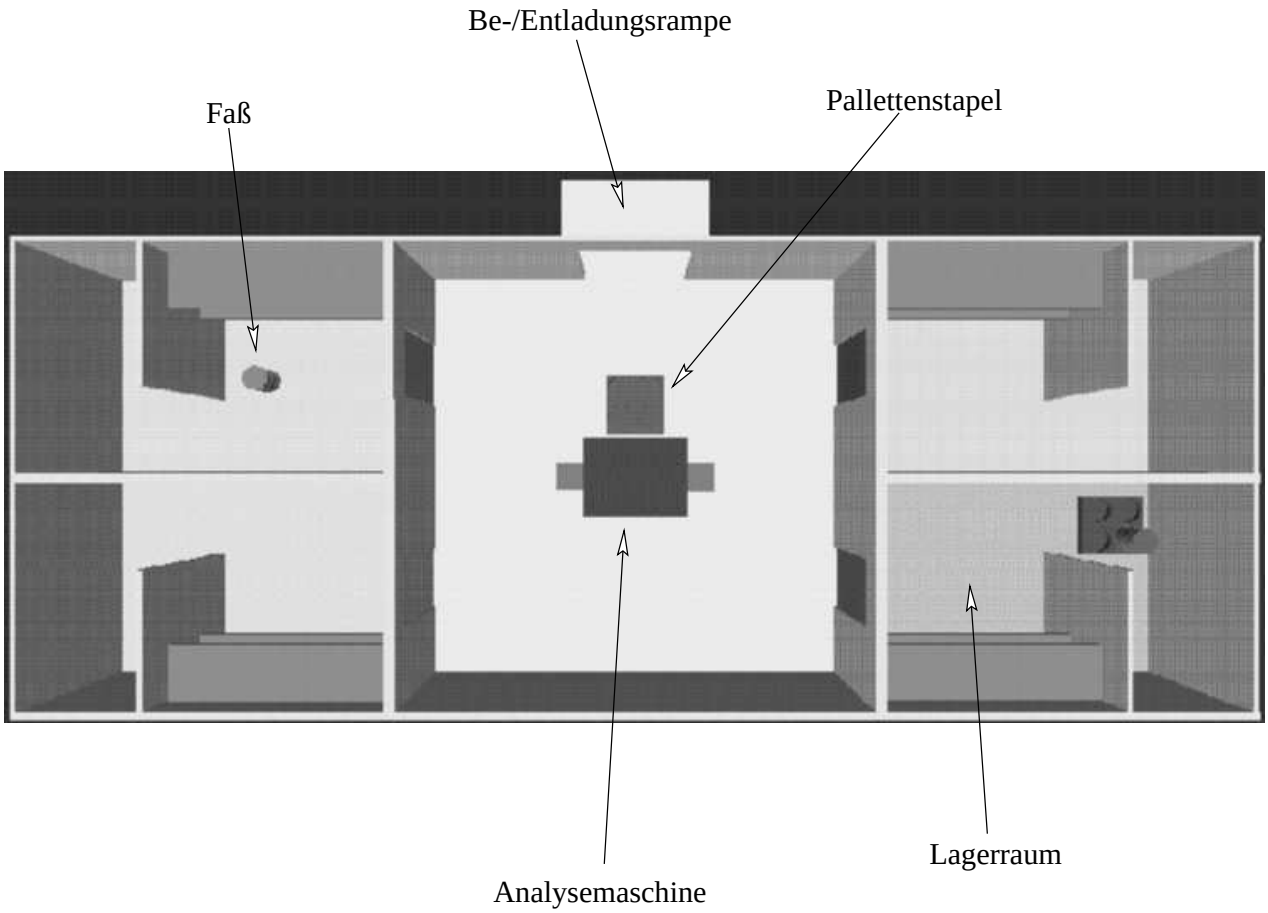


Abbildung 5.20: Ausschnitt einer graphischen Robotersimulation

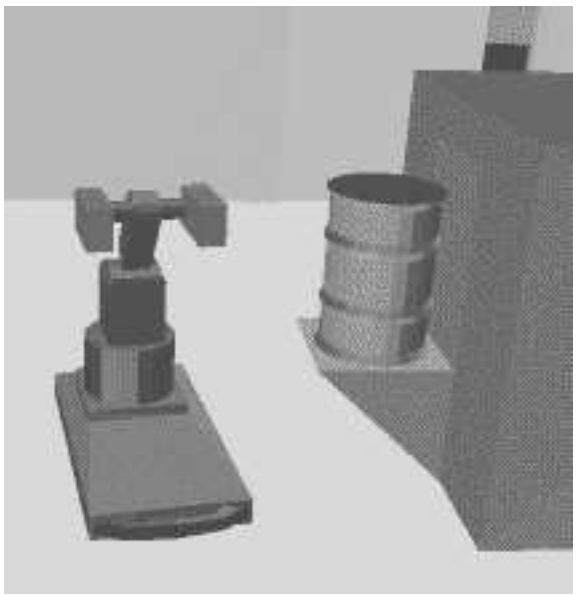


Abbildung 5.21: Roboter nach Ausführung von load_device

analyse, unload_device, new_pallet, pickup_pallet. Die Aktion load_device zur Beschreibung der Bestückung der Analysemaschine ist beispielsweise beschrieben in der Form

Referenz auf das Szenario

Eingangsposition der Analysemaschine

zugeordneter Roboterstandpunkt

Typ des Objektes

Maschine ist unbeladen

Position des Roboters

Objekt, das vom Roboter getragen wird

```
prover:ax_schema(load_device,
      (name(var7)=chew) &
      (inloc(device(var7))=var4) &
      (robo_loc(obj_loc(var4,var7))=var5) &
      (type(object(var1,var7))=barrel) &
      (loaded(device(var7))=0) &
      (location(robot(var7))=var5) &
      (carries(robot(var7))=var1) &
      ( meta(A) \\
            [ [free(robot(var7)),1],
              [carries(robot(var7)),nil],
              [occupied(location(var4,var7)),1],
              [is_on(location(var4,var7)),var1],
              [loaded(device(var7)),1],
              [loaded_obj(device(var7)),var1] ]
      ex(load_device(var1)),
      next meta(A), [ var7,chew,var5,var1,var4 ]).
```

Wie bereits diese Axiomatisierung deutlich macht, enthalten eine Zustandsbeschreibung

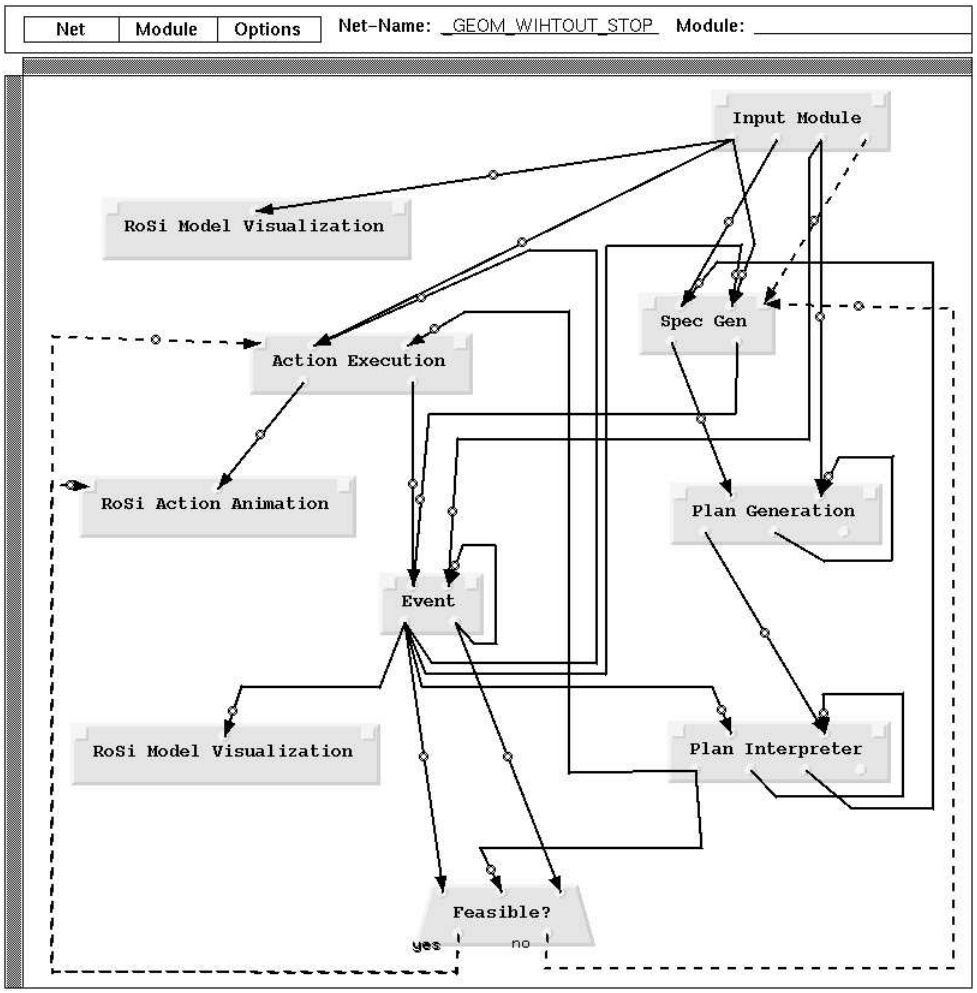


Abbildung 5.22: Modulnetz zur sicheren Planausführung

und damit auch die Vorbedingungen von Aktionen zwei unterschiedliche Arten von Bedingungen: *statische*, wie etwa der Typ eines Objektes oder Kodierungen der räumlichen Beziehungen zwischen Lokationen, und *dynamische*, wie z.B. aktuelle Eigenschaften, die den Roboter charakterisieren. Eine unterschiedliche Handhabung dieser Bedingungen während des Planungsvorganges ist von großer Bedeutung für die Effektivität der Planungsstrategie. Wenn es darum geht, Aktionen zu selektieren, um sie zur Erreichung eines bestimmten Teilzieles zu verwenden, so sind im allgemeinen nicht alle im Aktionenschema vorkommenden Variablen bereits instantiiert. Bezogen auf die eben dargestellte Aktion `load_device` und das angenommene Ziel `loaded(device(robosim))=1` bedeutet dies, daß für vier Variablen zunächst noch keine Instantiierung vorgegeben ist. Um mögliche Kandidaten für Instantiierungen in einer gegebenen Situation zu bestimmen, ist es effektiv, zunächst statische Bedingungen zu evaluieren, um die Geltungsbereiche für Variablen einzuschränken.
Eine typische Problemsituation ist nun wie folgt gegeben:

> *Der Roboter und ein nicht-analysiertes Faß stehen auf der Rampe. Ziel ist, daß das Faß in einen Lagerraum transportiert wird, der kompatibel ist mit dem analysierten Inhalt des Fasses.*

Entsprechend der axiomatisierten Domäne ergibt sich folgende (verkürzt wiedergegebene) Planspezifikation:

