Eine parallele Architektur zur inkrementellen Generierung multimodaler Dialogbeiträge

Dissertation
zur Erlangung des Grades
des Doktors der Naturwissenschaften
der Technischen Fakultät
der Universität des Saarlandes

von

Norbert Reithinger

Saarbrücken
1991

Tag des Kolloquiums:	16. Oktober 1991
Dekan:	Prof. Dr. R. Maurer
Berichterstatter:	Prof. Dr. W. Wahlster
	Prof. Dr. H. Uszkoreit

Bemerkung zu dieser Version

Lieber Leser,

diese Dissertation wurde vor über 12 Jahren auf einem ATARI ST geschrieben. Für diese Version ist es mir gelungen, die meisten Dateien von alten ATARI-Floppies zu rekonstruieren. Leider fehlen zwei Bildschirmabzüge, die sich nicht mehr finden ließen. Außerdem entspricht die Formatierung nicht ganz dem Original, da sich inzwischen auch LaTex weiterentwickelt hat. Für erste Blicke in die Informatik-Historie ist es aber sicherlich ausreichend.
Wer das Original lesen möchte, sei auf die gedruckte Version verwiesen, die mit dem gleichen Titel im Infix Verlag, St. Augustin, 1992 erschienen ist.

Viel Spaß beim Lesen!

Saarbrücken, im Oktober 2003, Norbert Reithinger

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einführung

1.1 Das Ziel dieser Arbeit

Sind Sie überhaupt Abitur?

Dr. h.c. Franz–Josef Strauß sel.,
ehem. bayerischer Ministerpräsident

Sprache, gesprochene oder geschriebene, ist das primäre Kommunikationsmittel des Menschen. Mit Sprache teilen wir uns mit, an der Fähigkeit, mit Sprache umzugehen, messen wir den Gesprächspartner und versuchen, aus den Äußerungen dessen Einstellungen, Absichten und Emotionen herauszuhören. Sprache ist das "Vielzweckwerkzeug" zur Interaktion mit den Mitmenschen [Hovy, 1987]. [Bußmann, 1983, p.475] definiert Sprache als

> "Auf mentalen Prozessen basierendes, gesellschaftlich bedingtes, historischer Entwicklung unterworfenes Mittel zum Ausdruck bzw. Austausch von Gedanken, Vorstellungen, Erkenntnissen und Informationen, sowie zur Fixierung und Tradierung von Erfahrungen und Wissen."

Vom Menschen werden bei der Produktion und dem Verstehen von gesprochener Sprache erstaunliche kognitive Leistungen erbracht. Ein Mensch spricht etwa 150 Wörter pro Minute, d.h. alle 400 Millisekunden ein Wort [Levelt, 1989, p.199]. Wenn nötig, kann die Sprechgeschwindigkeit verdoppelt werden. Während der Produktion werden vielfältige Entscheidungen getroffen, z.B. welche Wörter die Sprecherintentionen am besten ausdrücken und welche der sozialen Stellung zum Gesprächspartner angemessen sind, welche syntaktischen Konstruktionen verwendet werden, in welcher Reihenfolge was gesagt wird, und vieles andere mehr. Diese Leistungen erbringt der Mensch ohne offensichtliche Anstrengung, da die Prozesse größtenteils unbewußt ablaufen.

Systeme zur Generierung natürlicher Sprache sind bei weitem noch nicht in der Lage, die Performanz und Kompetenz eines menschlichen Sprechers zu erreichen, sowohl hinsichtlich der Flexibilität als auch der Geschwindigkeit. Diese Arbeit untersucht in ihrem theoretischen Teil, wie die Architektur eines Generierungssystems aussehen muß, damit diese beiden Anforderungen auch von einem Computersystem besser erfüllt werden können. Im Anwendungsteil stellt sie das System *POPEL*[1] vor, das auf der Basis der hier entwickelten Architektur schriftliche Dialogbeiträge generiert.

Die Sicht auf den Gebrauch von Sprache im Rahmen dieser Arbeit ist nicht die, daß Sprache als Produkt angesehen wird, das für sich selbst steht, sondern Sprachverwendung im Diskurs wird als Prozeß aufgefaßt [Brown and Yule, 1983]. In diesem Prozeß versuchen die Teilnehmer, Ziele zu erreichen, indem sie sprechen, wobei sie Annahmen über Ziele und Wissen des Gesprächspartners in Betracht ziehen. Diese Annahmen fließen, zusammen mit Wissen über die Strukturierung von Diskursbeiträgen, in den Kommunikationsprozeß ein.

Die zentrale Fragestellung dieser Arbeit ist die des Aufbaus und der Verarbeitungsstrategie eines Generierungssystems, das flexibel ist und es erlaubt, multimodale Dialogbeiträge zu erzeugen. Auf dem Hintergrund einer gegebenen Anwendungssituation in einem natürlichsprachlichen Dialogsystem wird diese Thematik untersucht. Die auf dieser Basis entwickelte Architektur des Systems POPEL führt zu Lösungen, die in Modellen der menschlichen Sprachproduktion Parallelen besitzen.

Die entscheidenden Merkmale der Architektur, die erstmalig in einem implementierten System realisiert wurden, sind:

Inkrementalität: POPEL ist das erste Generierungssystem, das die Verarbeitung vollständig inkrementell durchführt. Inkrementell heißt, daß Teilergebnisse einer Komponente an die nachfolgenden Komponenten schon weitergegeben werden, bevor die Verarbeitung in der Ausgangskomponente vollständig abgeschlossen wurde.

Rückwirkungen: Zwischen den Komponenten des Systems erfolgen Rückwirkungen, so daß eine Komponente Daten, die für die Bearbeitung notwendig sind, aber aufgrund einer unterspezifizierten Eingabe fehlen, bei der Komponente anfordern kann, die diese Daten liefert.

Parallelität: Damit die inkrementelle Verarbeitung mit Rückwirkungen in allen Ebenen gleichzeitig durchgeführt werden kann, arbeiten in POPEL die Komponenten parallel zueinander. Dadurch kommt es zu einer Verzahnung der Verarbeitung zwischen den Komponenten.

Dialogorientierung: Das System erzeugt keine Einzelsätze, sondern alle in einem Dialog auftretenden Äußerungen, die in ihrem Umfang von Ellipsen bis zu kurzen Texten reichen.

Neben der Architektur des Systems, die die *Performanz* maßgeblich bestimmt, stellt die geforderte Flexibilität Anforderungen an die Wissensbasis des Systems, die seine *Kompetenz* beeinflussen:

[1] POPEL ist ein Akronym für "**P**roduction **O**f {**P**erhaps, **P**ossibly, **P**...} **E**loquent **L**anguage".

Deklarativität: Die Wissensquellen des Systems müssen deklarativ dargestellt sein, damit diese wechselnden Anwendungssituationen angepaßt werden können.

Kompositionalität: Die Wissensquellen müssen kompositional aufgebaut sein, so daß eine sukzessive Konstruktion der Repräsentationsstrukturen für die Äußerungen möglich ist.

Kontextuelles Wissen: Das System muß kontextuelle Wissensquellen wie Dialog– und Benutzermodell verwenden, damit Äußerungen verbalisiert werden können, die dem Wissensstand des Dialogpartners angepaßt sind.

Vor allem der letzte Punkt ist entscheidend, wenn das System im größeren Kontext eines natürlichsprachlichen Dialogsystems eingesetzt wird, da dort die Äußerungen des Dialogpartners bei der Generierung berücksichtigt werden müssen. POPEL ist derzeit in das System XTRA (siehe Abschnitt 3.1) eingebunden, einem natürlichsprachlichen Zugangssystem zu Expertensystemen. Der Sprachumfang in diesem Systemkontext, der von POPEL generiert werden muß, reicht von elliptischen Äußerungen bis zu Erklärungen der konzeptuellen Wissensquelle des Systems, die aus mehreren Sätzen bestehen können. Bei der Entwicklung des Systems in diesem Kontext sind die folgenden Aspekte besonders berücksichtigt worden:

Multimodalität: POPEL kann neben sprachlichen Mitteln auch Zeigegesten auf eine Graphik erzeugen, die auf dem Bildschirm dargestellt ist und den gemeinsamen visuellen Kontext des Systems und eines Dialogpartners darstellt. Durch die Kombination von Sprache und Geste ist in vielen Fällen eine effizientere und adäquatere Kommunikation möglich. Dies gilt besonders in Dialogsituationen, in denen eine Referenz auf ein Objekt des visuellen Kontexts, das dem Dialogpartner möglicherweise unbekannt ist, unter Verwendung einer Zeigegeste durchgeführt werden kann.

Flexibilität: Das System ist hinsichtlich zweier Punkte flexibel:

- POPEL ist nicht nur auf einen Anwendungsbereich beschränkt.
- Der Sprachumfang reicht von Äußerungen wie

  Mit dem Bus?

  bis zu kurzen Texten, wie im folgenden Beispiel.

Ein Beispiel für die Arbeitsweise von POPEL ist die Generierung der Definition des Konzepts für steuerlich absetzbare Handlungen. In der konzeptuellen Wissensquelle des Systems hat dieses Konzept den internen Bezeichner `GC49`. Die Eingabe in das Generierungssystem ist der Effekt, der mit der Äußerung erreicht werden soll. In diesem Beispiel ist dies, daß das System möchte, daß eine gemeinsame Überzeugung besteht, daß der Dialogpartner `GC49` kennt:[2]

[2]Eine graphische Darstellung der Struktur dieses Konzepts ist in Abbildung 5.5 (Seite 80) zu sehen. 'SWMB' steht für "**s***ystem* **w***ants that* **m***utual* **b***elief exists*", 'h' für "**h***earer*". Die Äußerungen, bei denen in dieser Arbeit *SAY* bzw. *SYS* vorangestellt ist, werden von POPEL generiert.

```
POPEL-WHAT>(popel '(swmb (know h gc49)))

SAY: steuerhandlungen werden von personen ausgefuehrt
SAY: sie verursachen kosten die abgesetzt werden koennen
SAY: die steuerhandlungen koennen wiederholt werden
```

1.2 Der weitere Aufbau der Arbeit

Mit dieser Arbeit soll der Entwurf und die Realisierung eines leistungsfähigen, flexiblen Generierungssystems für multimodale Dialogbeiträge beschrieben werden, das inkrementell und parallel arbeitet. Der Schwerpunkt liegt dabei auf der Herausarbeitung eines Architekturmodells für ein solches System und dessen prototypischer Realisierung.

Dazu ist es zunächst notwendig, die Aufgaben zu definieren, die ein Generierungssystem zu erfüllen hat, und die einzelnen Komponenten eines solchen Systems vorzustellen. Dieser Überblick wird in Kapitel 2 gegeben. Dabei werden auf der Grundlage des Datenflusses zwischen den Komponenten drei Klassen von Systemarchitekturen herausgearbeitet, die exemplarisch an implementierten Systemen vorgestellt werden. Den drei Generierungssystemen wird danach ein Modell des menschlichen Sprachproduktionsprozesses gegenübergestellt und es wird untersucht, welche Faktoren für dessen Kompetenz und Performanz maßgeblich sind.

Diese Vorarbeiten bilden die Grundlage für den Entwurf der Systemarchitektur von POPEL in Kapitel 3. Zunächst werden die Anforderungen diskutiert, die aus dem Einsatz in einem natürlichsprachlichen Dialogsystem stammen. Aus diesen Rahmenbedingungen werden in diesem zentralen Kapitel auf der Grundlage der herausgearbeiteten Architekturklassen der Aufbau und die inkrementelle und parallele Arbeitsweise von POPEL festgelegt.

POPEL wurde innerhalb eines Dialogsystems entwickelt und kann auf die Wissensbasis und Wissensrepräsentationsformalismen des Gesamtsystems zurückgreifen. Kapitel 4 führt diese ein und gibt ferner einen Überblick über die Kontextwissensquellen im System.

Nachdem die Grundlagen der Architektur und die zur Verfügung stehenden Wissensquellen vorgestellt sind, werden in den Kapiteln 5–7 die einzelnen Verarbeitungskomponenten von POPEL präsentiert. Diese Darstellung umfaßt sowohl das jeweils benötigte Wissen als auch die Verarbeitung dieses Wissens. Bei der Entwicklung wurde darauf geachtet, daß, wenn irgend möglich, Theorien und Methoden verwendet werden, die auch in anderen Systemen eingesetzt werden, und keine Einzellösungen, die speziell auf POPEL zugeschnitten sind. Der Ablauf der einzelnen Schritte der Verarbeitung wird jeweils an ausführlichen Beispielen demonstriert.

Die Auswahl und die Strukturierung des Inhalts in POPEL wird in Kapitel 5 dargestellt. Sie basiert auf einer Operationalisierung der *Rhetorical Structure Theory* und arbeitet inkrementell, d.h. die bereits zur Verbalisierung ausgewählten Teile werden sofort an die weiterverarbeitenden Komponenten übergeben, bevor der Inhalt vollständig bestimmt ist.

Kapitel 6 zeigt, wie nach der Auswahl des Inhalts für diesen in der inhaltsrealisierenden Komponente von POPEL eine semantische und syntaktische Beschreibung erzeugt wird.

Die Realisierungskomponente besteht aus einem verteilten parallelen System unabhängiger, kooperierender Prozesse, die Ausschnitte der Wissensquellen beschreiben und die Transformationen zwischen den Repräsentationsebenen durchführen. Es wird auch gezeigt, wie Anforderungen nach zusätzlichen Daten bei einem Verarbeitungsschritt erzeugt und bearbeitet werden.

In Kapitel 7 wird beschrieben, welches Wissen und welche Verarbeitungsmethoden in POPEL verwendet werden, um Diskursobjekte so zu spezifizieren, daß der Dialogpartner diese identifizieren kann. Die Festlegung des Inhalts von Referenzausdrücken und deren Realisierung erfolgt in einem zweistufigen Prozeß, der besonders Kontextwissen berücksichtigt.