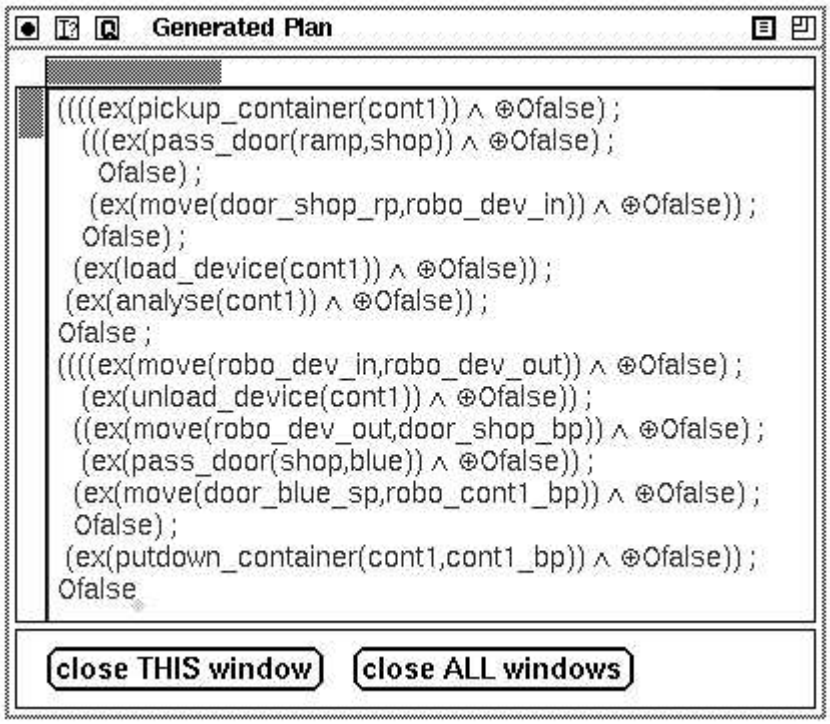
```
prover:plan_specification(robo_sim,
    [constant(dev_in,locID),constant(dev_out,locID),constant(cont1,objID),
     constant(ramp,areaID),local_var(robosim,chew), ... ],
( exists area : (exists l :
  ((meta(_) &
   (name(robosim)=scenario1) &
   (location(robot(robosim))=ramp_robo_pos) &
   (carries(robot(robosim))=nil) &
   (occupied(location(ramp_cont_pos,robosim))=1) &
   (occupied(location(ramp_robo_pos,robosim))=1) &

   ...   &  ...

   ~(area=ramp) & ~(area=shop) )
~~>
sometimes( (category(container(cont1,robosim))=area) &
           (area(location(l,robosim))=area) &
           (is_on(location(l,robosim))=cont1) )))).
```

Hierin sind die Symbole `area` und `l` existenzquantifizierte Variablen, für die während der Plangenerierung durch eine entsprechende Verfeinerung ein geeigneter Substitutionsterm gefunden werden muß. Ein linearer, dichter Plan, wie er z.B. durch die implementierte

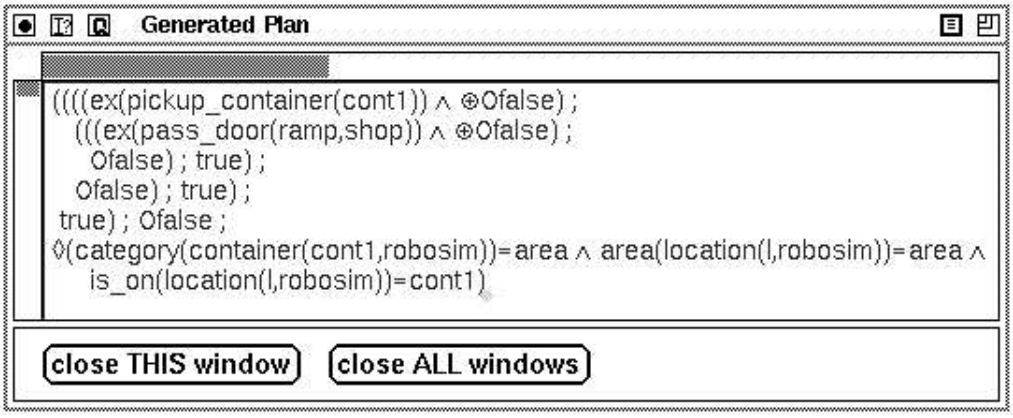
progressive Planungsstrategie generiert wird, ist original wie folgt dargestellt:



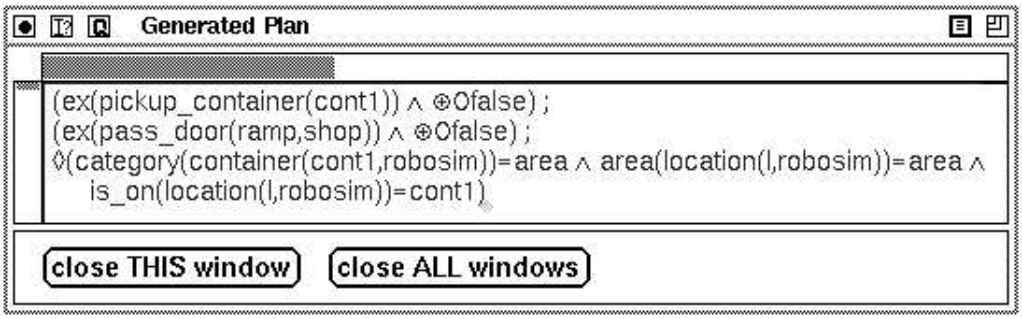
Es wird hier vorausgesetzt, daß die Aktion `analyse`, die den Inhalt eines Fasses innerhalb der Analysemaschine kategorisiert, den Inhalt des Fasses mit Namen `cont1` als für den blauen Raum bestimmt analysiert. Weiterhin werden Bewegungen des Roboters nur bzgl. signifikanter Positionen im Raum geplant. Welchen konkreten Weg der Roboter zwischen jeweils zwei Positionen genau wählt, wird nicht auf der deduktiven Ebene symbolisch entschieden, sondern obliegt in der jeweiligen Situation dem Roboter selbst, d.h. der Weg wird in unserem konkreten Fall durch geometrische Pfadsucheverfahren entschieden.
Die *leeren* Teilpläne in der Form ○false, die in dem obigen Plan an verschiedenen Stellen auftreten, sind entstanden, weil eine während der Lösungssuche eingeführte Aufsplittungsverfeinerung zur Zielvereinfachung sich als unnötig erwiesen hat; die Verfeinerung des ersten Teilintervalles hat bereits die Lösung für die Problemspezifikation des zweiten Teilintervalles als Seiteneffekt mitproduziert.

Das durch RAP konfigurierte Planausführungskontrollsystem ist in der Lage, mit Ereignissen in der Roboterumgebung, die nicht vom Roboter hervorgerufen wurden, umzugehen. Gemeint sind damit Zustandsveränderungen, die keinen kausalen Ursprung in den vom Roboter aktuell ausgeführten Aktionen haben. Damit ein vom Planausführungssystem abzuarbeitender Plan den dynamischen Gegebenheiten der Domäne besser angepaßt ist, ist es sinnvoll, ihn als kombinierten Aktions-/Zielplan zu erstellen, d.h. er stellt eine Verfeinerung der Art dar, daß einem dichten linearen Teilplan aus einer konkreten Aktionsfolge ein Teilplan in Form eines abstrahierten Zieles folgt. Die Motivation für die Bereitstellung eines solchen Planes ist, daß der Roboter, nachdem er einige Aktionen ausgeführt hat, unter der dann geltenden Zustandsbeschreibung bzgl. der verbleibenden Ziele erneut plant. Damit ist sichergestellt, daß zwischenzeitlich aufgetretene Ereignisse im weiteren Vorgehen des Roboters berücksichtigt werden können. Bis zu welchem Grad ein Plan nun

konkret verfeinert wird, ist domänenabhängig; gewählt wurde hier die Heuristik, nach Etablierung einer `pass_door`-Aktion, also dem Wechsel zwischen jeweils zwei Räumen, den Rest des Planes als abstraktes Ziel zu instantiieren. Verwendet wird hierzu als Basis eine progressive Strategie im Sinne der Beweisstrategie auf den Seiten 149ff. Sobald dann eine `pass_door`-Aktion etabliert wird, werden die bis zu diesem Zeitpunkt noch offenen Teilproblemspezifikationen durch Verfeinerungen gemäß der Strategie aus Abschnitt 4.2.7.7 gelöst. Die durch die Basisstrategie eingeführte syntaktische Unterscheidbarkeit zwischen notwendigen und Hilfszielspezifikationen wird genutzt, um Teilpläne bzgl. der Hilfsziele durch Formeln T ($true$) zu instantiieren; nur bzgl. der notwendigen Ziele erfolgt eine Instantiierung durch konkrete Ziele. Für das obige Beispiel einer Planspezifikation und den zugehörigen vollständigen Plan ergibt sich mit der angepaßten Strategie dann zunächst ein inkrementeller Plan der Art



der dann als erste Eingabe für das Planausführungssystem dient; allerdings in der logisch äquivalenten vereinfachten Form



Kapitel 6

Abschließende Bemerkungen

6.1 Zusammenfassung der Arbeit

Die vorliegende Arbeit untersucht die logische Formalisierung und Anwendung eines generischen, logischen Planungsansatzes im Kontext intelligenter Assistenzsysteme. Der Ansatz ermöglicht, unterschiedliche, konsumentengerechte Planerstellungs- und verarbeitungsmethoden zu spezifizieren und zu operationalisieren, die die vielfältigen, notwendigen Planschlußfolgerungsmechanismen für Aktionsplanungsaufgaben in Assistenzsystemen entweder definieren oder zumindest unterstützen.
Um dies zu erreichen, wird für eine temporale Planungslogik ein allgemeines kompositionales *Verfeinerungs*-Verfahren eingeführt, das es erlaubt, einen deduktiven, konstruktiven Beweis nach planungsspezifischen Gesichtspunkten zu *komponieren*. Das Verfahren erlaubt die freie Kombinierbarkeit von *progressivem* und *regressivem* (jeweils inkrementellem) Vorgehen, sowohl aus der Perspektive von *Zuständen* und *partiellen Plänen* als auch aus einer *aufgabenorientierten* Betrachtungsweise heraus. Pläne mit temporalen und konditionalen Kontrollstrukturen entlang eines großen Abstraktionsspektrums können auf diese Weise erzeugt werden. Grundlage ist ein verallgemeinerter Planbegriff, der die aus Adaptivitätssicht nötige, stufenlose Flexibilität in der Planrepräsentation ermöglicht. Aktionen sind dabei nicht mehr *notwendige* Basisbestandteile von Plänen, sondern stellen nur noch eine unter vielen Möglichkeiten dar, Aktionsverhalten zu charakterisieren; als allgemeines Schema zur Charakterisierung von Plänen wird *das zeitliche Verhalten von Eigenschaften und Ereignissen*, das auf bestimmte Weise das Erreichen spezifizierter Ziele beschreibt, benutzt.
Zur Erreichung der geforderten Adaptivität wird der logische Ansatz spezialisiert durch die Verwendung *konfigurierbarer Beweisstrategien*, die aus der Sicht des Planens eine direkte Abbildung von Planungsstrategien sind. Ausgehend von dem Metasprachenkonzept zur Beweisprogrammierung wird eine Metasprache zur Berücksichtigung planungsspezifischer Aspekte definiert. Sie sieht Konzepte zur Konfigurierung von Strategien, in diesem Kontext auch *Taktiken* genannt, vor und bietet Konstrukte zur Integration von *Entscheidungsfunktionen* an. Diese Funktionen repräsentieren in allgemeiner Weise planungsrelevante Präferenzen oder Heuristiken, beispielsweise zur Aktionsauswahl und Teilzielordnung.
Die technische und praktische Umsetzung des Metasprachenkonzeptes erfolgt mit den Methoden metalogischer Programmierung, da hier sowohl die Spezifikation eines Beweis-

kalküls als auch die geforderte Dynamik und Adaptivität nahezu optimal unterstützt werden.
Gemäß dem verfolgten flexiblen Verfeinerungsverfahren zur Planerstellung werden *Beweisbausteine* in Form von Taktiken für die verschiedenen planungsspezifischen Phasen mit Berücksichtigung der Schnittstelle "Planspezifikation" spezifiziert. Ein konfigurierbarer Beweisplan mit einer definierbaren Zahl von *Adaptionspunkten* wird als Taktikskelett spezifiziert. Als Spezifikationssprache zur Variantenspezifikation des generischen Planers wird eine deklarative *Merkmalsalgebra* verwendet. Die geeignete Übersetzung von Termen dieser Algebra in Taktiken und deren Integration in ein Taktikskelett definiert dann den Prozeß der Beweisstrategiekonfiguration.
In Beziehung zu bisher existierenden Planungsansätzen ergeben sich folgende Vorteile:

- Die gesamte Verarbeitungskette der Planerstellung, von der Definition und Repräsentation der Anwendungsdomäne und der gewünschten Lösungen, der Spezifikation von Kontrollstrategien, ihrer Anpassung auf eine konkrete Domäne, bis hin zur Ausführung der Generierung und Lösungsextraktion, wird als ein wohldefinierter, zusammenhängender Prozeß mit präziser Semantik beschrieben.