Neben sprachlichen Mitteln verwenden Menschen Zeigegesten, um sich auf Elemente des gemeinsamen visuellen Kontexts zu beziehen. Sie bieten den Vorteil, daß sie oft einfacher und schneller verständlich sind. Die Bedingungen zum Gebrauch von Zeigegesten und die Realisierung einer Komponente zur Generierung simulierter Zeigegesten werden ebenfalls in diesem Kapitel vorgestellt. Die Entwicklung dieser Komponente kann sich dabei auf empirische Untersuchungen zur Verwendung von Zeigegesten in einer speziellen Anwendung stützen.

Abschließend werden in Kapitel 8 einige Details der Implementation von POPEL beschrieben, insbesondere die Simulation der parallelen Verarbeitung. Die qualitative Analyse der Generierung eines Beispielsatzes zeigt, wie die inkrementelle Verarbeitung verläuft und welche Vorteile die parallele Verarbeitung bietet.

Kapitel 9 gibt eine Zusammenfassung der erreichten Ergebnisse und zeigt in einem Ausblick, wie POPEL weiterentwickelt werden kann.

Kapitel 2

Grundlagen und Modelle von Generierungssystemen

Das Gebiet der Generierung natürlicher Sprache umfaßt viele verschiedene Aspekte. So finden sich in der Literatur Arbeiten, die hauptsächlich Probleme der syntaktischen Verarbeitung untersuchen und auf semantisch-pragmatische Fragestellungen nicht eingehen. Andere wiederum beschäftigen sich mit Problemen der Auswahl der Inhalte und setzen das Vorhandensein eines Moduls zur syntaktischen Verarbeitung voraus. Dieses Kapitel gibt einen Überblick über die Aufgaben eines Generierungssystems und beschreibt die zentralen Fragestellungen bei der Entwicklung eines solchen Systems. Anhand dreier Systeme, die vollständig implementiert wurden, wird gezeigt, welche Lösungen dort gefunden wurden. Zum Vergleich wird diesen Systemen ein Modell der menschlichen Sprachproduktion gegenübergestellt.

Ein vollständiger Überblick über die Arbeiten auf dem Gebiet der Generierung natürlicher Sprache ist aufgrund der in den letzten Jahren stark zunehmenden Forschungen kaum mehr möglich (siehe z.B. die Sammelbände [Kempen, 1987b, Zock and Sabah, 1988, Dale *et al.*, 1990, McKeown *et al.*, 1990, Paris *et al.*, 1991]). Zum Abschluß dieses Kapitels wird versucht, einige implementierte Systeme aufgrund ihrer Leistungen und der Architekturprinzipien einzuordnen.

2.1 Was ist die Aufgabe eines Generierungssystems?

Die Aufgabe, natürliche Sprache zu generieren, wird üblicherweise aufgeteilt in die zwei Fragestellungen

a) der *Inhaltsfestlegung*, bei der zu entscheiden ist, *was* gesagt werden soll und

b) der *Inhaltsrealisierung*, bei der zu entscheiden ist, *wie* die Inhalte zu realisieren sind.

In der englischsprachlichen Literatur finden sich dafür die Bezeichnungen *what to say* und *how to say*, oder *strategic* und *tactical component* [McKeown, 1985b]. Diese Trennung existiert nicht erst seit Beginn der Generierung natürlicher Sprache mit Computern, sondern findet sich seit langem in der Sprachphilosophie und Linguistik. Mit den ersten

Generierungssystemen wurden schwerpunktmäßig die Probleme des 'wie' untersucht (vgl. [Simmons and Slocum, 1972, Goldman, 1975, McDonald, 1981]). Erst in den 80er Jahren traten die Arbeiten an den Problemen der Inhaltsfestlegung in den Mittelpunkt des Forschungsinteresses.

2.1.1 Die Festlegung des Inhalts

Bei den ersten Einsätzen von Generierungssystemen erhielten diese von einem Programm, dessen Ausgaben verbalisiert werden sollten, eine Repräsentationsstruktur, die in eine für die realisierende Komponente verständliche Form gebracht und vollständig verbalisiert wurde. Es ergab sich bald das Bedürfnis nach einer Komponente, die diese Repräsentation so strukturiert, daß die daraus erzeugten Sätze kommunikativ adäquat sind. Die Festlegung des Inhalts und dessen Strukturierung wird von Faktoren beeinflußt, die aus der Domäne und dem situativen Kontext stammen. Wissensquellen sind z.B.

- Faktenwissen über die Anwendungssituation, in der das System operiert,
- ein Benutzermodell, das Annahmen über alle die Aspekte des Benutzers repräsentiert, die für das Dialogverhalten des Systems relevant sind (vgl. [Kobsa and Wahlster, 1988]),
- ein Modell der Ziele des Systems,
- ein Diskursmodell, das die Struktur des Diskurses enthält (siehe Abschnitt 4.3),
- rhetorische Ziele (vgl. [Hovy, 1987]),
- Textschemata bzw. Diskursoperatoren (vgl. [McKeown, 1985b, Mann and Thompson, 1987a, Moore and Paris, 1989, Hovy, 1991].

Problematisch ist die Definition der *Eingabe*, von der aus die Generierungsaufgabe in Angriff genommen wird. Für einen 'kognitiv plausiblen' Generator stellt Ward [Ward, 1989] die These auf, daß es keine explixit festgelegte Eingabe gibt, sondern daß diese aus dem Gesamtzustand des Systems besteht. Deswegen ist der Begriff der 'Eingabe' unangebracht, da er suggeriert, daß der Generator ein Modul ist, der nur einige Symbole als Eingabe erhält und ansonst isoliert abläuft, ohne Zugriff auf andere Komponenten des Systems zu haben.

Das System KAMP [Appelt, 1985] erhält z.B. ein intentionales Ziel, das durch Kommunikation mit dem Benutzer erreicht werden soll, und für das das Generierungssystem die entsprechenden Äußerungen finden soll. In der Beispieldomäne 'Reparatur einer Pumpe' soll das System erreichen, daß die Pumpe `PU` nach dem kommunikativen Akt nicht mehr an der Plattform `PL` befestigt ist. Die Eingabe, um dieses Ziel zu erreichen, ist

```
True(¬Attached(PU, PL)).
```

Die Ausgabe, die das System erzeugt, ist eine Aufforderung an den Benutzer, die Schrauben zu lösen. Um sie erzeugen zu können, muß das System über vielfältiges Wissen über den gesamten Kontext verfügen, in dem es operiert (vgl. Abschnitt 2.5).

Ist das kommunikative Ziel gegeben, besteht die Aufgabe der Komponente zur Inhaltsfestlegung darin, Informationen aus den Wissensquellen aufzubereiten und an die Realisierungskomponente zu übergeben, so daß als Ergebnis der Benutzer einen *kommunikativ adäquaten* und *kohärenten* Text als Ausgabe erhält. Dabei ergeben sich zwei Probleme, die nicht unabhängig voneinander gesehen werden können:

- Welche Information wird ausgewählt?
- In welcher Reihenfolge wird sie geäußert?

Das Problem der Auswahl ist, je nach Situation, unterschiedlich komplex. Falls es z.B. das Ziel ist, die Uhrzeit zu erfahren, kann direkt gefragt werden

Wieviel Uhr ist es?

oder mit einem längeren Text

Wie spät ist es jetzt? Ich muß den Zug um zehn noch erreichen.

Es ist aber auch ein indirekter Sprechakt möglich, z.B.

Haben Sie eine Uhr?

Wie man an diesen Sätzen erkennt, muß untersucht werden, ob das Ziel direkt mit einem Sprechakt erreicht werden kann, oder ob zusätzliche Unterziele ausgewählt werden sollen, z.B. um die Frage zu motivieren, oder um die Neigung des Gefragten zu erhöhen, die Frage zu beantworten. Letzteres kann auch erreicht werden, indem ein indirekter Sprechakt gewählt wird, wie bei der konventionalisierten Frage nach der Uhrzeit. Ein Sprechakt kann aber auch mehrere Ziele gleichzeitig realisieren, z.B.

Ich habe den Bastard in Florenz gesehen.

Der Satz informiert darüber, daß sich eine Person in Florenz aufhält und daß der Sprecher eine bestimmte negative Meinung über diese Person hat. Levelt [Levelt, 1989, p.123] bemerkt dazu "This many-to-many mapping from communicative intentions onto speech acts complicates the analysis of macroplanning greatly".

Das zweite Problem, das nicht unabhängig von der Auswahl betrachtet werden kann, ist die Festlegung der Reihenfolge. Das *Linearisierungsproblem* (siehe z.B. [van Dijk and Kintsch, 1983, p.274] oder [Levelt, 1989, p.138]) folgt aus der Tatsache, daß Sprache eine Funktion der Zeit ist. Man beabsichtigt, durch Sprache ein Ziel zu erreichen und bestimmt verschiedene Unterziele, die verbalisiert werden sollen. Die entsprechenden Sprechakte können jedoch nicht alle auf einmal gesagt werden. Es muß festgelegt werden, was zuerst gesagt wird und wie sich weitere Äußerungen kohärent anschließen lassen. Die Reihenfolge kann dabei entscheidend die Interpretation des Gesagten durch den Hörer bestimmen, z.B. in den Sätzen [Levelt, 1989, p.138]:

a) *Sie heiratete und wurde schwanger.*
b) *Sie wurde schwanger und heiratete.*

Levelt und van Dijk & Kintsch nennen einige grundlegende Faktoren, die die Reihenfolge beeinflussen. Diese Faktoren kommen dann zum Tragen, wenn keine kontextuellen oder pragmatischen Bedingungen, z.B. der Wunsch rhetorische Effekte zu erzielen, eine andere Reihenfolge erzwingen. Wenn von diesen Basisfaktoren abgewichen wird, erfolgt in der Äußerung zumeist eine explizte Markierung. Soll in Beispiel (b) die Interpretation der zeitlichen Abfolge wie in Beispiel (a) erreicht werden, kann der Satz z.B. lauten

Sie wurde schwanger, nachdem sie geheiratet hatte.

Levelt teilt die Faktoren ein in inhaltsbezogene und prozeßbezogene. Erstere folgen dem *Prinzip der natürlichen Reihenfolge*. Die natürliche Reihenfolge ist nicht generell definierbar, und domänen– bzw. kulturabhängig, wie das Beispiel mit der Schwangerschaft zeigt. Jedoch liegt die Reihenfolge fest, wenn es z.B. um die Abfolge Schwangerschaft–Geburt geht, die biologisch determiniert ist. Bei Ereignissen, die einer festen zeitlichen Ordnung unterliegen, entspricht die natürliche Reihenfolge, in der diese verbalisiert werden, genau dieser Ordnung.

Prozeßbezogene Faktoren beeinflussen die Reihenfolge der Äußerungen, wenn eine mehrdimensionale Informationsstruktur verbalisiert werden soll, z.B. bei der Beschreibung des Plans einer Wohnung. Hier verbalisiert man zusammenhängende Strukturen nacheinander. Wenn dabei eine Teilstruktur abgearbeitet ist, kehrt man zu der am nächsten benachbart liegenden, noch nicht verbalisierten Teilstruktur zurück. Wenn mehrere Alternativen zur Auswahl stehen, wählt man diejenige, bei der der Zuhörer sich am wenigsten merken muß, um nach ihrer Verbalisierung zum Punkt der Entscheidung durchzukehren. [van Dijk and Kintsch, 1983, p.275f] geben zusätzlich noch Reihenfolgestrategien für Beschreibungen von Objekten, Personen und Zuständen an. Die Reihenfolge ist hier vom Generellen zum Speziellen, vom Übergeordneten zum Untergeordneten, von Mengen zu Elementen, und vom Ganzen zu den Einzelteilen.

2.1.2 Die Realisierung des Inhalts

Hat die inhaltsfestlegende Komponente die Inhalte und deren Reihenfolge festgelegt, können sie von der inhaltsrealisierenden Komponente in eine sprachliche Form überführt werden. Die Verarbeitung in diesem Teil erzeugt zu den konzeptuellen Einheiten, die den Inhalt der Äußerung beschreiben, semantische und syntaktische Strukturen. Mit Hilfe der syntaktischen Information können dann die Lexeme flektiert werden und die Sätze der Äußerung syntaktisch korrekt ausgegeben werden.

Diese Aufgaben sind nur von der jeweiligen Sprache abhängig, nicht jedoch von der Domäne. Deswegen wurden einige inhaltsrealisierende Komponenten als unabhängige Module konzipiert, die an beliebige Programme angeschlossen werden können, deren Ausgabe verbalisiert werden soll. Ein solches System ist z.B. MUMBLE-86 [Meteer, 1990a].

Wissensquellen des realisierenden Teils eines Generators sind

- die Grammatik,
- das Lexikon, das semantische, syntaktische und morphologische Informationen enthält.

Sind beide Wissensquellen deklarativ formuliert, kann das grammatische und lexikalische Wissen einfach erweitert und an neue Domänen angepaßt werden.

Die Eingabe in die inhaltsrealisierende Komponente besteht aus einer Struktur, die entweder in den semantischen Primitiven des Lexikons formuliert ist, oder mittels Transformationsregeln auf diese Primitive abgebildet werden kann. Zudem wird zumeist vorausgesetzt, daß die Eingabe einer Proposition oder mehreren Propositionen entspricht, aus der/denen sich unmittelbar die Satzstruktur ableiten läßt.

Eine typische Eingabestruktur ist die semantische Kasusstruktur, wie sie z.B. das in [Reithinger, 1984] beschriebene System erfordert, um den Satz

Der Zug nach Bremen verläßt Nürnberg um 10.01.

zu erzeugen:[1]

```
(satz_131
  PROPOSITION
      (VERB verlassen NIL)
       (SOURCE nuernberg NIL)
       (TIME 10.01 PRAEP um)
       (OBJECT zug (GOAL bremen NIL) NIL)
  MODALITAET
       TENSE praesens MODE indikativ TYPE aussage FOCUS NIL)
```

2.1.3 Die Auswahl der Wörter

Der Bereich der *Wortwahl* ist derjenige, dessen Zuordnung zu einem Generatorteil am umstrittensten ist. In natürlicher Sprache kann man grob zwei Klassen von Wörtern unterscheiden

- die *Inhaltswörter*, d.h. Nomen, Verben, Adjektive und Adverbien, die den Inhalt und die Struktur der Äußerung bestimmen, und
- die *Funktionswörter*, d.h. Artikel und Präpositionen, die vor allem grammatische Bedeutung tragen. Die Funktionswörter sind in ihrer Anzahl begrenzt und gehören im allgemeinen den geschlossenen Wortklassen an. Deswegen werden sie von [Meteer, 1990b] zum *grammatischen* Wissen einer Sprache gerechnet.

Schwerpunktmässig wurde bis jetzt das Problem der Auswahl von Inhaltswörtern untersucht. Es gibt die Ansicht, daß der Prozeß der Wahl von Inhaltswörtern zur Inhaltsfestlegung gehört [McDonald, 1991], daß er parallel zur Festlegung und Realisierung ablaufen sollte [Danlos, 1987], oder daß er, wie in einem der ersten Ansätze zur Wortwahl mit Diskriminationsnetzwerken, als ein Problem des inhaltsrealisierenden Teils des Generators aufgefaßt werden muß [Goldman, 1975]. In [Marcus, 1987] führten diese Unklarheit und die wenig ausgearbeiteten Methoden der Wortwahl zu der Kritik an Generierungssystemen, daß diese eigentlich überhaupt nicht wissen, was die Wörter, die sie verwenden,

[1] Eine ähnliche Struktur benötigt auch das System *SUTRA* [Busemann, 1984] als Eingabe.

eigentlich bedeuten. Die Verfahren seien so, daß jeder konzeptuellen Einheit genau ein Wort oder eine Phrase zugeordnet ist. Die Wortwahl beschränkt sich dann mehr oder weniger auf einen Zugriff in einer Abbildungstabelle. Weiter schreibt Marcus [Marcus, 1987, p.211]:

> "... and trivializing the problem of lexical semantics to the claim that the meaning of the word can be represented by the same word in upper case, more or less."

Ein weiteres Problem ist die Wahl von Konjunktionen und Partikeln. Diese Wörter, z.B. "außerdem" oder "nur", transportieren primär pragmatische Informationen, die von den Einstellungen der Dialogpartner und deren Absichten abhängen.

2.2 Interaktionen zwischen den Komponenten

Die Schilderung der Zweiteilung der Generierungsaufgabe legt es nahe, für jede Teilaufgabe eine unabhängige Komponente zu implementieren. Zunächst wird der Inhalt festgelegt, worauf dann die Realisierungsphase folgt. Die Übermittlung des Inhalts an der Schnittstelle der zwei Komponenten erfolgt über eine Repräsentation, die einem oder mehreren Sätzen entspricht. Damit erhält man eine unabhängige Realisierungskomponente, die in vielen verschiedenen Systemen eingesetzt werden kann. Dies ist die grundlegende Idee, die z.B. bei der Entwicklung der Realisierungskomponente MUMBLE [McDonald, 1981] und der Weiterentwicklung MUMBLE–86 [Meteer, 1990b] Pate stand (s.o.).

Interessant sind in diesem Zusammenhang Erfahrungen, die Karlin [Karlin, 1985] bei der Anpassung der ihrem Anspruch gemäß *transportablen* inhaltsrealisierenden Komponente MUMBLE–86 an das System ROMPER [McCoy, 1987] gemacht hat. Obgleich ROMPER eine Spezifikation liefert, die nach Propositionen strukturiert ist, stellte es sich als schwierig heraus, diejenige Wissensquelle von MUMBLE zu definieren, die die Eingabe in eine interne Darstellung überführt, die sogenannten *realization classes* und *attachment points*.

Daß die 'historische' Trennung in zwei Teile mehr aus Bequemlichkeit als wegen ihrer Korrektheit beibehalten wird, zeigte sich in vielen Arbeiten. Der Frage, wie Planung und Realisierung zueinander stehen, wurde sogar ein eigener Workshop gewidmet (vgl. [Hovy *et al.*, 1989]).

Daß die Prozesse der Inhaltsfestlegung und Inhaltsrealisierung nicht getrennt werden können, soll am Beispiel der Sätze

a) Ich sehe den Studenten, der ein Moped besitzt.
b) (*)Ich sehe ihn, der ein Moped besitzt.

gezeigt werden. Die Sätze bestehen aus zwei Propositionen, wobei die eine als Hauptsatz, die andere als eingebetteter Relativsatz realisiert wird. In Satz (b) wird die Bezugsphrase für den Relativsatz pronominalisiert, z.B. weil sie derzeit fokussiert ist. Die Frage ist, wo welche Entscheidung getroffen wird. Erhält der realisierende Teil die zwei Propositionen und entscheidet über die Einbettung der Propositionen, hat der Textplaner

keinen Einfluß mehr auf die Effekte, die die Strukturierung der Propositionen im Text bewirken. Andererseits kann der Textplaner nicht das syntaktische Wissen enthalten, daß die Bezugsphrase pronominalisiert werden kann, und daß daraufhin die Einbettung nicht mehr möglich ist.

Bei dem Problem der Strukturierung eines Textes und der Wahl einer bestimmten Satzstruktur kommt Danlos [Danlos, 1987, p.95f] zu dem Ergebnis, daß die Wahl einfacher Sätze, Modifikationen dieser Sätze (z.B. Passivierung), die Festlegung der Reihenfolge und die Zusammenfassung zu komplexen Sätzen zusammenhängen. Wenn diese Entscheidungen in einer festgelegten Abfolge vorgenommen werden, werden bestimmten Teilen des Generators höhere Prioritäten eingeräumt, nach deren Entscheidungen sich die anderen Teile zu richten haben.

Diese Probleme wurden gesehen, als die ersten vollständigen Generierungssysteme entwickelt waren. Da z.B. im System TEXT [McKeown, 1985b] die Priorität auf der Entwicklung der Einzelkomponenten lag, wurde das Problem zwar erkannt, aber nicht weiter behandelt. Das System PAULINE [Hovy, 1987] definiert sog. *decision points*, an denen ein begrenzter Informationsaustausch beider Komponenten stattfindet. Einen völlig anderen Weg geht das System KAMP [Appelt, 1985]. Das Modell zweier Komponenten wird verworfen. Stattdessen wird die Generierung von Sprache als einheitliches Planungsproblem angesehen, bei dem syntaktische Bedingungen einen Backtracking-Prozeß auslösen können, der Auswirkungen bis auf inhaltsfestlegende Entscheidungen hat.

Zusammenfassend lassen sich Generierungssysteme bezüglich ihrer Architektur in drei Klassen einteilen [Reithinger, 1987]:

- *Sequentielle Modelle:* zwischen den Verarbeitungskomponenten besteht nur ein unidirektionaler Datenfluß vom inhaltsfestlegenden zum realisierenden Teil.

- *Modelle mit Rückwirkungen:* eine bidirektionale Kommunikation zwischen den beiden Teilen findet über eine begrenzte, fest definierte Schnittstelle statt.

- *Integrierte Modelle:* zwischen den beiden Komponenten besteht keine Trennung, sie sind beide Facetten ein- und desselben Prozesses.

Vergleicht man diese drei Klassen mit psycholinguistischen Modellen, können interessante Parallelen festgestellt werden. Auch dort kann man diese Dreiteilung finden: sequentiell [Garrett, 1980], interagierend [Bock, 1987] und integriert [Danks, 1977].

Um zu sehen, welche Kriterien bei dem Entwurf von POPEL beachtet werden sollten, werden die drei implementierten Generierungssysteme TEXT, PAULINE und KAMP in ihrem Aufbau und Ablauf in den nächsten Abschnitten vorgestellt. Zum Vergleich wird das derzeit ausgearbeitetste Modell der menschlichen Spracherzeugung, das Modell von Levelt, danebengestellt.

2.3 Das System TEXT

2.3.1 Aufbau und Domäne

Die Aufgabe des Systems TEXT [McKeown, 1985a, McKeown, 1985b] ist es, als Reaktion auf eine Benutzeranfrage einen Text in der Länge eines Abschnittes zu generieren. Das System hat als Diskursdomäne eine Datenbank und kann drei Fragetypen bearbeiten, die den Inhalt der Datenbank betreffen:

1) Was ist über einen Eintrag der Datenbank bekannt?

2) Definiere einen Eintrag!

3) Was ist der Unterschied zwischen zwei Einträgen?

Um diese Fragen beantworten zu können, werden aus der Wissensquelle die notwendigen Informationen selektiert, geordnet und verbalisiert. Schwerpunkt der Arbeit war es zu untersuchen, wie mit Hilfe von Wissen über Textstrukturen und Relevanzkriterien aus dem Inhalt einer Wissensquelle kohärenter Text generiert werden kann.

Abbildung 2.1 (nach [McKeown, 1985b, p.86]) zeigt den Aufbau und den Datenfluß im System. Wie leicht zu sehen ist, folgt der Aufbau dem *sequentiellen Modell*: Daten fließen unidirektional von der inhaltsfestlegenden zur inhaltsrealisierenden Komponente. In der *strategic component* wird eine *Message* erzeugt, die in der *tactical component* verbalisiert wird.

Das System läuft unabhängig von anderen Systemen und ist nicht in ein Dialogsystem integriert. Die Domäne von TEXT sind Anfragen an eine Datenbank der US NAVY, die Daten von Schiffen und deren Waffen enthält. Die Wissensrepräsentation, auf der TEXT arbeitet, ist nicht die Datenbank selbst, sondern ein aus dieser abgeleiteter Formalismus, der mit einer Objekt- und Attributhierarchie arbeitet und einem einfachen semantischen Netz entspricht. Die Transformation der Datenbank in die Wissensrepräsentation von TEXT erfolgt halbautomatisch.

Ein Beispieltext, den TEXT auf die Frage nach der Definition eines Schiffes erzeugen kann, die in der Form `(definition SHIP)` gestellt wird, ist [McKeown, 1985b, p.50]:

> *A ship is a water-going vehicle that travels on the surface. Its surface-going capabilities are provided by the DB attributes DISPLACEMENT and DRAFT.*

2.3.2 Ablauf der Verarbeitung

Der erste Schritt in der Verarbeitung ist die Auswahl des Wissens, das verbalisiert werden soll, der sogenannte *Relevant Knowledge Pool*. In diesem sollen alle diejenigen Teile der Datenbank enthalten sein, die zur Beantwortung der Frage benötigt werden.

Bei der Frage nach einer Definition oder Information zu einem Konzept der Wissensquelle werden das Konzept selbst und alle Unterkonzepte in den *Relevant Knowledge Pool* aufgenommen. Dazu kommen noch alle Attributinformationen zu diesen Konzepten. Bei Vergleichen ist die Auswahl etwas komplizierter, da sie davon abhängt, ob die zu vergleichenden Objekte sehr ähnlich oder sehr verschieden sind. Sind sie ähnlich, werden

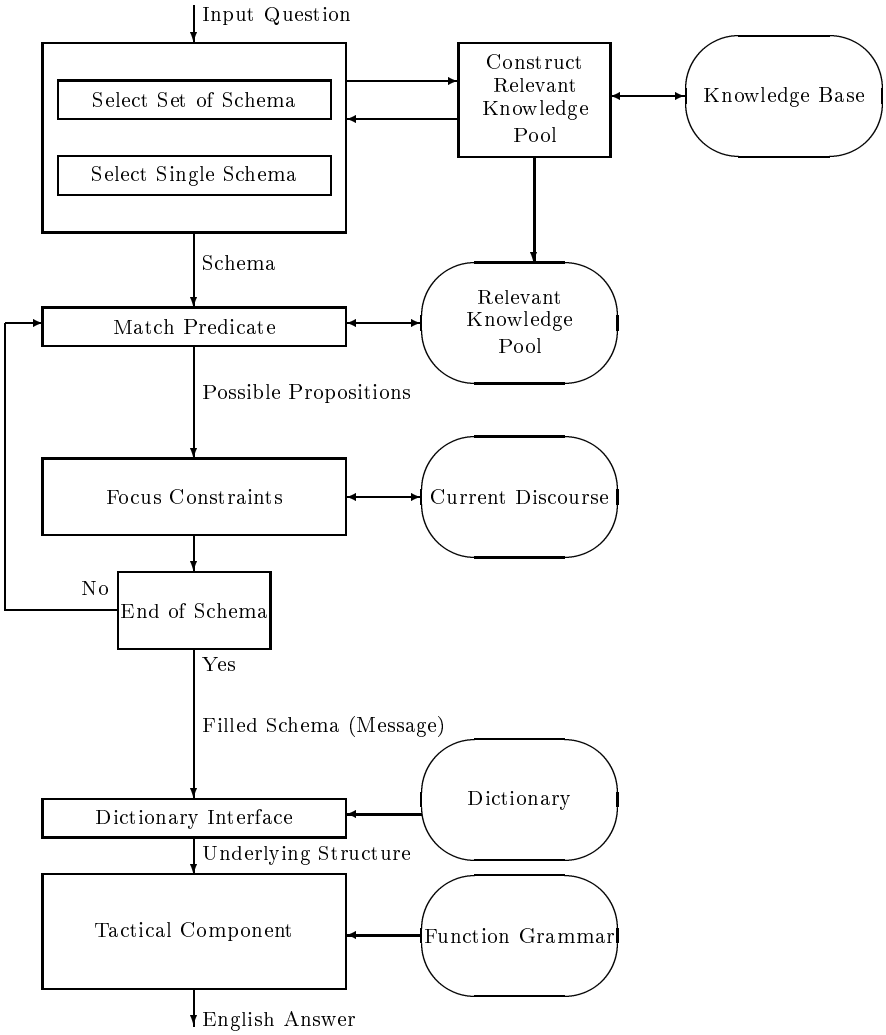


Abbildung 2.1: Der Aufbau von TEXT

diejenigen Teile der Wissensquelle dem *Relevant Knowledge Pool* hinzugefügt, in denen die Objekte sehr unterschiedlich sind. Sind sie sehr verschieden, wäre eine Verbalisierung aller Unterschiede sehr langwierig. Deshalb werden nur die Oberkonzepte aufgenommen, da in der Zugehörigkeit zu unterschiedlichen Teilen der Konzepthierarchie der deutlichste Unterschied besteht.