- Es wird ein verallgemeinerter Planbegriff zugrundegelegt, der neben Aktionen auch intensionale Beschreibungen vorsieht, um intendiertes Aktionsverhalten zu charakterisieren.

- Das formale Modell der adaptierbaren Kontrollstrategien ist unabhängig von dem konkreten logischen Formalismus zur Planrepräsentation und -erstellung und erlaubt eine flexible Festsetzung von Adaptionspunkten.

- Die formale Methode zur Spezifikation und Operationalisierung des Planungsprozesses unterstützt die allgemeine Verifikation dieses Prozesses.

- In der konkretisierten Planungslogik können Pläne durch eine Vielzahl unterschiedlicher Kontrollstrukturen und Abstraktionsgrade beschrieben werden.

- Es besteht eine hohe Variationsmöglichkeit bei der Beschreibung von Planungsproblemen und alle erstellten Lösungen sind beweisbar korrekt.

- Die semantische Nähe von Planrepräsentation und Plangenerierung erlaubt eine unmittelbar verifizierbare Anpassung und Veränderung beider Prozesse.

- Die mit der Plangenerierung nahe verwandten Prozesse der Planinterpretation, Planverifikation und Planvalidierung erhalten in dem logischen Rahmenwerk eine natürliche Abbildung und sind mit nahezu den selben Strategien und Inferenzregeln wie die Plangenerierung durchführbar.

Im Rahmen dieser Arbeit wurden die folgenden wissenschaftlichen Lösungen erarbeitet:

1. Aufgrund der vielfältigen Anforderungen an die Qualität von Plänen und die Vorgehensweisen ihrer Erzeugung und Verarbeitung wurde ein formales Modell konfigurierbarer, dynamischer Planungsstrategien entworfen. Das Konzept des taktischen Beweisens wurde dazu nach planungsspezifischen Gesichtspunkten angepaßt und

auf die Konfigurierung von Taktiken/Strategien erweitert. Die Operationalisierung adaptierbarer Strategien erfolgt durch ihre Implementierung in Form ausführbarer, algebraischer Spezifikationen, für die ein dynamisches Ausführungsmodell in der logischen Programmiersprache PROLOG angegeben wird; dieses Modell erlaubt unter gewissen Einschränkungen die selbstadaptive Veränderung einer Planungsstrategie während ihrer Ausführung. Die Schnittstelle zur aktuellen Konfiguration einer Strategie ist durch eine Merkmalsalgebra definiert, für die Übersetzungsmethoden zu Taktikspezifikationen definiert wurden.

2. Als Konkretisierung des formalen Modelles wurde ein deduktiver Planungsansatz in einer modalen, temporalen Planungslogik spezifiziert und implementiert. Dieser integriert die klassischen Verfahrensweisen des zustandsbasierten, planbasierten und aufgabenorientierten Verfeinerns und macht sie flexibel kombinierbar. Es wurde ein verallgemeinerter Planbegriff basierend auf Aktionsmodellen eingeführt. Dieses Konzept macht sich zunutze, daß Pläne in der gewählten logischen Sprache als Formeln repräsentiert sind und dadurch als Informationsstrukturen sowohl intensionale als auch extensionale Beschreibungen von Aktionsintervallen beinhalten können.

3. Unterschiedliche Planungsstrategien zur Erzeugung einer Vielzahl von Planabstraktionen und Kontrollstrategien wurden spezifiziert und jeweils hinsichtlich progressiver und regressiver Vorgehensweise untersucht, um die freie Kombinierbarkeit verschiedener Verfeinerungsstrategien zu gewährleisten.

4. Für typische Adaptionspunkte, die den Plangenerierungsprozeß detailliert in 6 korrelierte Phasen untergliedern, wurden ihre Beziehungen zum Verfeinerungsverfahren analysiert und konkrete Adaptionsmerkmale und -kombinationen basierend auf den eingeführten Planungsstrategien spezifiziert. Eine Konkretisierung des formalen Modelles zur Kontrollstrategiekonfiguration wurde exemplarisch durchgeführt.

6.2 Offene Fragen und weiterführende Forschungen

In der vorliegenden Arbeit wurde erstmals ein umfassendes, formallogisches Modell eines adaptierbaren deduktiven Planungsansatzes erarbeitet. Die vorgestellten Lösungen bieten einen geeigneten Ausgangspunkt, um weitere Untersuchungen durchzuführen, die durch folgende Fragestellungen spezifiziert werden:

- *Kann der adaptierbare Planungsansatz homogen hin zu komplexeren Planausführungsmodellen erweitert werden?*
  Untersuchungen in dieser Arbeit stützen sich auf die Verwendung eines sequentiellen Ausführungsmodelles. Eine natürliche Erweiterung bildet das Erlauben von echter Parallelität auf Aktionsebene. Zu untersuchen wäre, wie weit die Verwendung des dann veränderten Planbegriffes Veränderungen am Planungsmodell mitsichbringt: eine Erweiterung der Planungslogik und die Anpassung von Aktionsetablierungsmechanismen ist offensichtlich. Durch die erlaubte Parallelität ergeben sich möglicherweise auch neue Adaptionspunkte im Planungsverfahren, sicherlich jedoch neue Adaptionsmerkmale. Unklar ist ebenfalls, welche zusätzlichen Intervallverfeinerungsarten und Kontrollstrategieeinflußmöglichkeiten benötigt werden.

- *Wie kann die Akquisition von Wissen zur Konkretisierung von Entscheidungsfunktionen automatisiert werden?*
  Bezüglich der Definition von Entscheidungsfunktionen wurde bisher angenommen, daß z.B. benutzerspezifische Daten wie allgemeine oder situationsbedingte Aktionspräferenzen bereits in Entscheidungsfunktionsformat umgewandelt vorliegen. Da diese dynamischen Daten in einem intelligenten Assistenzsystem von einer Benutzermodellierungskomponente und/oder durch statistische Methoden gewonnen werden, müssen sie auf geeignete Weise konzentriert und umgewandelt werden. Der Einsatz von Lernverfahren zur Abstraktion von typischem Planverhalten eines Benutzers, das dann in Form relevanter Muster in Entscheidungsfunktionen umgewandelt werden kann, bietet hier ebenfalls einen möglichen Ansatz zur Wissensakquisition.

- *Kann das Konzept adaptierbarer Kontrollstrategien auch auf prinzipiell andere Planungsverfahren angewendet werden?*
  Im Kontext des deduktiven Planens müßten hier die Verfahren untersucht werden, bei denen die Beweisstruktur nicht wie bei dem in dieser Arbeit angewandten Verfahren direkt mit der Planstruktur assoziiert ist. Verläßt man den deduktiven und insbesondere klassischen Planenkontext, so kann man untersuchen, wie etwa probabilistische Planungsmethoden adaptiv angesteuert werden können. Dabei treten u.a. Schwierigkeiten bei der Spezifikation der Begriffe *korrekter* bzw. *ausführbarer* Plan auf.

- *Auf welche Weise könnte ein um Größenordnungen verbessertes Laufzeitverhalten der Planerrealisierung erreicht werden?*
  Ein aktueller Trend im Bereich Planen befaßt sich mit speziellen Kodierungsverfahren für Planungsprobleme und dem Einsatz von hochspezialisierten Suchmethoden zur Problembeherrschung, um dadurch enorme Laufzeitverbesserungen gegenüber konventionellen Planungsverfahren zu erzielen. Es könnte untersucht werden, in welcher Weise, die starren, aber sehr effizienten Verfahren mit der nötigen Flexibilität zur adaptierbaren Planerzeugung ausgestattet werden könnten.
  Eine weiteres Untersuchungsfeld könnte sich durch eine konsequente Constraintbasierung des Planungsverfahrens ergeben. Die eingeführten Verfeinerungsmethoden bilden durch ihre Variabilität und nahezu freie Kombinierbarkeit hierzu eine gute Grundlage. Man könnte versuchen, den funktionalen Charakter der Beweisstrategien durch ein constraintbasiertes Steuerungsmodell zu ersetzen.

Anhang A

Grammatik verfügbarer Pläne

$$
\begin{array}{rcl}
\texttt{Plan} & ::= & \texttt{S\_Plan} \ ; \ \bigcirc F \ \mid \\
 & & \bigcirc F \ ; \ \texttt{S\_Plan} \ \mid \\
 & & \texttt{Plan} \vee \texttt{Plan} \ \mid \\
 & & \bigcirc F \ \mid \\
 & & \texttt{Plan} \wedge \texttt{Plan} \wedge \Box \ \texttt{MfF} \ \mid \\
 & & \texttt{S\_Plan} \ \mid \\
 & & \texttt{Plan} \ ; \ \texttt{S\_Plan} \\
\texttt{S\_Plan} & ::= & \texttt{S\_Plan} \ ; \ \texttt{S\_Plan} \ \mid \\
 & & \texttt{S\_Plan} \vee \texttt{S\_Plan} \ \mid \\
 & & \texttt{MfF} \wedge \texttt{S\_Plan} \ \mid \\
 & & ex(Aktionsterm) \wedge \odot\bigcirc F \ \mid \\
 & & \bigcirc F \ \mid \\
 & & \texttt{Strong} \wedge \diamondsuit\bigcirc F \ \mid \\
 & & \texttt{Weak} \wedge \diamondsuit\bigcirc F \ \mid \\
 & & \texttt{S\_Plan} \ ; \ T \ ; \ \texttt{MfF} \wedge ex(Aktionsterm) \wedge \odot\bigcirc F \ \mid \\
 & & \texttt{S\_Plan} \ ; \ T \ ; \ \texttt{MfF} \wedge\bigcirc F \ \mid \\
 & & \texttt{S\_Plan} \ ; \ \texttt{MfF} \wedge\bigcirc F \ \mid \\
 & & \diamondsuit[\texttt{MfF} \wedge \bigcirc F] \wedge [\bigcirc F \wedge \odot\bigcirc F] \ \mid \\
 & & \bigcirc\texttt{MfF} \wedge \odot\bigcirc F \ \mid \\
 & & \diamondsuit[\texttt{MfF} \wedge \bigcirc F] \\
\texttt{Strong} & ::= & \Box \ \texttt{MfF} \ \mid \ \texttt{Strong} \wedge \texttt{Length} \\
\texttt{Weak} & ::= & \Box[\bigcirc F \rightarrow \ \texttt{MfF} \ ] \ \mid \\
 & & \texttt{Weak} \wedge \texttt{Length} \ \mid \\
 & & \texttt{Weak} \wedge \texttt{Repair} \\
\texttt{Length} & ::= & \odot^i\bigcirc F \ \mid \ [\neg\diamondsuit \odot^i \bigcirc F];\bigcirc F \ \mid \ \odot^i\bigcirc F;T \\
\texttt{Repair} & ::= & \Box[[\ \texttt{MfF} \rightarrow \texttt{S\_Plan} \ ;T] \vee T] \\
\texttt{MfF} & ::= & \texttt{modalfreie LLP-Formel}
\end{array}
$$

Anhang B

Verwendete Sequenzenregeln

Bemerkung: Die übrigen Regeln bis zu der laufenden Nummer sind in den vorangegangenen Kapiteln bereits beschrieben worden.