Die Strukturierung und Linearisierung des *Relevant Knowledge Pools* erfolgt unter Verwendung sogenannter *Schemata*. Die Definition der Schemata und deren Einsatz bei der Bestimmung der Abfolge des Inhalts in Texten ist der wichtigste Beitrag von McKeowns Arbeit. Sie analysierte verschiedene Texte und fand darin immer wieder auftretende Textmuster, die die Grundlage für die Definition der Schemata bilden. Die Schemata bestehen aus *rhetorischen Prädikaten*, die die Beziehungen zwischen den Textteilen beschreiben. Der erste Satz in dem oben angeführten Beispieltext ist ein Beispiel für das Prädikat *identification*. Insgesamt definiert sie zehn solcher Prädikate. Ein Schema legt fest, in welcher Reihenfolge die einzelnen Prädikate auftreten können. Sie sind damit vergleichbar mit Textgrammatiken für Abschnitte.

Mit den Prädikaten werden vier Schemata definiert. Das *constituency* Schema wird gewählt, wenn die Frage nach Definition oder Information gestellt wird und der *Relevant Knowledge Pool* mehr Informationen über Subkonzepte des zu definierenden Objektes enthält als über dieses selbst. Ansonsten wird das *identification* Schema für die Frage nach Informationen zu einem Objekt verwendet, das *attributive* Schema zur Definition und das *contrastive* Schema für Vergleiche zwischen zwei Objekten.

Nach der Auswahl eines Schemas wird dieses dazu verwendet, den *Relevant Knowledge Pool* zu linearisieren. Die Schemata sind als verallgemeinerte Übergangsnetzwerke (*Augmented Transition Network*, ATN) [Winograd, 1983] implementiert.

Im Gegensatz zum Einsatz von ATNs bei der Analyse von Sprache, bei der beim Kantenübergang ein Symbol der Eingabe konsumiert wird, wird in TEXT ein Teil des *Relevant Knowledge Pools* abgearbeitet. An den Kanten des ATNs sind die rhetorischen Prädikate des Schemas annotiert. Ist ein Prädikat auf einen Teil des *Relevant Knowledge Pools* anwendbar, wird dieser konsumiert und eine Teilstruktur der *Message* erzeugt. Die Semantik der Prädikate ist von der jeweiligen Domäne abhängig. Nach der Abarbeitung des *Relevant Knowledge Pools* wird die erzeugte *Message* an die taktische Komponente übergeben.

Für den ersten Satz des oben angeführten Beispieltexts wird das *identification* Prädikat ausgewählt, das als Argument ein Element der Wissensquelle erhält, und in der US-NAVY Datenbankdomäne folgende *Message* erzeugt [McKeown, 1985b, p.52]:

```
(identification SHIP WATER-VEHICLE
   (restrictive TRAVEL-MODE SURFACE)
   (non-restrictive TRAVEL-MEDIUM WATER))
```

Schemata erlauben eine Vielzahl von Alternativen. Welche davon beim Durchlaufen des ATNs ausgewählt wird, um einen kohärenten Text zu erhalten, kontrollieren Fokusbedingungen. Angelehnt an [Sidner, 1979], wo Fokus bei der Interpretation von Sprache untersucht wird, sind in TEXT vier Regeln definiert, nach denen das nächste zu bearbeitende Element des *Relevant Knowledge Pools* ausgewählt wird. Zuerst wird beispielsweise

versucht, über ein Element, das in der letzten Proposition ausgewählt wurde, zusätzliche Informationen zu verbalisieren.

Nachdem das Schema abgearbeitet ist, wird die *Message* an die taktische Komponente übergeben. Als Zwischenstufe dient das *Dictionary Interface*, das Lexikon, in dem die Bezeichner der Wissensquelle, die in der *Message* stehen, durch Wörter und deren funktionale Beschreibung ersetzt werden. Durch die Auswahl des Verbs und dessen funktionaler Beschreibung wird gleichzeitig die semantische Struktur der Propositionen festgelegt. In dem Interface werden aufgrund von Fokusinformationen, die aus der strategischen Komponente stammen, auch die Entscheidungen über Passivbildung und den Gebrauch von Pronomen getroffen. Die Einträge des *Dictionary Interface* müssen per Hand für die jeweilige Anwendung aufgebaut werden, was einen erheblichen Aufwand mit sich bringt.

Die taktische Komponente bestimmt aus den Tiefenstrukturen, die das Interface erzeugt, die syntaktische Struktur und liefert als Resultat englische Sätze. Sie benutzt eine Unifikationsgrammatik, die auf der *Functional Unification Grammar* [Kay, 1979] basiert. Um einen Satz zu erzeugen, wird die Tiefenstruktur eines Satzes mit der Grammatik unifiziert. Das Resultat ist eine syntaktisch vollständige Beschreibung des Satzes, aus der der abschließende Linearisierungsprozeß die textuelle Form extrahiert.

2.3.3 Wertung

Der wichtigste Beitrag, den das System TEXT für das Gebiet der Generierung natürlicher Sprache geleistet hat, ist der Einsatz von Schemata als Mittel zur Auswahl und Linearisierung einer Wissensstruktur. Diese Schemata strukturieren den Aufbau des Texts und legen die Reihenfolge fest, nach denen die Information präsentiert wird. Die Auswahl, welcher Pfad durch das Schema gewählt wird, steuern dabei Fokusprinzipien, die für eine kohärente Abfolge der einzelnen Propositionen sorgen sollen.

TEXT demonstriert diese Funktionalität in einer *statischen* Domäne, die sich im Lauf der Generierung nicht verändert. Über die Repräsentation der Domäne und über die Repräsentationsstruktur werden zudem Annahmen gemacht, die nicht auf andere Domänen oder Repräsentationsformalismen übertragbar sein müssen. Beispiele sind die Begriffe von semantischer Nähe zweier Konzepte in einer Repräsentationshierachie oder von inhaltlichem 'Reichtum' (*richness*) eines Konzeptes. Was nicht berücksichtigt wurde (siehe [McKeown, 1985b, p.171ff, p.204]), aber im Zuge späterer Arbeiten im Umfeld schemabasierter Generierung bearbeitet wurde [McCoy, 1987, Paris and McKeown, 1987], ist der Einsatz von Diskurs- und Benutzermodellen, um die Auswahl und die Abarbeitung der Schemata so vorzunehmen, daß der erzeugte Text dem Wissensstand des Benutzers angepaßt ist. TEXT selbst verwendet keine Kontextwissensquellen, arbeitet nicht in einem Dialogsystem und generiert auf die gleiche Anfrage immer den gleichen Text.

In TEXT sind die beiden Komponenten des Generators strikt getrennt. Die taktische Komponente kann im Prinzip herausgelöst und in einem anderen System eingesetzt werden. Wie in Abschnitt 2.2 bereits erwähnt, wurde diese Trennung als nachteilig erkannt. Da es keine Interaktionsmöglichkeiten gibt, muß die strategische Komponente in der Lage sein, die Informationen zu liefern, die die taktische Komponente benötigt, um syntaktische Entscheidungen und Entscheidungen zur Wortwahl zu treffen.

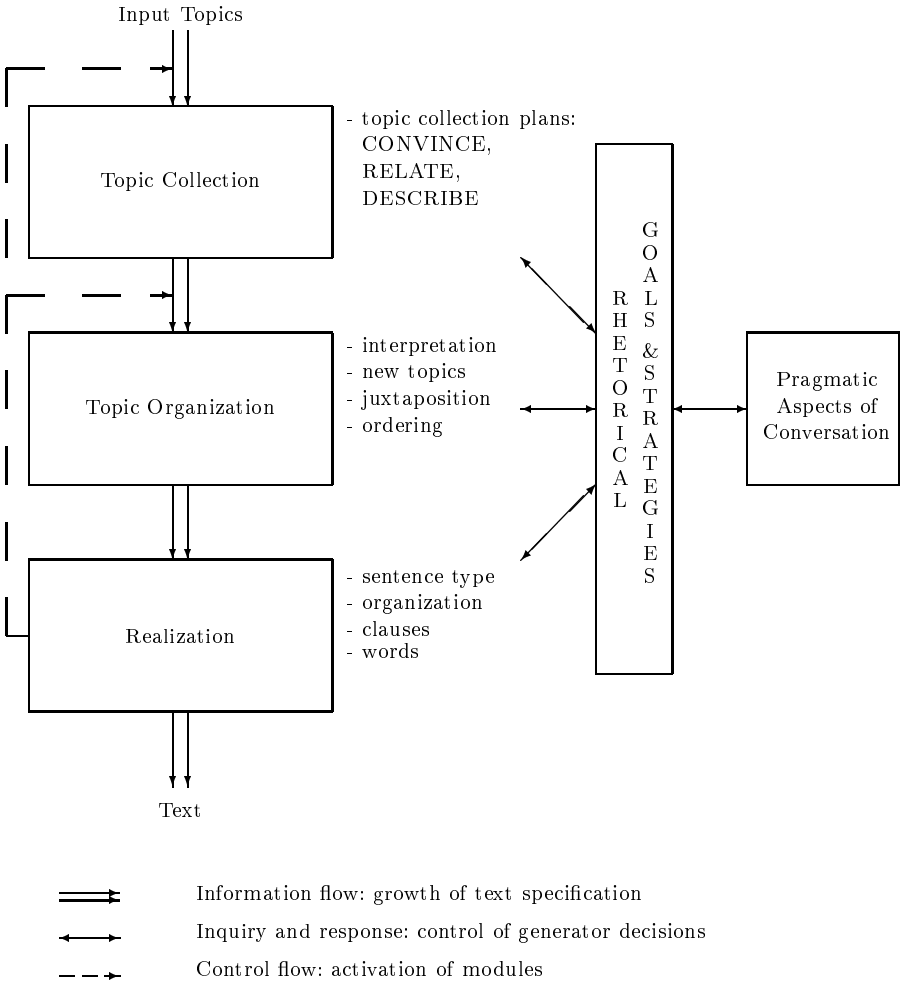


Abbildung 2.2: Der Aufbau von PAULINE

In TEXT sind die Prädikate so aufgebaut, daß sie *Message* Strukturen erzeugen, die im *dictionary interface* in funktionale Strukturen der Grammatik umgesetzt werden können. Damit legen die Prädikate gemeinsam mit den Einträgen des *dictionary* fest, wie die Abbildung von der domänenabhängigen Wissensstruktur in domänenunabhängige sprachliche Beschreibungen vorzunehmen ist.

2.4 Das System PAULINE

2.4.1 Aufbau und Domäne

Schwerpunkt bei der Entwicklung des Systems PAULINE [Hovy, 1985, Hovy, 1987, Hovy, 1988, Hovy, 1990b] war die Untersuchung, welche Auswirkungen pragmatische Informationen wie Sprecherintention, Gesprächssituation und Charakteristika des Hörers auf den zu ge-

nierierenden Text haben. PAULINE wurde nicht als Generierungssystem für eine spezielle Domäne entworfen, vielmehr wurde die Funktionalität mit Texten aus verschiedenen Domänen demonstriert. Die Funktionalität des Systems zeigen zwei Beispieltexte, die aus einer identischen Eingabe stammen, einer Repräsentationsstruktur, die die Besetzung eines Platzes auf dem Campus der Yale-University darstellt, aber mit verschiedenen pragmatischen Voreinstellungen generiert worden sind ([Hovy, 1987, p.8f]):

> *I am angry about Yale's actions. The university had officials destroy a shantytown called Winnie Mandela City on Beinecke Plaza at 5:30 AM on April 14. A lot of concerned students built it in early April. ...*

versus

> *It pisses me off that a few shiftless students were out to make trouble on Beinecke Plaza one day: they built a shantytown, Winnie Mandela City, because they wanted Yale University to pull their money out of companies with buisiness in South Africa. ...*

Der Aufbau des Systems ist in Abbildung 2.2 zu sehen (nach [Hovy, 1990b, p.173]). Zwischen den drei Komponenten, die die Auswahl und Organisation der Inhalte sowie die Realisierung der Sätze durchführen, besteht ein unidirektionaler Datenfluß. In Bezug auf die Architektur des Generators hatte die Konzentration auf pragmatische Phänomene jedoch die Konsequenz, daß dieser eingeschränkte Datenfluß von der Textplanung zur Verbalisierung nicht aufrecht erhalten werden konnte. Zusätzliche Kontrollinformation erlaubt eine Aktivierung der Komponenten zur Auswahl und Organisation des Inhalts durch die zur Realisierung.