Regel 23 ($\wedge$-rechts)

$$\frac{\Gamma \Rightarrow \Lambda, \phi, \ \Delta \quad \Gamma \Rightarrow \Lambda, \psi, \ \Delta}{\Gamma \Rightarrow \Lambda, \phi \wedge \psi, \ \Delta}$$

□

Regel 24 ($\wedge$-links)

$$\frac{\Gamma, \ \phi, \ \psi, \Lambda \Rightarrow \Delta}{\Gamma, \ \phi \wedge \psi, \Lambda \Rightarrow \Delta}$$

□

Regel 25 ($\vee$-rechts)

$$\frac{\Gamma \Rightarrow \Lambda, \phi, \psi \ \Delta}{\Gamma \Rightarrow \Lambda, \phi \vee \psi, \ \Delta}$$

□

Regel 26 ($\vee$-links)

$$\frac{\Gamma, \ \phi, \Lambda \Rightarrow \Delta \qquad \Gamma, \ \psi, \Lambda \Rightarrow \Delta}{\Gamma, \ \phi \vee \psi, \Lambda \Rightarrow \Delta}$$

□

Regel 27 ($\rightarrow$-rechts)

$$\frac{\Gamma, \phi \ \Rightarrow \Lambda, \psi \ \Delta}{\Gamma \Rightarrow \Lambda, \phi \rightarrow \psi, \ \Delta}$$

□

Regel 28 ($\rightarrow$-links)

$$\frac{\Gamma, \Delta \Rightarrow \phi, \Lambda \qquad \psi, \Gamma, \Delta \Rightarrow \Lambda}{\Gamma, \phi \rightarrow \psi, \Delta \Rightarrow \Lambda}$$

□

Regel 29 ($\neg$-links)

$$\frac{\Gamma \Rightarrow \phi, \Lambda}{\Gamma, \neg\phi \Rightarrow \Lambda}$$

□

Regel 30 ($\neg$-rechts)

$$\frac{\Gamma, \phi \Rightarrow \Lambda}{\Gamma \Rightarrow \Lambda, \neg\phi}$$

□

Regel 31 ($\forall$-rechts)

$$\frac{\Gamma \Rightarrow \Lambda, \phi[y/x], \ \Delta}{\Gamma \Rightarrow \Lambda, \forall x \ \phi, \ \Delta}$$

y ist Eigenvariable

□

Regel 32 ($\forall$-links)

$$\frac{\Gamma, \phi[t/x], \forall x \ \phi \Rightarrow \Delta}{\Gamma, \forall x \ \phi \Rightarrow \Delta}$$

□

Regel 33 ($\exists$-rechts)

$$\frac{\Gamma \Rightarrow \Lambda, \phi[t/x],\ \exists x\ \phi,\ \Delta}{\Gamma \Rightarrow \Lambda, \exists x\ \phi,\ \Delta}$$

□

Regel 34 (∃-links)

$$\frac{\Gamma, \phi[y/x], \Rightarrow \Delta}{\Gamma, \exists x\ \phi \Rightarrow \Delta}$$

y ist Eigenvariable

□

Regel 35 (cut-Regel)

$$\frac{\Gamma \Rightarrow \Theta, A \qquad A, \Delta \Rightarrow \Lambda}{\Gamma, \Delta \Rightarrow \Theta, \Lambda}$$

□

Regel 36 (○-Distributivität)

$$\frac{\Gamma \Rightarrow \Lambda, \bigcirc\phi \wedge \bigcirc\psi,\ \Delta}{\Gamma \Rightarrow \Lambda, \bigcirc[\phi \wedge \psi],\ \Delta}$$

□

Regel 37 (*weakening*-rechts)

$$\frac{\Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta,\ \phi}$$

□

Regel 38 (◇-rechts)

$$\frac{\Gamma \Rightarrow \Delta,\ \phi}{\Gamma \Rightarrow \Delta,\ \Diamond\phi}$$

□

Regel 39 (◇-links)

$$\frac{\Gamma^*,\ \phi \Rightarrow \Delta^*}{\Gamma,\ \Diamond\phi \Rightarrow \Delta}$$

$\Gamma^* = \{\Box\psi \mid\ \Box\psi \in \Gamma\}$ *und* $\Delta^* = \{\Diamond\psi \mid\ \Diamond\psi \in \Delta\}$

□

Regel 40 (□-links)

$$\frac{\Gamma,\ \phi \Rightarrow \Delta}{\Gamma,\ \Box\phi \Rightarrow \Delta}$$

□

Regel 41 (◇○-rechts)

$$\frac{\Gamma \Rightarrow \bigcirc\phi \quad \Gamma \Rightarrow \neg\bigcirc F}{\Gamma \Rightarrow \Diamond\phi}$$

□

Regel 42 (if splitting)

$$\frac{\Gamma, \epsilon, \mathsf{P}^+ \Rightarrow \Delta \qquad \Gamma, \neg\epsilon, \mathsf{P}^- \Rightarrow \Delta}{\Gamma, if(\epsilon, \mathsf{P}^+, \mathsf{P}^-) \Rightarrow \Delta}$$

□

Regel 43 (if or splitting)

$$\frac{\Gamma, \alpha, \mathsf{P}^1 \Rightarrow \Delta \qquad \Gamma, \beta, \mathsf{P}^2 \Rightarrow \Delta}{\Gamma,\ \alpha \vee \beta,\ if(\alpha, \mathsf{P}^1, \mathsf{P}^2) \Rightarrow \Delta}$$

□

Literaturverzeichnis

[AAA 91] *Proceedings of the 9th National Conference of the American Association for Artificial Intelligence*, Anaheim, CA, July 1991. MIT Press.

[AAA 96] *Proceedings of the 14th National Conference of the American Association for Artificial Intelligence*, Portland, Oregon, 1996. MIT Press.

[Allen & Ferguson 94] J.F. **Allen** und G. **Ferguson**. *Actions and Events in Interval Temporal Logic.* Technical Report TR-521, University of Rochester, Computer Science Department, 1994.

[Allen & Hayes 89] J.F. **Allen** und P.J. **Hayes**. *Moments and Points in an Interval-based Temporal Logic.* Computational Intelligence, 5:225–238, 1989.

[Allen et al. 91] J.F. **Allen**, H.A. **Kautz**, und R.N. **Pelavin**. *Reasoning About Plans.* Series in Representation and Reasoning. San Mateo, California: Morgan Kaufmann, 1991.

[Allen et al. 94] J.F. **Allen**, L. **Schubert**, G. **Ferguson**, P. **Heeman**, C. **Hwang**, T. **Kato**, M. **Light**, N. **Martin**, B. **Miller**, M. **Poesio**, und D. **Traum**. *The TRAINS Project: A case study in building a conversational planning agent.* TRAINS Technical Note 94-3, University of Rochester, Computer Science Department, 1994.

[Allen 83] J.F. **Allen**. *Maintaining Knowledge about Temporal Intervals.* Comm. ACM, 26:832–843, 1983.

[Allen 84] J.F. **Allen**. *Towards a General Theory of Action and Time.* Artificial Intelligence, 23:123–154, 1984.

[Allen 91] J.F. **Allen**. *Planning as Temporal Reasoning.* In: J.F. Allen, R. Fikes, und E. Sandewall (Hrsg.), Principles of Knowledge Representation and Reasoning: Proc. of the Second International Conference (KR'91), pp. 3–14. San Mateo, CA: Kaufmann, 1991.

[Anderson 94a] P. **Anderson**. *Program Extraction in a Logical Framework Setting.* Technical Report 2261, INRIA, Sophia-Antipolis, 1994.

[Anderson 94b] P. **Anderson**. *Representing Proof Transformations for Program Optimization.* Technical Report 2229, INRIA, Sophia-Antipolis, 1994.

[Andre 95] E. **Andre**. *Ein planbasierter Ansatz zur Generierung multimedialer Präsentationen.* Dissertation, Universität des Saarlandes, Saarbrücken, 1995.

[Antoy et al. 93] S. **Antoy**, R. **Echahed**, und M. **Hanus**. *A Needed Narrowing Strategy.* Technical Report MPII-1993-243, Max-Planck-Institut für Informatik, Saarbrücken, 1993.

[Antoy 91] S. **Antoy**. *Non-determinism and lazy evaluation in logic programming.* In: T.P. Clement und K.-K. Lau (Hrsg.), LOPSTR '91. Proceedings of LOPSTR 91, International Workshop on Logic Program Synthesis and Transformation, Manchester, UK, pp. 318–331, London - Berlin - Heidelberg, 1991. Springer - Verlag.

[Bacchus & Kabanza 95] F. **Bacchus** und F. **Kabanza**. *Using Temporal Logic to Control Search in a Forward Chaining Planner.* In: Ghallab und Milani [Ghallab & Milani 96], pp. 141–153.

[Bacchus & Kabanza 96] F. **Bacchus** und F. **Kabanza**. *Planning for Temporally Extended Goals.* In: AAAI-96 [AAA 96], pp. 1215–1222.

[Bacchus et al. 89] F. **Bacchus**, J. **Tenenberg**, und J. A. **Koomen**. *A Non-Reified Temporal Logic.* In: R. J. Brachman, H. J. Levesque, und R. Reiter (Hrsg.), KR'89: Proc. of the First International Conference on Principles of Knowledge Representation and Reasoning, pp. 2–10. San Mateo, CA: Kaufmann, 1989.

[Bäckström & Nebel 93] C. **Bäckström** und B. **Nebel**. *Complexity Results for SAS^+ Planning.* In: IJCAI-93 [IJC 93], pp. 1430–1435.

[Bäckström & Sandewall 94] C. **Bäckström** und E. **Sandewall** (Hrsg.). *Current Trends in AI Planning: EWSP'93 – 2nd European Workshop on Planning.* IOS Press, Amsterdam, 1994.

[Bäckström 92] C. **Bäckström**. *Computational Complexity of Reasoning about Plans.* Dissertation, Linköping University, Sweden, 1992.

[Basin & Constable 92] D. **Basin** und R. **Constable**. *Metalogical Frameworks.* Technical Report MPI-I-92-205, Max-Planck-Institut für Informatik, Saarbrücken, 1992.

[Basin 94] D. **Basin**. *Logic Frameworks for Logic Programs.* Technical Report MPI-I-94-218, Max-Planck-Institut fuer Informatik, Saarbruecken, 1994.

[Bauer et al. 93] M. **Bauer**, S. **Biundo**, D. **Dengler**, J. **Koehler**, und G. **Paul**. *PHI – A Logic Based Tool for Intelligent Help Systems.* In: IJCAI-93 [IJC 93], pp. 460–466.

[Bauer et al. 96] M. **Bauer**, S. **Biundo**, D. **Dengler**, H. **Feibel**, J. **Koehler**, und G. **Paul**. *RAP–Reasoning about Plans - A Progress Report.* In: J. Sauer, A. Günter, und J. Hertzberg (Hrsg.), Proceedings des 10. Workshop "Planen und Konfigurieren" (PuK-96), pp. 193–204. infix, 1996. Ebenso in Workshop Notes des ECAI'96 Workshop über "Cross-Fertilization in Planning".

[Benyon & Murray 93] D. **Benyon** und D. **Murray**. *Applying user modelling to human-computer interaction design.* Artificial Intelligence Review, 7:199–225, 1993.

[Benyon 93] D. **Benyon**. *Accommodating Individual Differences through an Adaptive User Interface.* In: M. Schneider-Hufschmidt, T. Küme, und U. Malinowski (Hrsg.), Adaptive User Interfaces, pp. 149–165. North-Holland, 1993.

[Benzmüller et al. 97] C. **Benzmüller**, L. **Cheikhrouhou**, D. **Fehrer**, A. **Fiedler**, X. **Huang**, M. **Kerber**, M. **Kohlhase**, K. **Konrad**, E. **Melis**, A. **Meier**, W. **Schaarschmidt**, J. **Siekmann**, und V. **Sorge**. *Ω Mega: Towards a mathematical assistant.* In: W. McCune (Hrsg.), Proceedings 14te International Conference on Automated deduction, Band 1249: LNAI, pp. 252–255, Townsville, North Queensland, Australia, 1997. Springer.