Hovy [Hovy, 1985, p.848] zeigt die Notwendigkeit der Rückwirkung an der Generierung der noch in Bearbeitung befindlichen Äußerung

> *the* `[say-age AGE-INSTANCE-65-YEARS]` *woman is homeless.*

Diese Struktur soll durch die Verbalisierung der internen Repräsentation des Alters der Frau so erweitert werden, daß im Hörer Sympathie für die Frau geweckt wird. Die Entscheidung, daß die sprachliche Realisierung des Alters der Frau an dieser Stelle positiv durch ein Adjektiv, z.B. *old* oder *ancient*, ausgedrückt werden kann, ist erst dann möglich, wenn der syntaktische Rahmen bereits festliegt. Das heißt, erst nach der syntaktischen Spezifikation der Umgebung kann geplant werden, daß und wie das Alter gesagt werden kann.

Die Basis der Interaktion zwischen den Verarbeitungsebenen ist die Trennung in top-down und bottom-up Verarbeitung oder in *präskriptive* und *restriktive* Planung [Hovy, 1987, p.176ff]. Top-Down wird über größere Texteinheiten geplant, während bottom-up immer dann vorgegangen wird, wenn aus einer begrenzten Menge von Alternativen eine Auswahl getroffen werden muß. Entscheidend für die Auswahl sind die rhetorischen Mittel, welche die pragmatischen Ziele am besten realisieren.

2.4.2 Ablauf der Verarbeitung

Die Eingabe für PAULINE sind die *input topics*, die ein Teil der Repräsentation der Episode sind und die als *conceptual dependency* Graph vorliegen [Schank, 1975], die Charakteristika der Sprecherin, also PAULINEs, die des Hörers, die der Beziehung zwischen den Partnern und der Konversationssituation.[2] Diese Charakteristika umfassen z.B. die Zeit, die zur Produktion des Textes zur Verfügung steht, die sozialen Relationen zwischen den Partnern, und das Interesse an dem Thema. Aus diesen Werten werden die rhetorischen Ziele des Systems (`RG`) berechnet. Das Ziel `RG:force` beispielsweise, das sich auf die Direktheit und Überzeugungskraft des Textes auswirkt, kann die Werte *forceful*, *neutral* und *quiet* annehmen. Der Wert *forceful* wird unter anderem dann gewählt, wenn der Hörer in seinem Verhalten so beeinflußt werden soll, daß er an der Unterhaltung über das Thema engagiert teilnimmt. Wichtige rhetorische Ziele, die in dieser Phase berechnet werden, sind auch diejenigen, die kontrollieren, wieviel Zeit vorhanden ist, um den Text zu erzeugen (`RG:haste`), und wie detailliert er sein soll (`RG:detail`). Viele der nächsten Schritte stehen unter der Kontrolle dieser beiden Ziele. So kann es in PAULINE vorkommen, daß überhaupt kein Text erzeugt wird, wenn keine Zeit vorhanden ist, um ihn auszuarbeiten.

Zunächst wird zu den gegebenen *input topics* in der Repräsentation nach Teilstrukturen gesucht, die als Kandidaten für die Verbalisierung vorgesehen sind. Die Planung ist an dieser Stelle top-down, die Pläne entsprechen in etwa den Schemata in TEXT. Wenn genug Teilstrukturen gesammelt sind, was u.a. von dem Ziel `RG:haste` abhängt, werden diese interpretiert. Das bedeutet beispielsweise, daß sie daraufhin untersucht werden, wie sie zu einem Begriff zusammengefaßt werden können.

Von den rhetorischen Zielen, die die Einstellungen bestimmen, werden top-down eventuell neue Teile der Repräsentation bestimmt, die in den Text mit aufgenommen werden sollen. Ebenso werden bottom-up, ausgehend von den vorliegenden Wissensstrukturen und generatorspezifischen Regeln, neue Strukturen in der Eingaberepräsentation ausgewählt. PAULINE faßt dann wenn möglich Strukturen zusammen, z.B. eine Abfolge von `schlagen` Ereignissen zu einem `kampf` Ereignis, ordnet sie und bestimmt syntaktische Strukturen. Eine zentrale Stellung hat hierbei das *phrasal lexicon*, das Wörter, Phrasen und syntaktische Regeln in einer homogenen Form enthält.

Die Verarbeitungsschritte werden wiederholt, solange dies möglich ist. Das heißt, nach jedem Schritt wird überprüft, welche Konsequenzen sich aus der neu entstandenen Konfiguration ergeben und welche zusätzlichen Fakten berechnet werden können. An den fünf Entscheidungspunkten (*decision points*) *topic choice*, *sentence content*, *sentence organisation*, *clause organisation* und *word choice* sind Planung und Realisierung verzahnt. Das heißt, daß bottom-up die weitere Planung beeinflußt wird, indem bereits teilweise in ihrer Reihenfolge und syntaktisch-semantischen Struktur festgelegte Teile des Textes den Planer steuern, beeinflußt von den rhetorischen Strategien. Alle Zwischenergebnisse werden in der *text specification* gehalten, einer Liste, deren Elemente entweder *topic goals*, *syntax goals* oder syntaktisch vollspezifizierte Wörter enthalten. Sobald ein solches Wort bestimmt wurde, wird es aus der Liste entfernt und ausgegeben.

[2] Die Darstellung des Verarbeitungsablaufs folgt [Hovy, 1987, p.197ff]

2.4.3 Wertung

Das System PAULINE ist das erste Generierungssystem, das sich hauptsächlich mit der Einbeziehung pragmatischer Information beschäftigt hat. Unter dem Gesichtspunkt der Architektur ist am interessantesten, daß aus dem Einsatz rhetorischer Strategien Auswirkungen auf die Architekur und den Datenfluß des Systems folgten. An den fünf Entscheidungspunkten erfolgt eine Aufhebung der Trennung von Inhaltsfestlegung und Inhaltsrealisierung. Als Begründung nennt Hovy [Hovy, 1985, p.849] folgende Punkte:

Modularität: Der Planer muß nicht alle Entscheidungen im voraus treffen, sondern kann gezielt auf Fragen reagieren, die von der Realisierungskomponente gestellt werden.

Sparsamkeit: Es müssen nicht mehr Entscheidungen getroffen werden als diejenigen, die wirklich notwendig sind.

Angemessenheit: Man kann sprachabhängige Informationen berücksichtigen, wie z.B. den Gebrauch von Redewendungen, die der Planer nicht bestimmen kann.

Nachteilig an PAULINE ist der prozedurale Aufbau des Systems, der eine Übertragung der Algorithmen in andere Generierungssysteme schwer durchführbar erscheinen läßt. Eine klare Übersicht aller Regeln und Interpretationsprozeduren läßt sich aus den vorliegenden Veröffentlichungen kaum gewinnen. Das Problem der prozeduralen Wissensquellen wurde in [Hovy, 1988, p.147] als Schwachstelle des Ansatzes erkannt.

Durch die Festlegung auf die Generierung von Einzeltexten, die von einem vorher festgelegten situativen Kontext ausgehen, der die Einstellungen der verschiedenen rhetorischen Ziele bestimmt, wurden auch Probleme des Dialogkontexts nur marginal behandelt.

2.5 Das System KAMP

2.5.1 Aufbau und Domäne

Das System KAMP [Appelt, 1985] folgt dem integrierten Modell und ist eigentlich mehr als nur ein Generierungssystem, was sich schon in der Abkürzung '**K***nowledge* **a***nd* **M***odalities* **P***lanner*' ausdrückt. KAMP soll sprachliche Aktionen (z.B. Sprechakte) und nichtsprachliche Aktionen (z.B. Zeigegesten), eines Agenten so planen, daß sie dazu beitragen, ein den Kommunikationspartnern gemeinsames Ziel zu erreichen. Das System wurde soweit implementiert, daß es einzelne Sätze generieren kann. Die zentrale Komponente von KAMP ist ein hierarchisches Planungssystem, das auf dem System NOAH [Sacerdoti, 1978] basiert. In Abbildung 2.3 (nach [Appelt, 1985, p.8]) sind die verschiedenen Wissensquellen dargestellt, die ein Planungssystem für Äußerungen benötigt.

In KAMP werden jedoch nur das Weltwissen, Wissen über den Hörer und sprachliches Wissen neben den Plänen berücksichtigt. Die Darstellung der Aktionen und Ziele im System erfolgt in einem 'Mögliche-Welten' Formalismus, der sich an dem Ansatz von [Moore, 1980] orientiert. Da die Sichtweise, daß eine Äußerung eine planbare Aktion ist, bei der Beschreibung syntaktischer Regularitäten nicht mehr angemessen ist

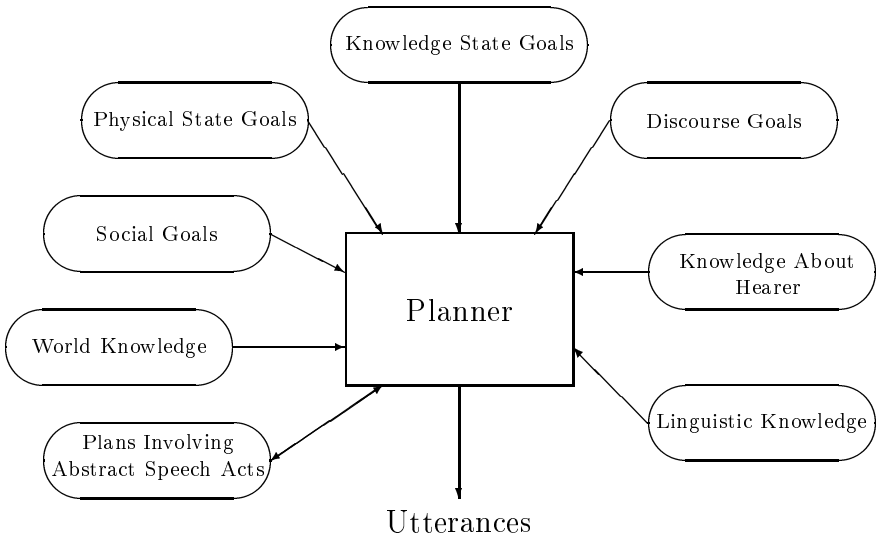


Abbildung 2.3: Der Aufbau eines Planungssystems für Äußerungen

[Appelt, 1985, p.101], wurde das grammatische Wissen mit der Grammatik TELEGRAM dargestellt, die auf der *functional unification grammar* [Kay, 1979] basiert. Sie ist so aufgebaut, daß sie eine Verzahnung mit dem Planungsprozeß zuläßt und enthält dazu in den Regeln spezielle Merkmale. Zudem wurde der Unifikationsalgorithmus so erweitert, daß immer dann der Planer aufgerufen wird, wenn eine funktionale Beschreibung unvollständig ist. Damit ist die Grammatik in den Planungsprozeß mit einbezogen.

Die Funktionalität von KAMP wird demonstriert mit der Erzeugung eines Satzes in einer Domäne, in der es um die Demontage einer Wasserpumpe geht (siehe auch Abschnitt 2.1.1).

2.5.2 Ablauf der Verarbeitung

Die Wissensquellen von KAMP sind die Axiome, die den Zustand der Welt und das private und wechselseitige Wissen der Akteure beschreiben, und Wissen über die sprachbezogenen Aktionen, die das System ausführen kann. Abbildung 2.4 (nach [Appelt, 1985, p.9]) zeigt die Hierarchie dieser Aktionen. Zuoberst stehen die *illokutionären Aktionen*[Searle, 1983], die noch unabhängig von einer sprachlichen Realisierung beschreiben, was mit der Äußerung bewirkt werden soll. Die nächste Ebene sind die *Sprechakte*, die eine abstrakte funktionale Beschreibung des Satzes umfassen, und bei denen die generelle syntaktische Struktur schon festliegt. Darunter liegt die Aktivierung von Konzepten, was sprachlich durch die Generierung einer Nominalphrase oder außersprachlich durch Zeigen auf ein Objekt erfolgen kann. Die letzte Ebene ist die der sprachlichen und gestischen Aktionen, in der die syntaktischen Restriktionen berücksichtigt werden müssen, die bei der Planung auf der Sprechaktebene entstehen.

Als Eingabe erhält KAMP ein Ziel, zu dem ein Plan erzeugt werden soll, und die Axiome. Zu dem Ziel wird ein Planknoten erzeugt und diesem Knoten die Beschreibung

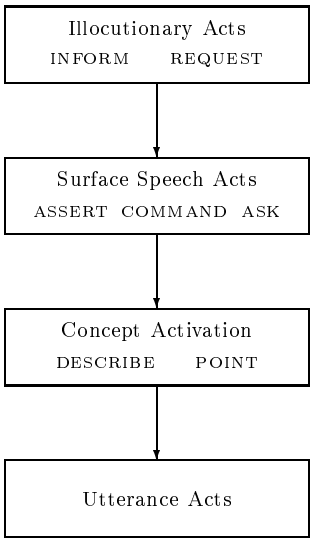


Abbildung 2.4: Die Hierarchie der Aktionen in KAMP

des gegenwärtigen Zustands, eine *mögliche Welt*, zugeordnet. Die Planung erfolgt jetzt so, wie in jedem hierarchischen Planer, der auf NOAH basiert. Zu dem Planknoten werden anwendbare Aktionen gesucht, wobei ein Deduktionssystem überprüft, ob in einer möglichen Welt die Aktion erfolgreich ist. Die möglichen Interaktionen zwischen Planknoten werden mit sogenannten *critics* überprüft, und der Plan, repräsentiert in einem *procedural net*, erweitert. Wenn keine Aktion erfolgreich ist, findet ein Backtracking statt. Dieser Zyklus wird für jede Ebene der Aktionshierarchie wiederholt, bis die Planung die unterste Ebene erreicht hat. Dann wurde ein vollständiger und ausführbarer Plan gefunden.

2.5.3 Wertung

Unter dem Gesichtspunkt der Architektur von Generierungssystemen war der wichtigste Beitrag, den KAMP zur Theorie der Generierung geleistet hat, der, daß die Trennung der beiden Generatorteile aufgehoben werden kann. Erreicht wurde dieses durch die konsequente Betrachtung der Produktion von Äußerungen als Planung von Aktionen zwischen Akteuren. Damit wurde es auch möglich, extralinguistische Handlungen wie das Zeigen auf Gegenstände in die Produktion mit einzubeziehen.