[Berens 97] S. **Berens**. *PiK - Deduktives Planen mit KIV in einer Variante des Dynamischen Logik.* Diplomarbeit, Universität des Saarlandes, Fachbereich Informatik, 1997.

[Bibel et al. 89] W. **Bibel**, L. **Farinas del Cerro**, B. **Fronhöfer**, und A. **Herzig**. *Plan Generation by Linear Proofs: On Semantics.* In: D. Metzing (Hrsg.), GWAI-89: Proc. of the 13th German Workshop on Artificial Intelligence, Eringerfeld, Deutschland, pp. 49–62. Berlin, Heidelberg: Springer, 1989.

[Bibel 83] W. **Bibel**. *Matings in matrices.* Communications of the ACM, 26:844–852, 1983.

[Bibel 86] W. **Bibel**. *A Deductive Solution for Plan Generation.* New Generation Computing, 4:115–132, 1986.

[Bibel 87] W. **Bibel**. *Automated Theorem Proving. 2nd rev. Edition.* Artificial Intelligence. Braunschweig - Wiesbaden: Vieweg, 1987.

[Bibel 92] W. **Bibel** (Hrsg.). *Deduktion: Automatisierung der Logik.* München: Oldenbourg, 1992.

[Bibel 97] W. **Bibel**. *Let's plan it deductively!* In: IJCAI-97 [IJC 97], pp. 1549–1562.

[Bicarregui et al. 94] J. C. **Bicarregui**, J. S. **Fitzgerald**, P. A. **Lindsay**, R. **Moore**, und B. **Ritchie**. *Proof in VDM: A Practioner's Guide.* Formal Approaches to Computing and Information Technology - FACIT. London - Berlin - Heidelberg: Springer - Verlag, 1994.

[Biundo et al. 92a] S. **Biundo**, D. **Dengler**, und J. **Koehler**. *Deductive Planning and Plan Reuse in a Command Language Environment.* In: Neumann [Neumann 92], pp. 628–632.

[Biundo et al. 92b] S. **Biundo**, D. **Dengler**, und J. **Koehler**. *Deductive Planning and Plan Reuse in a Command Language Environment.* Technical Report RR-92-11, Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), 1992.

[Biundo 94] S. **Biundo**. *Present-Day Deductive Planning.* In: Bäckström und Sandewall [Bäckström & Sandewall 94], pp. 1–5.

[Blaesius & Buerckert 92] K. H. **Blaesius** und H.-J. **Buerckert** (Hrsg.). *Deduktionssysteme. Automatisierung des logischen Denkens. 2., voellig ueberarb. und erw. Aufl. 1992.* Muenchen - Wien: Oldenbourg, 1992.

[Blanning & King 93] R. **Blanning** und D. **King**. *Current Research in Decision Support Technology.* IEEE Computer Society Press Tutorial. Washington - Brussels - Tokyo: IEEE Computer Society, 1993.

[Blum & Furst 95] A. **Blum** und M. **Furst**. *Fast planning through planning graph analysis.* In: IJCAI-95 [IJC 95], pp. 1636–1642.

[Blythe & Reilly 93] J. **Blythe** und W. **Reilly**. *Integrating Reactive and Deliberative Planning for Agents.* Technical Report CMU-CS-93-155, Carnegie Mellon University, 1993.

[Boddy & Dean 89] M. **Boddy** und T. **Dean**. *Solving Time-Dependent Planning Problems.* In: IJCAI-89 [IJC 89], pp. 979–984.

[Bowen & Stavridou 93] J.P. **Bowen** und V. **Stavridou**. *Safety-Critical Systems, Formal Methods and Standards.* Sotware Engineering Journal, 8(4):189–209, 1993.

[Boy 91] G. **Boy**. *Intelligent Assistant Systems*, Band 6: Knowledge-Based System Series. London - San Diego - New York: Academic Press, 1991.

[Boyer & Moore 79] R. S. **Boyer** und J. S. **Moore**. *A Computational Logic.* ACM Monograph Series. Orlando - San Diego - New York: Academic Press, 1979.

[Brown 87] F.M. **Brown** (Hrsg.). *The Frame Problem in Artificial Intelligence. Proceedings of the 1987 Workshop, April 12-15, 1987, Lawrence, Kansas*, Los Altos, California, 1987. Kaufmann.

[Bundy et al. 90] A. **Bundy**, F. **van Harmelen**, C. **Horn**, und A. **Smaill**. *The OYSTER-CLAM System.* In: M. E. Stickel (Hrsg.), 10th International Conference on Automated Deduction, pp. 647–648. Berlin, Heidelberg: Springer, 1990.

[Bundy 96] A. **Bundy**. *Proof Planning.* In: Drabble [Drabble 96], pp. 261–267.

[Bylander 91] T. **Bylander**. *Complexity Results for Planning.* In: IJCAI-91 [IJC 91], pp. 274–279.

[Bylander 92] T. **Bylander**. *Complexity Results for Extended Planning.* In: Hendler [Hendler 92], pp. 20–27.

[Calistri-Yeh & Segre 94a] R.J. **Calistri-Yeh** und A.M. **Segre**. *ALPS Year-1 Annual Report.* Technical Report TM-94-0011, ORA, 1994.

[Calistri-Yeh & Segre 94b] R.J. **Calistri-Yeh** und A.M. **Segre**. *The Design of ALPS: An Adaptive Learning and Planning System.* In: Kristian Hammond (Hrsg.), AIPS '94. 2nd International Conference on Artificial Intelligence Planning Systems (AIPS 94), University of Chicago, Chicago, Illinois, June 13-15, 1994: Proceedings, pp. 207–212, Menlo Park, California, 1994. AAAI Press.

[Calistri-Yeh & Segre 96] R.J. **Calistri-Yeh** und A.M. **Segre**. *The Peaks and Valleys of ALPS: An Adaptive Learning and Planning System for Transportation Scheduling.* In: A. Tate (Hrsg.), Advanced Planning Technology, Technological Achievements of the ARPA/Rome Laboratory Planning Initiative, pp. 89–96, Menlo Park, California, 1996. AAAI Press.

[Carbonell & the PRODIGY Group 92] J. G. **Carbonell** and **the PRODIGY Group**. *PRODIGY 4.0: The manual and tutorial.* Technical Report CMU-CS-92-150, Carnegie Mellon University, 1992.

[Cheng & Irani 89] J. **Cheng** und K. **Irani**. *Ordering Problem Subgoals.* In: Proc. of the 11 th IJCAI, pp. 931–936, Detroit, MI, 1989.

[Clocksin & Mellish 87] W.F. **Clocksin** und C.S. **Mellish**. *Programming in Prolog, 3. Edition.* Berlin - Heidelberg - New York: Springer, 1987.

[Constable 86] R.L. **Constable**. *Implementing Mathematics with the Nuprl Proof Development System.* Prentice-Hall, 1986.

[Cranefield 92] S. J. S. **Cranefield**. *A Logical Framework for Practical Planning.* In: Neumann [Neumann 92], pp. 633–637.

[Currie & Tate 91] K. **Currie** und A. **Tate**. *O-Plan: The Open Planning Architecture.* Artificial Intelligence, 52:49–86, 1991.

[Cyranek 91] G. **Cyranek**. *The Computer in the Learning Process: Tutorial Teaching Systems and Their Development Through Artificial Intelligence.* In: Proc. of the International IFIP-GI-Conference on Opportunities and Risks of Artificial Intelligence Systems ORAIS'89, pp. 286–297, Hamburg, Germany, 1991.

[Dean & Wellman 91] T. **Dean** und M.P. **Wellman**. *Planning and Control.* The Morgan Kaufmann Series in Representation and Reasoning. San Mateo, California: Morgan Kaufmann, 1991.

[Dean et al. 90] T. **Dean**, R. J. **Firby**, und D. **Miller**. *Hierarchical Planning Involving Deadlines, Travel Time, and Resources.* In: J. Allen, J. Hendler, und A. Tate (Hrsg.), Readings in Planning, pp. 369–386. San Mateo, CA: Kaufmann, 1990.

[Dengler 94a] D. **Dengler**. *An Adaptive Deductive Planning System.* In: T. Cohn (Hrsg.), Proceedings of the 11th European Conference on Artificial Intelligence, pp. 610–614, Amsterdam, The Netherlands, August 1994. John Wiley & Sons.

[Dengler 94b] D. **Dengler**. *An Adaptive Deductive Planning System.* Technical Report RR-94-06, German Research Center for Artificial Intelligence (DFKI), 1994.

[Dengler 95] D. **Dengler**. *Abstract Plans made for Plan Recognition.* In: M. Bauer (ed.), The Next Generation of Plan Recognition Systems: Challenges for and Insight from Related Areas of AI, pp. 29–32. 1995. Working Notes of a Workshop at IJCAI-95, Montreal.

[Dengler 96a] D. **Dengler**. *Customized Plans Transmitted by Flexible Refinement.* In: Wahlster [Wahlster 96], pp. 609–613.

[Dengler 96b] D. **Dengler**. *Plan Execution in a Temporal Logic Environment.* In: Plan Execution: Problems and Issues, Papers from AAAI Fall Symposium, Technical Report FS-96-01, pp. 41–47. AAAI Press, 1996.

[Dershowitz & Jouannaud 90] N. **Dershowitz** und J. **Jouannaud**. *Rewrite Systems.* In: Jan van Leeuwen (Hrsg.), Handbook of Theoretical Computer Science B: Formal Methods and Semantics, Kapitel 6, pp. 243–320. Amsterdam - New York - Oxford: Elsevier, 1990.

[Dieterich et al. 93] H. **Dieterich**, U. **Malinowski**, T. **Kühme**, und M. **Schneider-Hufschmidt**. *State of the Art in Adaptive User Interfaces.* In: M. Schneider-Hufschmidt, T. Kühme, und U. Malinowski (Hrsg.), Adaptive User Interfaces, pp. 13–48. North-Holland, 1993.

[Dijkstra 76] E. W. **Dijkstra**. *A Discipline of Programming.* Prentice Hall Series in Automatic Computation. Englewood Cliffs, NJ: Prentice Hall, 1976.

[Drabble 96] B. **Drabble** (Hrsg.). *Proceedings of the 3rd International Conference on Artificial Intelligence Planning Systems*, Edinburgh, Scotland, 1996. AAAI Press.

[Drummond 89] M. **Drummond**. *Situated Control Rules.* In: R. J. Brachman, H. J. Levesque, und R. Reiter (Hrsg.), KR'89: Proc. of the First International Conference on Principles of Knowledge Representation and Reasoning, pp. 103–113. San Mateo, CA: Kaufmann, 1989.

[Elkan 89] C. **Elkan**. *Conspiracy Numbers and Caching for Searching And/Or Trees and Theorem-Proving.* In: IJCAI-89 [IJC 89], pp. 341–346.

[Elsom-Cook 93] M. **Elsom-Cook**. *Student modelling in intelligent tutoring systems.* Artificial Intelligence Review, 7:227–240, 1993.

[Elzer et al. 94] S. **Elzer**, J. **Chu-Carroll**, und S. **Carberry**. *Recognizing and Utilizing User Preferences in Collaborative Consultation Dialogues.* In: Proceedings of the 4th International Conference on User Modeling (UM-94), pp. 19–24, 1994.

[Erol et al. 92] K. **Erol**, D. **Nau**, und V. **Subrahmanian**. *When is Planning Decidable?* In: Hendler [Hendler 92], pp. 222–227.

[Erol et al. 94] K. **Erol**, J. **Hendler**, und D. **Nau**. *HTN Planning: Complexity and Expressivity.* In: Proc. of the 12 th AAAI-94, pp. 1123–1128, Seattle, WA, 1994.

[Etzioni 91] O. **Etzioni**. *STATIC: A Problem-Space Compiler for PRODIGY.* In: Proc. of AAAI-91, pp. 533–540, Anaheim, CA, 1991.