Der Vorteil eines einheitlichen Verarbeitungsmodells ist jedoch zugleich auch eine entscheidende Schwäche des Ansatzes. Durch die notwendigerweise sehr feine Modellierung des Wissens mit einer Modallogik ist es sehr schwer, wenn nicht gar unmöglich, für eine komplexere Domäne eine korrekte Modellierung des Wissens zu erstellen (siehe auch [Moore, 1989, p.56f]). Ähnliche Probleme stellen sich bei der Erweiterung des Leistungsumfangs von Einzelsätzen auf Texte, der Berücksichtigung pragmatischen Wissens, z.B. bei der Wortwahl, oder bei der Integration in ein Dialogsystem. Die Integration in ein Dialogsystem ist kaum möglich, da KAMP als Planer für Einzelsätze konzipiert wurde und die Kohärenz der Sätze nur dann gewährleistet ist, wenn sie Teil eines einzigen Pla-

nes sind [Appelt, 1985, p.120]. Ein weiterer Kritikpunkt an diesem Ansatz ist, daß die Verarbeitung immer wieder von neuem jede Einzelheit plant und kein Wissen über schematisches, konventionalisiertes Sprachverhalten besitzt. Deshalb werden standardisierte Teilpläne immer wieder von Grund auf neu berechnet, was zu einem erheblichen Verarbeitungsaufwand führt. Zudem ist es ein Nachteil, daß das Planungswissen sich auch in Wissensquellen wie der Grammatik befinden muß, die somit nicht mehr unabhängig vom System getestet und verändert werden kann, da sie eng mit den anderen Wissensquellen interagieren muß [Hovy, 1985].

2.6 Ein psycholinguistisches Modell der Spracherzeugung

2.6.1 Aufbau des Modells

Einen Überblick der psycholinguistischen Forschungen auf dem Gebiet der Sprachproduktion gibt [Levelt, 1989]. Dort wird ein einheitliches Modell der Sprachproduktion vorgestellt (siehe Abbildung 2.5, nach [Levelt, 1989, p.9]), das sich auf die Produktion spontaner Sprache beschränkt und die Produktion von geschriebener Sprache ausklammert.

Levelts Modell besteht aus drei Komponenten. Der *Conceptualizer* erzeugt eine Nachricht (*Preverbal Message*), die an den *Formulator* gesendet wird. Dort wird eine Oberflächenstruktur erzeugt und phonologisch kodiert. Die letzte Stufe, der *Articulator*, erhält einen phonetischen Plan, aus dem die Äußerung erzeugt wird. Die Kontrolle dieses Systems wird durch ein parallel dazu existierendes sprachverstehendes System gewährleistet, das die Ausgabe des *Formulators* als interne Sprache, oder die des *Articulators* als geäußerte Sprache analysiert und das jeweilige Analyseergebnis dem *Conceptualizer* zugänglich macht. Dort kann, wenn nötig, eine Korrektur stattfinden.

Levelt stellt die These auf, daß die einzelnen Komponenten des menschlichen Spracherzeugungssystems relativ autonom arbeiten und stützt diese These mit einigen empirischen Untersuchungen. Ein weiteres Charakteristikum ist die *inkrementelle* Verarbeitung in den drei Ebenen. Jede Ebene arbeitet parallel zu den anderen und übergibt Nachrichtenfragmente, die in ihr bestimmt wurden, sofort an die darunterliegende Ebene weiter. Eine Rückwirkung einer 'tiefer' gelegenen Ebene auf einer 'höhere' wird jedoch nicht angenommen. Das System ist so aufgebaut, daß die Komponenten dem Prinzip der *Informational Encapsulation* folgen, d.h. daß die Eingabe in eine Komponente eng begrenzt ist und daß ihre Arbeit von anderen Komponenten möglichst wenig beeinflußt wird.

Die Grundlage der Verarbeitung im *Conceptualizer* sind die kontextuellen Wissensquellen, die ein Diskursmodell, Wissen über die Situation, allgemeines Weltwissen und anderes mehr umfassen. Auch das Lexikon hat eine wichtige Stellung im Modell, da die Generierung lexikongesteuert abläuft. Jeder Eintrag im mentalen Lexikon enthält Daten zur Semantik, Syntax, Morphologie und Phonologie eines Wortes. Ein Resultat der Untersuchung verschiedener Ansätze des Lexikonzugriffs ist, daß, da etwa 150 Wörter pro Minute gesprochen werden, der Zugriff auf das Lexikon parallel erfolgen muß. Sequentielle Verfahren sind zu langsam. Aus der Untersuchung von Sprachfehlern, z.B. Ersetzungen ei-

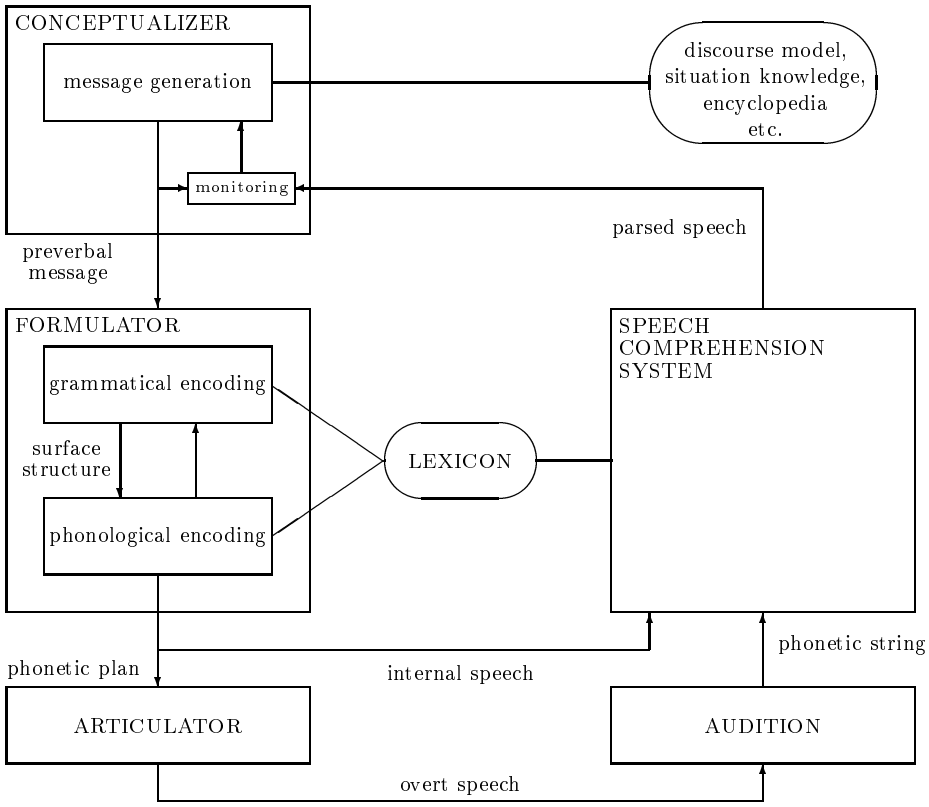


Abbildung 2.5: Levelts Entwurf des Sprachverarbeitungssystems

nes Wortes durch ein anderes, wird gezeigt, wie mehrere Wörter parallel aktiviert werden und welche Verbindungen zwischen den einzelnen Einträgen bestehen müssen.

2.6.2 Ablauf der Verarbeitung

Die Generierung einer *Message* im *Conceptualizer* erfolgt in zwei Stufen, der *Makro-* und *Mikroplanung*. In der ersten Stufe wird die Sprecherintention als eine Folge von Zielen ausgearbeitet und es werden Informationen gesammelt, die diese Ziele ausdrücken. Hierbei ist es häufig so, daß nur solche Informationen mitgeteilt werden, die es dem Hörer ermöglichen, die Sprecherintention zu erschließen. Andererseits werden Objekte oftmals mit redundanten Attributen bezeichnet, so daß hier eine Verletzung der Griceschen Ökonomiemaxime vorliegt [Grice, 1975]. In der *Makroplanung* wird auch das Linearisierungsproblem gelöst, nämlich in welcher Reihenfolge etwas gesagt wird. Die *Mikroplanung* bestimmt für zu verbalisierende Entitäten Informationen, aus denen der Hörer ableiten kann, wo der Referent gefunden werden kann, z.B. in gegenwärtig fokussierter Information. Zudem werden hier noch Informationen über Perspektive und sprachspezifische Eigenheiten kodiert, z.B. welche zeitlichen Relationen gesagt werden müssen.

Als Modell für die Struktur der *Message* schlägt Levelt eine propositionale Darstellung vor, z.B. eine logische Prädikat–Argumentstruktur. Einflußfaktoren beim Aufbau der

Message sind Informationen, die aus der Einstellung des Sprechers kommen, und auch Anforderungen, die aus der Zielsprache stammen. Als Beispiel werden die voneinander abweichenden deiktischen Systeme des Englischen und Japanischen angeführt.

Der nächste Schritt ist die Generierung der Oberflächenstruktur im *Formulator*. Diese Struktur hängt stark von der jeweiligen Sprache ab. Als Gegenpole werden das Englische mit fester Wort- und Phrasenstellung und Malayalam präsentiert, eine südindische Sprache, die eine freie Wortstellung besitzt. Die Wortstellung des Deutschen liegt zwischen diesen Extremen.

Die Generierung der Oberflächenstruktur verläuft größtenteils automatisch, d.h. sie unterliegt nicht der bewußten Kontrolle des Sprechers. Als ein Modell für die Verarbeitung auf dieser Ebene nennt Levelt die *Incremental Procedural Grammar* [Kempen and Hoenkamp, 1987]. Diese erfüllt die Anforderungen, die aus der Sicht eines Psycholinguisten an ein syntaktisches System gestellt werden müssen: sie ist lexikonbasiert, generiert inkrementell, akzeptiert die Fragmente der Eingabe in beliebiger Reihenfolge und verarbeitet Konstituenten parallel.

Die meisten maschinellen Generierungssysteme haben nach dieser Stufe der Verarbeitung noch ein Flektionsmodul, womit die Generierungsarbeit als beendet betrachtet wird. Dabei wird übersehen, daß die Absichten, die der Sprecher mit einer gesprochenen Äußerung übermitteln will, oftmals entscheidend von der Intonation abhängen. Dazu muß das System neben morphologischem Wissen zu den einzelnen Wörtern auch Wissen zur Betonung, Segmentierung, Metrik und Tonhöhe haben, um die phonetischen Pläne bei fließend gesprochener Sprache erzeugen zu können. Das erfordert einen Prosodiegenerator, der für die einzelnen Segmente Rhythmus und Betonung bestimmt.

Am Ende des Generierungsmodells steht die Artikulation, eine der komplexesten motorischen Fähigkeiten des Menschen. Hier wird anhand von empirischen Untersuchungen gezeigt, daß die phonetische Planung als atomare Einheiten Silben benutzt. Falls das Wort mehr als zwei Silben hat, müssen Teile der zweiten bereits verfügbar sein, damit die erste ausgesprochen werden kann.

2.6.3 Selbstkontrolle und Selbstkorrekturen

Während der Mensch spricht, überwacht er von der konzeptuellen Ebene bis zur Artikulation, ob er das, was er sagt, auch wirklich sagen will. Die Aufmerksamkeit ist dabei unterschiedlich stark: die Kontrolle nimmt zum Phrasenende hin zu. Levelt nennt zwei Theorien, mit denen die Überwachung durchgeführt wird: die *editierende* und die *konnektionistische*. Die Methode der Überwachung, die Levelt für die plausibelste hält, ist die editierende *Perceptual-Loop Theory*, bei der das sprachverstehende System als Kontrollinstanz arbeitet. Bei ihr wird die Produktion der Sprache auf zwei verschiedene Arten überwacht: intern, indem der phonetische Plan an das sprachverstehende System übergeben wird, und extern, wobei hier über das Gehör die Sprache wieder analysiert wird. Die konnektionistische Theorie nimmt an, daß das sprachproduzierende System mit dem analysierenden identisch ist und daß die Überwachung und Fehlerkorrektur durch eine rückwärts laufende Aktivierung erfolgt.

Die Kontrolle beschränkt sich jedoch nicht nur auf interne Verarbeitungsprozesse. Der

Sprecher beobachtet immer auch die Reaktionen des Zuhörers und reagiert sofort, wenn er wahrnimmt, daß seine Äußerung nicht verstanden wird.

Wird bei der Kontrolle ein Fehler entdeckt, unterbricht sich der Sprecher sofort selbst. Dabei zeigt sich, daß Unterbrechungen während des Sprechens nicht an Phonem- oder Phrasengrenzen gebunden sind. Eine Unterbrechung während der Produktion eines Wortes tritt nur dann auf, wenn dieses selbst fehlerhaft ist. Die Korrektur erfolgt so, daß immer möglichst viel von der Struktur der ursprünglichen Äußerung erhalten bleibt.

2.6.4 Wertung

Levelt beschreibt in einem großen Bogen das am besten funktionierende Sprachproduktionssystem, nämlich das des Menschen, und führt dabei Ergebnisse und Methoden der Psycholinguistik, der Linguistik und der Künstlichen Intelligenz in einem Überblick zusammen. Hierbei mußte er, wie er im Vorwort schreibt, mit der Fülle an Literatur kämpfen, die aber auf die vielen verschiedenen Disziplinen aufgeteilt ist. Diese Zusammenführung verschiedenster Forschungsfelder erfolgt mit der präsentierten Architektur als Klammer. Die mit dem Modell aufgestellten Hypothesen werden dabei immer an Ergebnissen gemessen, die mit Untersuchungen der menschlichen Sprache und des Sprechens gewonnen wurden. Es wird des öfteren deutlich, wo Hypothesen, die dem Modell zugrundeliegen und die teilweise auch Basis für die Arbeit an Systemen der Künstlichen Intelligenz sind, plausibel erscheinen und wo noch Lücken zu füllen sind.