[Etzioni 93] O. **Etzioni**. *A Structural Theory of Explanation-Based Learning.* Artificial Intelligence, 60:93–139, 1993.

[Fedeler 97] D. **Fedeler**. *POtA – Taktiken für die Generierung unterbrechbarer, partiell geordneter Handlungspläne in der PHI/RAP-Planungsumgebung.* Diplomarbeit, Universität des Saarlandes, Fachbereich Informatik, 1997.

[Feibel 97] H. **Feibel**. *Ein Konfigurationssystem für Planungsaufgaben in sicherheitskritischen Anwendungen.* Dissertation, Universität des Saarlandes, Saarbrücken, 1997.

[Felty & Howe 94] A. **Felty** und D. **Howe**. *Tactic Theorem Proving with Refinement-Tree Proofs and Metavariables.* In: A. Bundy (Hrsg.), Automated Deduction, CADE-12, pp. 605–619. Berlin, Heidelberg: Springer, 1994.

[Ferguson et al. 96] G. **Ferguson**, J. **Allen**, und B. **Miller**. *TRAINS-95: Towards a Mixed-Initiative Planning Assistant.* In: Drabble [Drabble 96], pp. 70–77.

[Ferguson 95] G. **Ferguson**. *Knowledge Representation and Reasoning for Mixed-Initiative Planning.* Dissertation, University of Rochester, 1995.

[Fields 92] B. **Fields**. *A Guide to Reading VDM Specifications.* Technical Report UMCS-92-12-4, Manchester, 1992.

[Fikes & Nilsson 71] R. E. **Fikes** und N. J. **Nilsson**. *STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving.* Artificial Intelligence, 2:189–208, 1971.

[Fikes & Nilsson 93] R. E. **Fikes** und N. J. **Nilsson**. *STRIPS: A Retrospective.* Artificial Intelligence, 59:227–232, 1993.

[Fink & Veloso 95] E. **Fink** und M. **Veloso**. *Formalizing the PRODIGY Planning Algorithm.* In: Ghallab [Ghallab & Milani 96], pp. 279–289.

[Fink & Yang 93] E. **Fink** und Q. **Yang**. *Characterizing and Automatically Finding Primary Effects in Planning.* In: IJCAI-93 [IJC 93], pp. 1374–1379.

[Foulser et al. 91] D. **Foulser**, M. **Li**, und Q. **Yang**. *A Quantitative Theory for Plan Merging.* In: AAAI-91 [AAA 91], pp. 673–678.

[Francez 92] N. **Francez**. *Program Verification.* International Computer Science Series. Wokingham, England - Reading, Massachusetts - Menlo Park, California: Addison-Wesley, 1992.

[Fromherz 93] M. **Fromherz**. *A Methodology for Executable Specifications.* Dissertation, Universität Zürich, 1993.

[Fuchs & Fromherz 92] N.E. **Fuchs** und M. **Fromherz**. *Schema-based Transformations of Logic Programs.* In: T.P. Clement und K.-K. Lau (Hrsg.), Logic Program Synthesis and Transformation. Springer, 1992.

[Fuchs 92] N.E. **Fuchs**. *Specifications Are (Preferably) Executable.* Software Engineering Journal, pp. 323–334, 1992.

[Gabbay et al. 93] D.M. **Gabbay**, C.J. **Hogger**, und J.A. **Robinson** (Hrsg.). *Handbook of Logic in Artificial Intelligence and Logic Programming. Vol. 1: Logical Foundations.* Handbooks of Logic in Computer Science and Artificial Intelligence and Logic Programming. Oxford: Clarendon Press, 1993.

[Gallier 86] J. H. **Gallier**. *Logic for Computer Science.* Cambridge, PA: Harper & Row, 1986.

[Geib 96] S. **Geib**. *REGgy - Integration von Regressionsstrategien in einem deduktiven Planer.* Diplomarbeit, Universität des Saarlandes, Fachbereich Informatik, Saarbrücken, 1996.

[Gelfond et al. 91] M. **Gelfond**, V. **Lifschitz**, und A. **Rabinov**. *What are the Limitations of the Situation Calculus?* In: R. Boyer (Hrsg.), Automated Reasoning. Essays in Honor of Woody Bledsoe, pp. 167–179. Dordrecht - Boston - London: Kluwer Academic Press, 1991.

[Gentzen 35] G. **Gentzen**. *Untersuchungen über das logische Schließen.* Mathem. Zeitschr., 39:176–210 und 405–431, 1935.

[Georgeff & Lansky 87] M. **Georgeff** und A. **Lansky**. *Reactive Reasoning and Planning.* In: Proceedings of the 6th National Conference of the American Association for Artificial Intelligence, pp. 677–682, Seattle, WA, July 1987.

[Georgeff 90] M. **Georgeff**. *Planning.* In: J. Allen, J. Hendler, und A. Tate (Hrsg.), Readings in Planning, pp. 5–25. San Mateo, CA: Kaufmann, 1990.

[Gervasio & DeJong 92] M. **Gervasio** und G. **DeJong**. *A Completable Approach to Integrating Planning and Scheduling.* Technical Report 1733, University of Illinois at Urbana-Champaign, 1992.

[Ghallab & Milani 96] M. **Ghallab** und A. **Milani** (Hrsg.). *New Directions in AI Planning: EWSP'95 – 3rd European Workshop on Planning.* IOS Press, Amsterdam, 1996.

[Ginsberg & Smith 87] M. **Ginsberg** und D. **Smith**. *Reasoning about Action II: The Qualification Problem.* In: F. M. Brown (Hrsg.), The Frame Problem in Artificial Intelligence, pp. 259–287. Los Altos, CA: Kaufmann, 1987.

[Ginsberg & Smith 88] M. **Ginsberg** und D. **Smith**. *Reasoning About Action I: A Possible Worlds Approach.* Artificial Intelligence, 35:165–195, 1988.

[Girard 87] J.Y. **Girard**. *Linear Logic.* Theoretical Computer Science, 50:1–102, 1987.

[Girard 93] J.Y. **Girard**. *Linear Logic: A Survey.* In: F. L. Bauer, W. Brauer, und H. Schwichtenberg (Hrsg.), Logic and Algebra of Specification, pp. 63–112. Berlin, Heidelberg: Springer, 1993.

[Giunchiglia & Traverso 92] F. **Giunchiglia** und P. **Traverso**. *A Metatheory of a Mechanized Object Theory.* Technical Report 9211-24, IRST, 1992.

[Giunchiglia & Traverso 93] F. **Giunchiglia** und P. **Traverso**. *Program Tactics and Logic Tactics.* Technical Report 9301-01, IRST, 1993.

[Giunchiglia & Traverso 94] F. **Giunchiglia** und P. **Traverso**. *Program Tactics and Logic Tactics.* In: F. Pfennig (Hrsg.), LPAR '94 - Logic Programming and Automated Reasoning. 5th International Conference, Kiev, Ukraine, July 16-21, 1994: Proceedings, Lecture Notes in Artificial Intelligence LNAI - Subseries of Lecture Notes in Computer Science LNAI, pp. 16–30, Berlin - Heidelberg - New York, 1994. Springer.

[Goodman & Litman 92] B. **Goodman** und D. **Litman**. *On the Interaction between Plan Recognition and Intelligent Interfaces.* User Modeling and User-Adapted Interaction, 2(1-2):83–116, 1992.

[Gordon & Melham 93] M. J. C. **Gordon** und T. F. **Melham**. *Introduction to HOL. A Theorem Proving Environment for H igher Order Logic.* Cambridge: Cambridge University Press, 1993.

[Gordon et al. 79] M. J. **Gordon**, A. J. **Milner**, und C. P. **Wadsworth**. *Edinburgh LCF. A Mechanised Logic of Computation.* (Lecture Notes in Computer Science, Vol. LNCS78). Springer, 1979.

[Green 69] C. **Green**. *Application of Theorem Proving to Problem Solving.* In: IJCAI-69 [IJC 69], pp. 219–239.

[Große et al. 92] G. **Große**, S. **Hölldobler**, J. **Schneeberger**, U. **Sigmund**, und M. **Thielscher**. *Equational Logic Programming, Actions, and Change.* In: K. Apt (Hrsg.), Joint International Conference and Symposium on Logic Programming JICSLP'92, pp. 177–191, 1992.

[Große 92] G. **Große** (Hrsg.). *Beyond Sequential Planning, Notes on a workshop held on ECAI'92*, August 1992.

[Grunst et al. 91] G. **Grunst**, R. **Oppermann**, und C. **Thomas**. *Intelligente Benutzerschnittstellen - Kontextsensitive Hilfen und Adaptivität.* Technical report, GMD, Arbeitspapier 553, 1991.

[Guinchiglia et al. 92] F. **Guinchiglia**, P. **Traverso**, A. **Cimatti**, und L. **Spalazzi**. *Tactics: Extending the Notion of Plan.* In: Große [Große 92].

[Gupta & Nau 92] N. **Gupta** und D. **Nau**. *On the Complexity of Blocks-World Planning.* Artificial Intelligence, 56:223–254, 1992.

[Halpern et al. 83] J. **Halpern**, Z. **Manna**, und B. **Moszkowski**. *A Hardware Semantics Based on Temporal Intervals.* LNCS 154, pp. 278–291, 1983.

[Hammond 89] K. **Hammond**. *Case-Based Planning: Viewing Planning as a Memory Task*, Band 1: Perspectives in Artificial Intelligence. Boston - San Diego - New York: Academic Press, 1989.

[Hanks & Weld 95] S. **Hanks** und D. **Weld**. *A Domain-Independent Algorithm for Plan Adaptation.* Journal of Artificial Intelligence Research, 2:319–360, 1995.

[Hanus 94] M. **Hanus**. *The Integration of Functions into Logic Programming: From Theory to Practice.* The Journal of Logic Programming, 19-20:583–628, 1994.

[Harel 79] D. **Harel**. *First-Order Dynamic Logic.* Springer, 1979. Erschienen in der Reihe Lecture Notes in Computer Science, Vol. LNCS68.

[Harper et al. 87] R. **Harper**, F. **Honsell**, und G. **Plotkin**. *A Framework for Defining Logics.* Technical Report ECS-LFCS-87-23 CSR-228-87, University of Edinburgh; Dept. of Computer Science; Laboratory for Foundations of Computer Science, 1987.

[Hayes 89] C.C. **Hayes**. *A Model of Planning for Plan Efficiency: Taking Advantage of Operator Overlap.* In: IJCAI-89 [IJC 89], pp. 949–953.

[Hayes 90] P. J. **Hayes**. *The Frame Problem and Related Problems in Artificial Intelligence.* In: J. Allen, J. Hendler, und A. Tate (Hrsg.), Readings in Planning, pp. 588–595. San Mateo, CA: Kaufmann, 1990.

[Hefley & Murray 93] W. **Hefley** und D. **Murray**. *Intelligent User Interfaces.* In: W. Gray, W. Hefley, und D. Murray (Hrsg.), Proceedings of the 1993 International Workshop on Intelligent User Interfaces, pp. 3–10. ACM Press, 1993.

[Heisel et al. 90] M. **Heisel**, W. **Reif**, und W. **Stephan**. *Tactical Theorem Proving in Program Verification.* In: Proceedings of the 10th Conference on Automated Deduction, pp. 117–131. Springer LNCS 449, 1990.

[Hendler et al. 90] J. **Hendler**, A. **Tate**, und M. **Drummond**. *AI Planning: Systems and Techniques.* AI Magazine, 11:61–77, 1990.

[Hendler 92] J. **Hendler** (Hrsg.). *Proceedings of the 1st International Conference on Artificial Intelligence Planning Systems*, Washington, D.C., 1992. Morgan Kaufmann.

[Hertzberg 89] J. **Hertzberg**. *Planen: Einführung in die Planerstellungsmethoden der Künstlichen Intelligenz*, Band 65: Reihe Informatik. Mannheim - Wien - Zuehrich: BI-Wissenschaftsverlag, 1989.

[Hill & Gallagher 96] P.M. **Hill** und J.G. **Gallagher**. *Meta-Programming in Logic Programming.* In: D. M. Gabbay, C. J. Hogger, und J. A. Robinson (Hrsg.), Handbook of Logic in Artificial Intelligence and Logic Programming. Volume 5. Oxford: Clarendon Press, 1996. Auch erschienen als Research Report 94.22, University of Leeds, School of Computer Studies, UK, 1994.

[Hölldobler & Schneeberger 90] S. **Hölldobler** und J. **Schneeberger**. *A New Deductive Approach to Planning.* New Generation Computing, 8:225–244, 1990.

[Hölldobler 89] S. **Hölldobler**. *Foundations of Equational Logic Programming.* (Lecture Notes in Artificial intelligence LNAI - Subseries of Lecture Notes in Computer Science, Vol. 353). Berlin, Heidelberg: Springer, 1989.

[Huang & Fiedler 96] X. **Huang** und A. **Fiedler**. *Presenting machine-found proofs.* In: M. A. McRobbie und J. K. Slaney (Hrsg.), Proceedings of the Thirteenth International Conference on Automated Deduction (CADE-96), Band 1104: LNAI, pp. 221–225, New Brunswick, NJ, USA, 1996. Springer.

[Huang & Fiedler 97] X. **Huang** und A. **Fiedler**. *Proof Verbalization as an Application of NLG.* In: IJCAI-97 [IJC 97], pp. 965–970.

[Hutter et al. 96] D. **Hutter**, B. **Langenstein**, C. **Sengler**, J. H. **Siekmann**, W. **Stephan**, und A. **Wolpers**. *Verification Support Environment (VSE).* Journal of High Integrity, 1(6), 1996.

[IJC 69] *Proceedings of the 1st International Joint Conference on Artificial Intelligence*, Washington, DC, May 1969.

[IJC 81] *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, Vancouver, Canada, August 1981.

[IJC 89] *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, Detroit, MI, August 1989. Morgan Kaufmann.

[IJC 91] *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, Sydney, Australia, August 1991. Morgan Kaufmann.

[IJC 93] *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, Chambéry, France, August 1993. Morgan Kaufmann.

[IJC 95] *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, Montreal, Canada, 1995.

[IJC 97] *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, Nagoya, Japan, 1997.

[Iwamoto 94] M. **Iwamoto**. *A Planner with Quality Goal and its Speed-up Learning for Optimization Problem.* In: K. Hammond (Hrsg.), AIPS '94. Artificial Intelligence Planning Systems. Proceedings of the 2nd International Conference, University of Chicago, Chicago,, 1994, pp. 281–286, Menlo Park, 1994. AAAI Press.

[Jaffar & Maher 94] J. **Jaffar** und M. J. **Maher**. *Constraint Logic Programming: A Survey.* The Journal of Logic Programming, 19-20:503–581, 1994.

[Jones et al. 93] N.D. **Jones**, C.K. **Gomard**, und P. **Sestoft**. *Partial Evaluation and Automatic Program Generation.* Prentice Hall International Series in Computer Science. New York - London - Toronto: Prentice Hall, 1993.

[Joslin & Pollack 96] D. **Joslin** und M. **Pollack**. *Passive and active decision postponement in plan generation.* In: Ghallab und Milani [Ghallab & Milani 96], pp. 37–48.

[Kambhampati & Srivastava 96] S. **Kambhampati** und B. **Srivastava**. *Universal Classical Planner: An algorithm for unifying State-space and Plan-space Planning.* In: Ghallab und Milani [Ghallab & Milani 96], pp. 81–94.

[Kambhampati 94] S. **Kambhampati**. *Refinement Search as a Unifying Framework for Analyzing Planning Algorithms.* In: J. Doyle, E. Sandewall, und P. Torasso (Hrsg.), Proc. of KR'94, pp. 329–340. Kaufmann, 1994.

[Kambhampati 96] S. **Kambhampati**. *Refinement Planning: Status and Prospectus.* In: AAAI-96 [AAA 96], pp. 1331–1336. Invited Talk.

[Kautz & Selman 96] H. **Kautz** und B. **Selman**. *Pushing the Envelope: Planning, Propositional Logic and Stochastic Search.* In: AAAI-96 [AAA 96], pp. 1194–1201.

[Kautz et al. 96] H. **Kautz**, D. **McAllester**, und B. **Selman**. *Encoding Plans in Propositional Logic.* In: L. Carlucci Aiello, J. Doyle, und S. Shapiro (Hrsg.), Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning, pp. 374–385, Cambridge, MA, USA, 1996. Morgan Kaufmann.

[Kautz 82] H. **Kautz**. *Planning within First-order Dynamic Logic.* Proceedings of CSCSI/SCEIO, pp. 19–26, 1982.

[Knoblock et al. 91] C. A. **Knoblock**, J. D. **Tenenberg**, und Q. **Yang**. *Characterizing Abstraction Hierarchies for Planning.* In: Proc. of AAAI-91, pp. 692–697, Anaheim, CA, 1991.

[Knoblock 94] C. A. **Knoblock**. *Automatically Generating Abstractions for Planning.* Artificial Intelligence, 68:243–302, 1994.

[Kobsa & Wahlster 89] A. **Kobsa** und W. **Wahlster** (Hrsg.). *User Models in Dialog Systems.* Symbolic Computation Artificial Intelligence. Berlin - Heidelberg - New York: Springer, 1989.

[Kobsa 93] A. **Kobsa**. *User Modeling: Recent Work, Prospects and Hazards.* Technical Report 26/93, Universität Konstanz, 1993.

[Koehler 94] J. **Koehler**. *Wiederverwendung von Plaenen in deduktiven Planungssystemen*, Band 65: DISKI - Dissertationen zur KI. St. Augustin: infix, 1994.

[Korf 85] R. E. **Korf**. *Depth-First Iterative-Deepening: An Optimal Admissible Tree Search.* Artificial Intelligence, 27:97–109, 1985.

[Kowalski 79] R. **Kowalski**. *Logic for Problem Solving.* The Computer Science Series. Artificial Intelligence Series. New York - Amsterdam - Oxford: North-Holland, 1979.

[Kowalski 91] R. **Kowalski**. *Logic Programming in Artificial Intelligence.* In: IJCAI-91 [IJC 91], pp. 596–603.

[Kraan et al. 92] I. **Kraan**, D. **Basin**, und A. **Bundy**. *Logic Program Synthesis via Proof Planning.* Technical Report MPI-I-92-244, Max-Planck-Institut für Informatik, Saarbrücken, 1992.

[Kraan et al. 93] I. **Kraan**, D. **Basin**, und A. **Bundy**. *Logic Program Synthesis via Proof Planning.* In: Yves Deville (Hrsg.), LOPSTR '93. Proceedings of LOPSTR 92, International Workshop on Logic Program Synthesis and Transformation,University of Manchester, 2-3 July 1992, Workshops in Computing Series, London - Berlin - Heidelberg, 1993. Springer - Verlag.

[Kröger 87] F. **Kröger**. *Temporal Logic of Programs.* Springer, 1987.

[Lafontaine et al. 91] C. **Lafontaine**, Y. **Ledru**, und P.Y. **Schobbens**. *An experiment in formal software development.* Communications of the ACM, 34(5):62–71, 1991.

[Lee et al. 93] J.J. **Lee**, W. Noris **II**, und P. **Fishwick**. *An Object-Oriented Mulitmodel Design for Integrating Simulation and Planning Tasks.* Technical Report TR-93-019, University of Florida, Department of Computer and Information Sciences, 1993.

[Lowe 91] H. **Lowe**. *Extending the Proof Plan Methodology to Computer Configuration Problems.* Applied Artificial Intelligence, 5:227–252, 1991.

[Lowry & McCartney 91] M. R. **Lowry** und R. D. **McCartney**. *Automating Software Design.* AAAI/MIT Press, 1991.

[Madden 92] P. **Madden**. *Automatic Program Optimization Trough Proof Transformation.* Lecture Notes in Artificial Intelligence 607, pp. 446–460, Saratoga Springs, 1992. Springer, Berlin.

[Manna & Pnueli 81] Z. **Manna** und A. **Pnueli**. *Temporal verification of concurrent programs: the temporal framework for concurrent programs.* In: R.S. Boyer und J.S. Moore (Hrsg.), The Correctness Problem in Computer Science, pp. 215–273. Academic Press, 1981.

[Manna & Waldinger 86] Z. **Manna** und R. **Waldinger**. *How to clear a block: Plan formation in Situational Logic.* In: Proceedings of the Eight Conference on Automated Deduction, pp. 622–640. LNCS 230, Springer-Verlag, 1986.

[Manna & Waldinger 87] Z. **Manna** und R. **Waldinger**. *How to clear a block: A Theory of Plans.* Journal of Automated Reasoning, 3:343–377, 1987.

[Manning et al. 93] A. **Manning**, A. **Ireland**, und A. **Bundy**. *Increasing the Versatility of Heuristic Based Theorem Provers.* In: A. Voronkov (Hrsg.), Logic Programming and Automated Reasoning: Proc. of the 4th International Conference LPAR'93, pp. 194–204. Berlin, Heidelberg: Springer, 1993.

[Masseron et al. 90] M. **Masseron**, C. **Tollu**, und J. **Vauzeilles**. *Generating Plans in Linear Logic.* In: K. V. Nori und C. E. Veni Madhavan (Hrsg.), Foundations of Software Technology and Theoretical Computer Science: Proc. of the Tenth Conference, pp. 63–75. Berlin, Heidelberg: Springer, 1990.

[McAllester & Rosenblitt 91] D. **McAllester** und D. **Rosenblitt**. *Systematic Nonlinear Planning.* In: Proc. of AAAI-91, pp. 634–639, Anaheim, CA, 1991.

[McCarthy & Hayes 69] J. **McCarthy** und P.J. **Hayes**. *Some Philosophical Problems from the Standpoint of Artificial Intelligence.* Machine Intelligence, 4:463–502, 1969.

[McDermott 82] D. **McDermott**. *A Temporal Logic for Reasoning about Processes and Plans.* Cognitive Science, 6:101–155, 1982.

[McDermott 87] D. **McDermott** (Hrsg.). *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, Milan, Italy, August 1987. Morgan Kaufmann.

[McDermott 91] D. **McDermott**. *Regression Planning.* International Journal of Intelligent Systems, 6:357–416, 1991.

[McTear 93] M.F. **McTear**. *User modelling for adaptive computer systems: a survey of recent developments.* Artificial Intelligence Review, 7:157–184, 1993.

[Melis 95] E. **Melis**. *A Model of Analogy-Driven Proof-Plan Construction.* In: IJCAI-95 [IJC 95], pp. 182–188.

[Miller & Shanahan 94] R. **Miller** und M. **Shanahan**. *Narratives in the Situation Calculus.* Journal of Logic and Computation, Special Issue on Actions and Processes, 4(5):513–530, 1994.

[Minton et al. 89] S. **Minton**, J. G. **Carbonell**, C. A. **Knoblock**, D. R. **Kuokka**, O. **Etzioni**, und Y. **Gil**. *Explanation-Based Learning: A Problem Solving Perspective.* Artificial Intelligence, 40:63–118, 1989.

[Nau et al. 90a] D. **Nau**, J. **Hendler**, und Q. **Yang**. *Exploiting Limited Interactions in Plan Optimization.* Technical report, University of Maryland, 1990. CS-TR-2411.

[Nau et al. 90b] D. **Nau**, Q. **Yang**, und J. **Hendler**. *Optimization of Multiple-Goal Plans with Limited Interaction.* In: Workshop on Innovative Approaches to Planning, Scheduling and Control, San Diego, CA, pp. 160–165, 1990.