Eine dieser Lücken nennt Levelt selbst. Im Gegensatz zu seiner Annahme der *informational encapsulation* gibt es zumindest zwischen den Ebenen der grammatischen und phonologischen Verarbeitung starke Evidenzen dafür, daß hier eine Rückwirkung stattfindet. Auch bei der Generierung von Nominalphrasen liegen neuere Untersuchungen vor [Pechmann and Zerbst, 1990], die das Prinzip zu wiederlegen scheinen.

Interessant an dem Modell ist die *performanzorientierte* Sicht auf die Sprachproduktion, aus der sich die Forderungen nach *Inkrementalität* und *Parallelität* des Produktionsprozesses ergeben, wenn in 'Realzeit' gesprochen werden soll. Module, welche die *Kompetenz* des System bestimmen, also beispielsweise Grammatik und Lexikon, müssen den aus dem Verarbeitungsablauf vorgegebenen Restriktionen genügen, um plausibel in das Modell integrierbar zu sein.

2.7 Vergleichende Einordnung einiger Generierungssysteme

Wie in der Einleitung zu diesem Kapitel gesagt wurde, ist ein vollständiger Überblick über die Arbeiten auf dem Gebiet der Generierung natürlicher Sprache aufgrund der in den letzten Jahren stark zunehmenden Forschungen kaum mehr möglich.

In Abbildung 2.6 sind einige Systeme aufgeführt, die anhand der Teile der Aufgaben, die sie durchführen, ihrer Architektur und des verwendeten Wissens eingeordnet sind. Die Kriterien für die Auswahl der Systeme sind die Verfügbarkeit von Literatur zu den einzelnen Systemen und die möglichst vollständige Implementation.

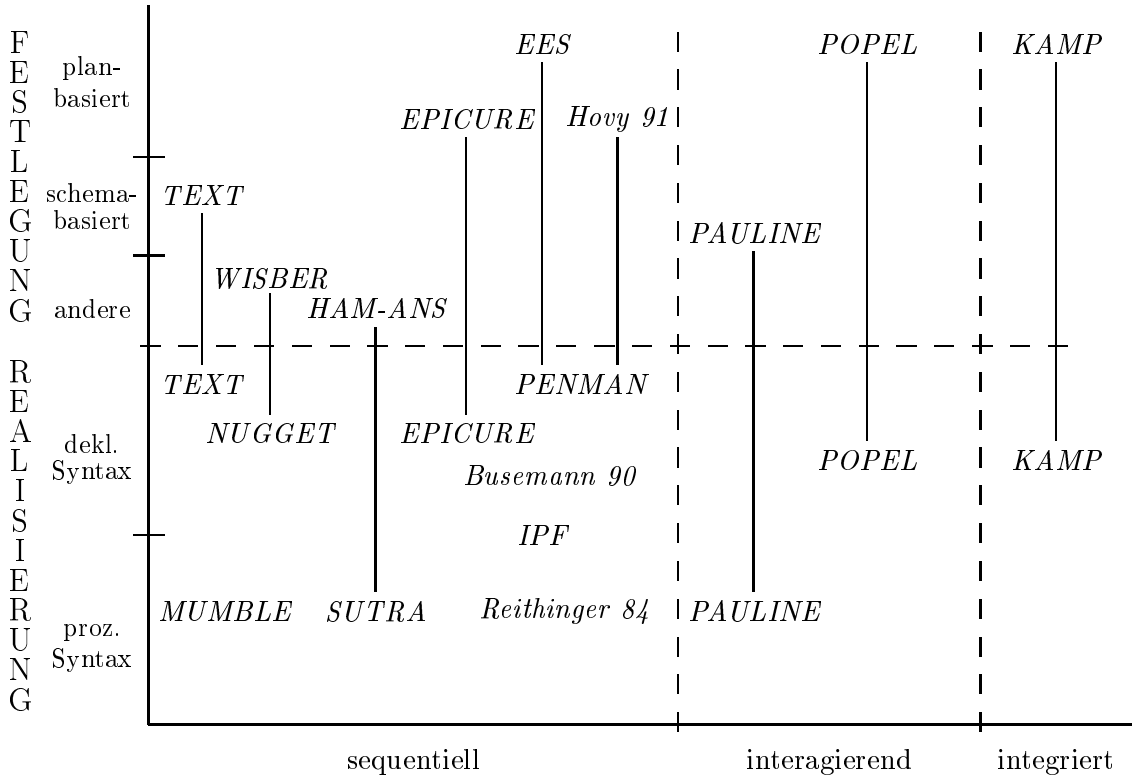


Abbildung 2.6: Eine Einordnung von Generierungssystemen anhand von Architekturmerkmalen

Entlang der x-Achse der Abbildung sind die drei Architekturansätze sequentiell, interagierend und integriert aufgetragen. Die y-Achse ist unterteilt in die beiden Bereiche der Festlegung und der Realisierung des Inhalts. Die Festlegung ist nochmals nach der Art des Wissens unterteilt, das bei der Auswahl des Inhalts verwendet wird, die Realisierung danach, ob das syntaktische Wissen deklarativ oder prozedural dargestellt ist. Vertikale Linien sind zwischen den Komponenten gezogen, die zusammen ein vollständiges Generierungssystem bilden.

Vollständig implementierte Systeme, die dem sequentiellen Architekturmodell folgen sind:

- das System TEXT (vgl. Abschnitt 2.3).

- WISBER ist ein natürlichsprachliches Konsultationssystem, in dem der inhaltsfestlegende Teil des Generators eng an das Gesamtsystem gekoppelt ist. In diesem Teil des Systems erfolgt die Transformation der zu verbalisierenden konzeptuellen Repräsentation in eine benutzerangepaßte Beschreibung, die Wahl der Referenzausdrücke und der Lexeme [Horacek, 1990]. Das Resultat ist eine Merkmalsstruktur, die an die in-

haltsrealisierende Komponente NUGGET übergeben wird [Jablonski *et al.*, 1990]. NUGGET ist ein unabhängiger, portabler Modul und besteht aus einer syntaktischen Komponente, die mit einer *Definite Clause Grammar* (DCG) arbeitet, und Komponenten zur Vereinfachung der syntaktischen Struktur und zur Flektion.

- EPICURE ist ein System, dessen Schwerpunkt bei der Generierung referentieller Ausdrücke liegt [Dale, 1988, Dale, 1990]. Die Domäne, in der es operiert, ist die Generierung von Kochrezepten. Um das Ziel der Eingabe erfüllen zu können, das in der Rezeptdomäne z.B. darin besteht, ein Rezept für Butterbohnen zu erzeugen, bestimmt ein hierarchischer Planer mit Hilfe von Planoperatoren eine Diskursspezifikation. Diese wird über eine semantische Zwischenrepräsentation in eine abstrakte syntaktische Struktur überführt. Eine DCG erzeugt daraus die Oberflächenstrukturen der Sätze, die als Resultat ausgegeben werden.

- PENMAN ist eine inhaltsrealisierende Komponente, die aus drei Teilen besteht: der systemischen Grammatik NIGEL, einem Basislexikon und dem sog. *Upper Model*, einer semantischen Konzepthierarchie [Bateman *et al.*, 1989]. PENMAN wird sowohl vom *Explainable Expert System* (EES) als auch vom System von Hovy zur Realisierung der Ausgabe verwendet (siehe die Abschnitte 3.2.3 und 5.1 sowie [Moore, 1989, Hovy, 1991]). Die beiden Komponenten verwenden operationalisierte Versionen der *Rhetorical Structure Theory*, um im Falle des EES die Reaktionen eines Expertensystems erklären zu können, bzw. um aus Teilen einer Wissensquelle einen kohärenten Text zu generieren.

- SUTRA ist eine prozedurale morphosyntaktische Realisierungskomponente für das Deutsche [Busemann, 1984]. Sie wurde im Rahmen des natürlichsprachlichen Dialogsystems HAM-ANS entwickelt, in dem auch die Inhaltsfestlegung durchgeführt wird [Hoeppner *et al.*, 1983]. Bei der Generierung von Nominalphrasen findet in diesem System eine Rückkopplung über die Analysekomponente statt, die mit der Theorie der editierenden Korrektur bei [Levelt, 1989, p.467ff] vergleichbar ist. Dabei wird eine Phrase, die erzeugt werden soll, analysiert, um zu sehen, ob sie für den Benutzer verständlich ist, und gegebenenfalls korrigiert [Jameson and Wahlster, 1982]. SUTRA wurde neben dem Einsatz in HAM-ANS auch in einigen anderen Systemen verwendet, teilweise in modifizierter Form [Rösner, 1986, Novak, 1987, Herzog *et al.*, 1989].

- MUMBLE ist eines der ersten Generierungssysteme, dessen verschiedene Versionen seit über einem Jahrzehnt entwickelt werden (siehe z.B. [McDonald, 1981, Meteer, 1990b]). Es kann als Front-End an Systeme gekoppelt werden, deren Ausgabe verbalisiert werden soll.

- Die in [Reithinger, 1984] beschriebene Komponente ist eine prozedurale Implementierung einer Dependenzgrammatik. Sie wurde im Rahmen eines Dialogsystems für gesprochene Sprache entwickelt.

- Eine deklarative Grammatik, nämlich eine generalisierte Phrasenstruktur-Grammatik, verwendet der syntaktische Generator aus [Busemann, 1990]. Dieses System wurde im Rahmen eines Systems zur automatischen Übersetzung eingesetzt.

- Zwischen einer deklarativen und prozeduralen Grammatikrepräsentation steht der *Incremental Parallel Formulator* (IPF) [DeSmedt, 1990, DeSmedt and Kempen, 1991]. Es existiert zwar eine formale Beschreibung der syntaktischen Strukturen, die Formulierung und Verarbeitung erfolgt aber mit einer speziellen Implementationssprache, die es nicht erlaubt, die Grammatik neutral bezüglich einer konkreten Applikation zu beschreiben. Dieses System ist als einziges in der Lage, Eingaben inkrementell und parallel zu verarbeiten.

Interagierende Systeme, die in der Abbildung zu sehen sind, sind PAULINE (vgl. Abschnitt 2.4) und das in dieser Arbeit vorgestellte System POPEL. Die Einordnung von POPEL zeigt bereits einige Merkmale des Systems, die in den folgenden Kapiteln festgelegt werden. Das einzige integrierte System ist KAMP (vgl. Abschnitt 2.5). In diesem System wurde auch zum ersten Mal die Verwendung von Gesten neben der Generierung von Sprache in die Konzeption mit einbezogen, jedoch nicht implementiert.

Kapitel 3

Das System POPEL – Anforderungen und Aufbau

Ziel dieses Kapitels ist es, den Aufbau und den Ablauf der Generierung innerhalb des Systems POPEL zu motivieren. Dazu werden zunächst die Anforderungen untersucht, die an das Generierungssystem gestellt werden. Durch die Integration in ein natürlichsprachliches Dialogsystem erhält die gemeinsame Verwendung der Wissensquellen durch die Analyse- und Generierungskomponenten besondere Bedeutung. In einem Exkurs soll gezeigt werden, ob und wie aus dieser Verwendung eine *Bidirektionalität* der Wissensquellen und Verarbeitungsverfahren abgeleitet werden kann. Auf der Grundlage der definierten Anforderungen wird dann das Modell und der Verarbeitungsablauf für POPEL vorgestellt.

3.1 Der Rahmen: Das System XTRA

Eine Randbedingung bei der Entwicklung von POPEL ist der Einsatz in einem natürlichsprachlichen Dialogsystem, dem System XTRA [Allgayer *et al.*, 1989a, Allgayer *et al.*, 1989b, Allgayer *et al.*, 1989c]. XTRA ist ein natürlichsprachliches Zugangssystem zu Expertensystemen, d.h. es soll einem Benutzer ermöglichen, mit einem Expertensystem in natürlicher Sprache zu kommunizieren. XTRA übernimmt in der Interaktion zwischen dem Benutzer und dem Expertensystem, wenn nötig, auch die Initiative und bietet damit eine Funktionalität, die über ein einfaches Übersetzen der natürlichsprachlichen Eingabe in eine für das Expertensystem verständliche Form und eine entsprechende Rückübersetzung der Ergebnisse weit hinaus geht. Dabei ist XTRA als transportables System konzipiert, das an verschiedene Expertensysteme anpaßbar ist. Abbildung **??** zeigt die Systemoberfläche mit einem kurzen Beispieldialog in der ersten Anwendung als Zugangssystem zu einem Expertensystem zur Lohnsteuerberechnung.