[Neumann 88] G. **Neumann**. *Meta-Programmierung und Prolog.* Bonn - Reading, Massachusetts - Menlo Park, California: Addison-Wesley, 1988.

[Neumann 92] B. **Neumann** (Hrsg.). *Proceedings of the 10th European Conference on Artificial Intelligence*, Vienna, Austria, 1992. John Wiley & Sons.

[Nwana 91] H.S. **Nwana**. *User Modelling and User Adapted Interaction in an Intelligent Tutoring System.* User Modeling and User-Adapted Interaction, 1:1–32, 1991.

[Ohlbach 93] H.J. **Ohlbach**. *Translation Methods for Non-Classical Logics.* Technical Report MPII-1993-225, Max-Planck-Institut für Informatik, Saarbrücken, 1993.

[Owen 89] S. **Owen**. *Issues in the Partial Evaluation of Meta-Interpreters.* In: H. Abramson und M. H. Rogers (Hrsg.), Meta-Programming in Logic Programming, pp. 319–339. Cambridge, MA: MIT Press, 1989.

[Papadimitriou 94] C. **Papadimitriou**. *Computational Complexity.* Reading, Massachusetts - Menlo Park, California - New York: Addison-Wesley, 1994.

[Paulson 89] L. **Paulson**. *The Foundation of a Generic Theorem Prover.* Automated Reasoning, 5:363–397, 1989.

[Paulson 90] L. **Paulson**. *Isabelle: The Next 700 Theorem Provers.* In: P. Odifredi (Hrsg.), Logic and Computer Science, pp. 361–386. Academic Press, 1990.

[Paulson 91] L. **Paulson**. *ML for the Working Programmer.* Cambridge University Press, 1991.

[Pednault 86] E. **Pednault**. *Toward a Mathematical Theory of Plan Synthesis.* Dissertation, Stanford University, 1986.

[Pednault 87] E. **Pednault**. *Formulating Multiagent, Dynamic-World Problems in the Classical Planning Framework.* In: M. P. Georgeff und A. L. Lansky (Hrsg.), Reasoning about Actions and Plans, pp. 47–82. Los Altos, CA: Kaufmann, 1987.

[Pednault 88] E. **Pednault**. *Synthesizing Plans that Contain Actions with Context-Dependent Effects.* Computational Intelligence, 4:356–372, 1988.

[Penberthy & Weld 92] J. S. **Penberthy** und D. S. **Weld**. *UCPOP: A Sound, Complete, Partial Order Planner for ADL.* In: B. Nebel, C. Rich, und W. Swartout (Hrsg.), Principles of Knowledge Representation and Reasoning: Proc. of the Third International Conference (KR'92), pp. 103–114. San Mateo, CA: Kaufmann, 1992.

[Perez & Carbonell 94] M. **Perez** und J. **Carbonell**. *Control Knowledge to Improve Plan Quality.* In: K. Hammond (Hrsg.), AIPS '94. Artificial Intelligence Planning Systems. Proceedings of the 2nd International Conference, University of Chicago, Chicago, 1994, pp. 323–328, Menlo Park, 1994. AAAI Press.

[Pinto 94] J. **Pinto**. *Temporal Reasoning in the Situation Calculus.* Dissertation, University of Toronto, Canada, 1994.

[Plaisted 90] D. A. **Plaisted**. *Mechanical Theorem Proving.* In: R.B. Banerji (Hrsg.), Formal Techniques in Artificial Intelligence. Elsevier Science Publishers, North-Holland, Amsterdam, 1990.

[Pomberger & Blaschek 93] G. **Pomberger** und G. **Blaschek**. *Grundlagen des Software Engineering. Prototyping und objektorientierte Software-Entwicklung.* München - Wien: Hanser Verlag, 1993.

[Robinson 65] J.A. **Robinson**. *A machine oriented logic based on the resolution principle.* ACM, 12(1):23–41, 1965.

[Rosenschein 81] S. **Rosenschein**. *Plan Synthesis: A Logical Perspective.* In: IJCAI-81 [IJC 81], pp. 331–337.

[Rosner & Pnueli 86] R. **Rosner** and A. **Pnueli**. *A Choppy Logic.* In: IEEE Symposium on Logic in Computer Science, pp. 306–314, 1986.

[Russell & Norvig 95] S. **Russell** und P. **Norvig**. *Artificial Intelligence. A Modern Approach.* Prentice Hall Series in Artificial Intelligence. Englewood Cliffs, NJ: Prentice Hall, 1995.

[Sacerdoti 74] E.D. **Sacerdoti**. *Planning in a hierarchy of abstraction spaces.* Artificial Intelligence, 5:115–135, 1974.

[Sacerdoti 75] E.D. **Sacerdoti**. *The Nonlinear Nature of Plans.* In: Proceedings of the 4th International Joint Conference on Artificial Intelligence, pp. 206–214, Tiblisi, USSR, September 1975.

[Sahlin 93] D. **Sahlin**. *Mixtus: An Automatic Partial Evaluator for Full Prolog.* New Generation Computing, 12:7–51, 1993.

[Sandewall & Rönnquist 86] E. **Sandewall** und R. **Rönnquist**. *A Representation of Action Structures.* In: Proc. of AAAI-86, pp. 89–97, Philadelphia, PA, 1986.

[Sarner & Carberry 92] M.H. **Sarner** und S. **Carberry**. *Generating Tailored Definitions Using a Multifaceted User Model.* User Modeling and User-Adapted Interaction, 2:181–210, 1992.

[Schneeberger 92] J. **Schneeberger**. *Plan Generation by Linear Deduction.* Dissertation, Technische Hochschule Darmstadt, 1992.

[Schoppers 87] M. **Schoppers**. *Universal Plans for Reactive Robots in Unpredictable Environments.* In: McDermott [McDermott 87], pp. 1039–1046.

[Schubert 90] L. **Schubert**. *Monotonic Solution of the Frame Problem in the Situation Calculus: An Efficient Method for Worlds with Fully Specified Actions.* In: H. E. Kyburg, R. P. Loui, und G. N. Carlson (Hrsg.), Knowledge Representation and Defeasible Reasoning, pp. 23–68. Boston: Kluwer, 1990.

[Segre & Scharstein 93] A. **Segre** und D. **Scharstein**. *Bounded-Overhead Caching for Definite-Clause Theorem Proving.* Journal of Automated Reasoning, 11:83–113, 1993.

[Shanahan 97] M. **Shanahan**. *Solving the Frame Problem.* MIT Press, 1997.

[Shoham 87a] Y. **Shoham**. *Temporal Logics in AI: Semantical and Ontological Considerations.* Artificial Intelligence, 33:89–104, 1987.

[Shoham 87b] Y. **Shoham**. *What is the Frame Problem?* In: M. P. Georgeff und A. L. Lansky (Hrsg.), Reasoning about Actions and Plans, pp. 83–98. Los Altos, CA: Kaufmann, 1987.

[Shu 94] H. **Shu**. *A Logic Approach to Planning in Dynamical Systems.* In: C. Bäckström und E. Sandewall (Hrsg.), Current Trends in AI Planning, pp. 278–291. IOS Pess, Amsterdam, 1994.

[Spivey 92] J. M. **Spivey**. *The Z Notation. A Reference Manual. 2nd Edition.* Prentice Hall International Series in Computer Science. New York - London - Toronto: Prentice Hall, 1992.

[Steel 97] S. **Steel** (Hrsg.). *ECP'97 – Fourth European Conference on Planning*, Toulouse, France, 1997.

[Stephan & Biundo 93] W. **Stephan** und S. **Biundo**. *A New Logical Framework for Deductive Planning.* In: IJCAI-93 [IJC 93], pp. 32–38.

[Stephan & Biundo 96] W. **Stephan** und S. **Biundo**. *Deduction-Based Refinement Planning.* In: Drabble [Drabble 96], pp. 213–220.

[Subramanian 93] S. **Subramanian**. *A Mechanized Framework for Specifying Problem Domains and Verifying Plans.* Dissertation, University of Texas at Austin, 1993.

[Tattersall 92] C. **Tattersall**. *A New Architecture for Intelligent Help Systems.* In: C. Frasson, G. Gauthier, und G. I. McCalla (Hrsg.), Intelligent Tutoring Systems: Proc. of the Second International Conference (ITS'92), pp. 302–316. Berlin, Heidelberg: Springer, 1992.

[Thies 94] Markus A. **Thies**. *Planbasierte Hilfeverfahren für direkt-manipulative Systeme. Erkennung, Vervollständigung und Visualisierung von Interaktionsplaenen*, Band 67: DISKI - Dissertationen zur KI. St. Augustin: infix, 1994.

[Traverso et al. 92] P. **Traverso**, A. **Cimatti**, und L. **Spalazzi**. *Beyond the Single Planning Paradigm: Introspective Planning.* In: Neumann [Neumann 92], pp. 643–647.

[Ullman 94] Jeffrey D. **Ullman**. *Elements of ML Programming.* Prentice Hall International Editions. Englewood Cliffs, New Jersey: Prentice Hall, 1994.

[Veloso & Carbonell 93] M. M. **Veloso** und J. G. **Carbonell**. *Toward Scaling Up Machine Learning: A Case Study with Derivational Analogy in PRODIGY.* In: S. Minton (Hrsg.), Machine Learning Methods for Planning, pp. 233–272. San Mateo, CA: Kaufmann, 1993.

[Wahlster et al. 95] W. **Wahlster**, A. **Jameson**, A. **Ndiaye**, R. **Schäfer**, und T. **Weis**. *Ressourcenadaptive Dialogführung: Ein interdisziplinärer Forschungsansatz.* Künstliche Intelligenz, 9(6):17–21, 1995.

[Wahlster 96] W. **Wahlster** (Hrsg.). *Proceedings of the 12th European Conference on Artificial Intelligence*, Budapest, 1996. John Wiley & Sons.

[Waldinger 77] R. **Waldinger**. *Achieving Several Goals Simultaneously.* Machine Intelligence, 8:94–136, 1977.

[Wallen 89] L. A. **Wallen**. *Automated Deduction in Nonclassical Logics.* Cambridge, London: MIT Press, 1989.

[Washington 94] R. **Washington**. *Abstraction Planning in Real Time.* Dissertation, Stanford University, 1994.

[Wikstroem 87] A. **Wikstroem**. *Functional Programming Using Standard ML.* Prentice Hall International Series in Computer Science. London - New York - Toronto: Prentice Hall, 1987.

[Wilkins 88] D. E. **Wilkins**. *Practical Planning: Extending the Classical AI Planning Paradigm.* San Mateo, CA: Kaufmann, 1988.

[Winston & Horn 88] P. **Winston** und B. **Horn**. *LISP.* Massachusetts - Menlo Park, California: Addison-Wesley, 3rd Auflage, 1988.

[Wirsing 90] M. **Wirsing**. *Algebraic Specification.* In: J. van Leeuwen (Hrsg.), Handbook of Theoretical Computer Science: Volume B: Formal Models and Semantics, pp. 675–788. Amsterdam: Elsevier, 1990.

[Zhang & Barringer 94a] Y. **Zhang** und H. **Barringer**. *Constraint Logic Planning.* Technical Report UMCS-94-9-2, Department of Computer Science, University of Manchester, U.K., 1994.

[Zhang & Barringer 94b] Y. **Zhang** und H. **Barringer**. *A Reified Temporal Logic for Nonlinear Planning.* Technical Report UMCS-94-7-1, Department of Computer Science, University of Manchester, U.K., 1994.

[Zhang 94] Y. **Zhang**. *Constraint Logic Planning: A Formal Framework for Nonlinear Planning.* Dissertation, Department of Computer Science, University of Manchester, U.K., 1994.

[Zilberstein 95] S. **Zilberstein**. *On the utility of planning.* In: M. Pollack (Hrsg.), SIGART Bulletin Special Issue on Evaluating Plans, Planners, and Planning Systems, Band 6. 1995.