Aus der Anforderung, daß XTRA an verschiedene Expertensysteme und damit an unterschiedliche Domänen anpaßbar sein muß und jeweils eine möglichst natürliche Interaktion ermöglichen soll, ergaben sich folgende drei Entwurfsprinzipien, die unmittelbaren Einfluß auch auf den Generator haben:

Bidirektionalität: XTRA muß sich als Dialogpartner kooperativ verhalten und Sprache

kohärent verarbeiten. Deshalb müssen die Komponenten zur Analyse und Generierung von Sprache auf die von der jeweils anderen Komponente erzeugten internen Repräsentationen zugreifen und diese verarbeiten können, z.B. um referentielle Ausdrücke korrekt zu interpretieren beziehungsweise zu generieren. Damit muß es eine gemeinsame Repräsentationsbasis geben, die innerhalb des Systems verwendet wird. Zudem soll so weit wie möglich vermieden werden, daß es für eine Verarbeitungsrichtung spezielle Wissensquellen gibt, sobald in diesen solches Wissen repräsentiert wird, das auch für die jeweils andere Verarbeitungsrichtung von Bedeutung ist. Damit werden Redundanzen und Inkohärenzen im Bereich der Wissensquellen vermieden, und es entfallen Komponenten, die Wissen zwischen verschiedenen Repräsentationsformalismen transformieren müssen. Neben der leichteren Anpaßbarkeit der Wissensbasis an eine neue Domäne ergibt sich aus der bidirektionalen Verwendung auch ein theoretisch interessanter Aspekt: die Anforderung nach Bidirektionalität schließt weitgehend aus, daß sich analyse- oder generierungsspezifische Eigenheiten in den Repräsentationsformalismen und dem repräsentierten Wissen befinden. Die Repräsentationsbasis im System kann in Anspruch nehmen, unabhängig von der Verarbeitungsrichtung zu sein (siehe auch Abschnitt 3.3). xtra9.ps15cm[Die Benutzeroberfläche von XTRA][gammelxtraober]

Multimodalität: XTRA bietet neben der Interaktion mittels Sprache zusätzlich die Möglichkeit, Elemente eines domänenspezifischen visuellen Kontexts mit Zeigegesten zu lokalisieren. Wenn Kommunikationspartner durch Zeigegesten auf einen gemeinsamen visuellen Kontext Bezug nehmen können, z.B. auf die Felder eines Lohnsteuerformulars, erleichtert dies die Kommunikation und befreit von dem Zwang, eventuell komplexe oder dem Dialogpartner unbekannte sprachliche Beschreibungen verwenden zu müssen. Da sowohl die Analyse– als auch die Generierungskomponente die Fähigkeit besitzen muß, Zeigegesten zusammen mit Sprache zu verarbeiten, erweitert sich die Aufgabenspezifikation für POPEL um den Punkt der Generierung von Zeigegesten.

Modularität: In einem natürlichsprachlichen System wird sowohl Wissen über den Aufbau und die Verwendung von Sprache – in XTRA zusätzlich Wissen über Zeigegesten – benötigt, als auch Wissen über die Domäne, in der das System arbeitet. Die Anforderung an XTRA, als ein *transportables* System in verschiedenen Domänen eingesetzt werden zu können, legt es nahe, kommunikationsmittelbezogenes Wissen von Domänenwissen getrennt zu repräsentieren. Diese Trennung ermöglicht es, das System an eine neue Domäne leichter anzupassen, da nur das Domänenwissen, nicht aber das sprachliche Wissen und das Wissen über die prinzipielle Verwendung von Zeigegesten ausgetauscht werden muß. Der modulare Aufbau der Wissensquellen ermöglicht auch einen modularisierten Aufbau der Verarbeitungskomponenten. Damit ist es möglich, Einzelkomponenten auch in anderen Systemen einzusetzen. Auch POPEL soll so aufgebaut werden, daß das System nicht nur als Generierungssystem in XTRA eingesetzt werden kann, sondern auch als geschlossenes Einzelsystem in andere KI-Systeme integriert werden kann, falls diese die Wissensquellen bereitstellen, die POPEL benötigt.

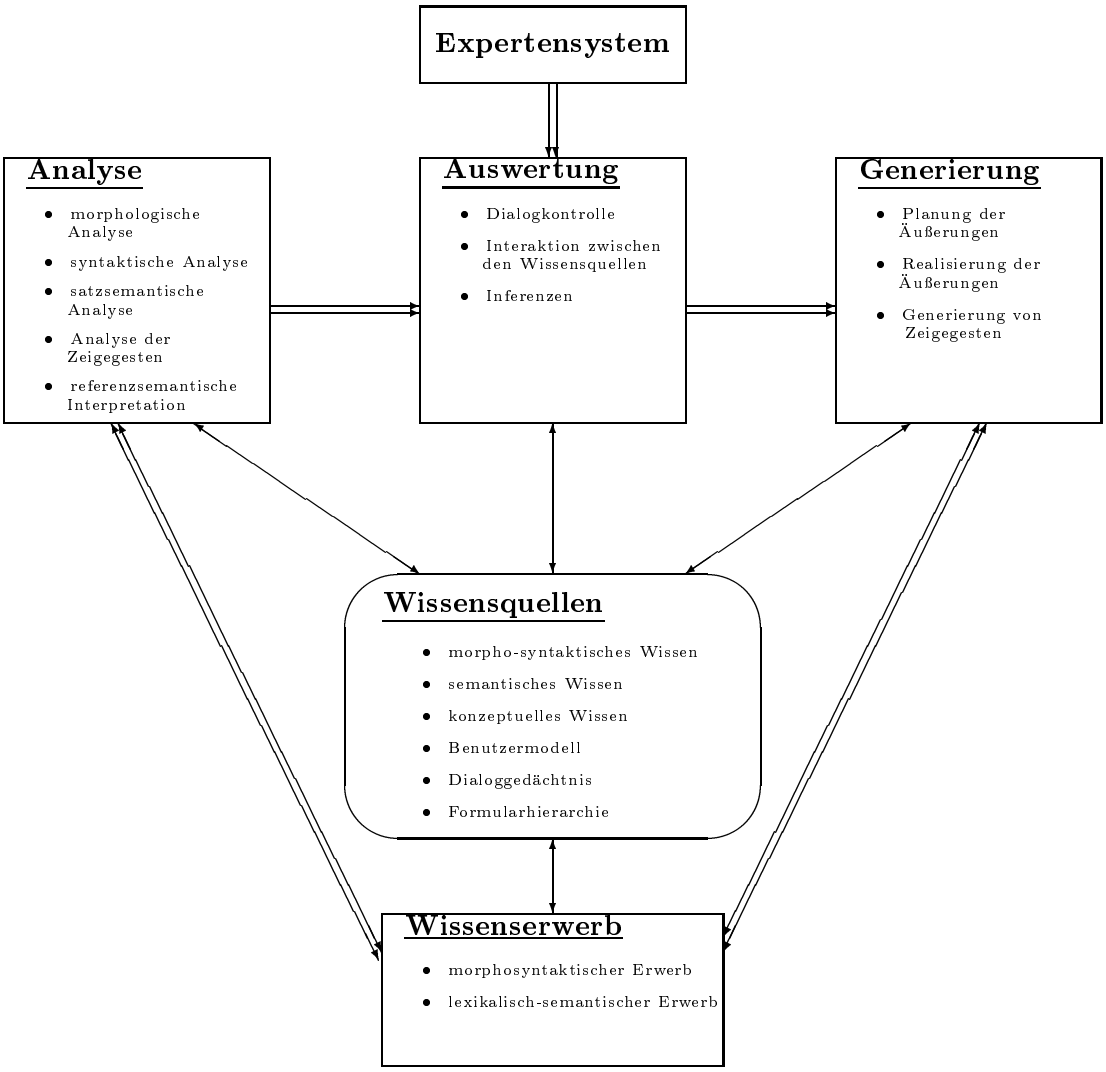


Abbildung 3.1: Die Architektur von XTRA

Abbildung 3.1 gibt einen Überblick über den Aufbau des gesamten Systems. Doppelpfeile zeigen den Kontrollfluß, einfache Pfeile den Fluß der Daten. Das System folgt der traditionellen Dreiteilung der Verarbeitung in die Phasen *Analyse*, *Auswertung* und *Generierung*. Zusätzlich verfügt XTRA über ein Modul zum *Wissenserwerb*.

Die Eingabe des Benutzers wird zunächst morphologisch und syntaktisch analysiert. Die Deflexion erfolgt mit dem Modul MORPHIX, das als lexikalische Wissensquelle das Lexikon XTRALEX verwendet [Finkler and Neumann, 1988]. Die syntaktische Analyse erfolgt mit dem Modul SB-PATR, einer modifizierten Version von D-PATR [Shieber *et al.*, 1983]. Danach wird die gewonnene syntaktische Struktur in eine domänenunabhängige satzsemantische Repräsentation, die *Functional Semantic Structure* (FSS), übersetzt. Die FSS repräsentiert domänenunabhängige wohlgeformte Prädikat/Argumentstrukturen der deutschen Sprache, die auf einer Kasusrahmendarstellung basieren (vgl. [von Polenz, 1985]).

Mit der Erzeugung einer FSS–Struktur ist die diskurskontextunabhängige, primär sprachorientierte Phase beendet.

Die referenzsemantische Interpretation überträgt die satzsemantische Repräsentation regelbasiert in das konzeptuelle Wissen des Systems, die *Conceptual Knowledge Base* (CKB). Dabei werden Referenzen aufgelöst, unter anderem durch die Auswertung des Dialoggedächtnisses und eventuell vorhandener Zeigegesten des Benutzers. Die Analyse der Zeigegesten wird vom Modul TACTILUS-II durchgeführt. Die geometrischen Beziehungen der Graphik, auf die gezeigt wird, werden in der *Formularhierarchie* repräsentiert [Allgayer, 1986, Wille, 1989].

In der anschließenden Auswertungsphase wird untersucht, welche Daten an das Expertensystem übergeben werden können, und wie auf die Eingabe des Benutzers, unter Berücksichtigung der vom Expertensystem erhaltenen Information, reagiert werden kann. Dabei wird die Kopplung zwischen der CKB und dem terminologischen Wissen des Expertensystems ausgenutzt. Die Kopplung wird so durchgeführt, daß eine redundante Repräsentation vermieden wird. Die Dialogkontrolle verwendet das Benutzermodell BGP-MS [Kobsa, 1990] und einen Dialogaktplaner [Reithinger, 1989].

In der derzeitigen Anwendung von XTRA ist das Expertensystem LST-1 angeschlossen, ein in der Expertensystem–Shell BABYLON implementiertes System zur Unterstützung beim Ausfüllen eines Lohnsteuerjahresausgleichformulars [Beiche, 1989].

Das Ergebnis des Auswertungsprozesses ist die Eingabe, die der Generator erhält und die den Ausgangspunkt des Verbalisierungsprozesses bildet. Dieser muß die Äußerungen des Systems festlegen und realisieren, u.U. unter Verwendung von Zeigegesten.

Die Komponente zum Wissenserwerb tritt dann in Aktion, wenn der Dialogpartner Wörter verwendet, die das System nicht kennt. Es ist derzeit in der Lage, morphosyntaktische und semantische Merkmale neuer Wörter in die Wissensbasis des Systems einzutragen. Bei der Verarbeitung verwendet das System den Generator des Systems, um durch die Generierung von Paraphrasen für neu erworbene Wörter dem Dialogpartner eine Kontrolle des Erwerbs zu ermöglichen [Jansen-Winkeln, 1988, Ndiaye, 1990, Jansen-Winkeln *et al.*, 1991]

3.2 Die Anforderungen an das Generierungssystem

Die im letzten Kapitel vorgestellten Systeme hatten ihren Schwerpunkt immer in einem speziellen Bereich der Generierung: TEXT auf der kohärenten Verbalisierung von Datenbankeinträgen, PAULINE auf der Berücksichtigung pragmatischer Phänomene und KAMP bei der Planung einer kommunikativen Aktion.

Typisch für die Verarbeitung in diesen Systemen ist, daß sie speziell und ausschließlich als Systeme entwickelt wurden, die jeden Generierungszyklus 'neu' durchlaufen, d.h. ohne einen Kontext zu berücksichtigen, der aus einem vorher durchgeführten Generatorlauf oder einer eventuellen Benutzereingabe stammen könnte.

Bei der Konzeption von POPEL ergeben sich die Rahmenbedingungen aus den Anforderungen, die das Einsatzgebiet festlegt. Im vorangehenden Abschnitt wurde das Dialogsystem beschrieben, in dem POPEL arbeiten soll. Die wichtigsten Faktoren, die bei dem

Entwurf berücksichtigt werden müssen, sind:

a) Integration Das System wird in einem Dialogsystem eingesetzt.

b) Flexibilität Flexibilität ist in zweierlei Hinsicht erforderlich:

- Die Domäne ist austauschbar.
- Der Sprachumfang hat die Bandbreite von kurzen Dialogbeiträgen bis zu erklärenden Texten.

c) Multimodalität Begleitend zur Ausgabe von Sprache sollen Zeigegesten auf eine Graphik visualisiert werden.

Auch müssen die Kriterien beachtet werden, die bei der Entwicklung des Systems XTRA zugrundegelegt wurden. Durch die Integration von POPEL in das Gesamtsystem stellt sich die Frage, inwieweit Analyse und Generierung *bidirektional* Wissensquellen und Verarbeitungsverfahren benutzen können. In Abschnitt 3.3 soll dieses Problem allgemeiner untersucht werden. Das Entwurfsziel *Multimodalität* findet sich direkt in den Anforderungen für POPEL wider. Erstmals wurde die Kopplung von Sprache und Gesten im System KAMP [Appelt, 1985, p.114ff] kurz erwähnt, aber nicht implementiert. Dort ist kommunikatives Zeigen eine mögliche Alternative bei der Realisierung einer *concept activation* Aktion. POPEL ist also das erste System, in dem ein Generierungssystem Sprechhandlungen und Zeigehandlungen kombiniert erzeugen soll.

Bleibt als dritte Forderung die der *Modularität*. Ein Generierungssystem in einem Dialogsystem ist als ein 'wissensintensiver' Modul [Jacobs, 1987] auf viele Wissensquellen angewiesen. Die Modularität hängt damit von der Parametrisierbarkeit dieser Wissensquellen ab. Im Laufe der Definition des Modells, dem POPEL folgt, wird sich zeigen, inwieweit das Gesamtsystem oder Teile davon modular aufgebaut werden können.

3.2.1 Integration in ein Dialogsystem

Die Anforderung, die den größten Einfluß für den Entwurf hat, ist zweifellos die Integration in ein Dialogsystem. Das System operiert in einem Gesamtkontext, in dem Wissensquellen nicht mehr nur vom Generator benutzt und erweitert werden. Durch die Ergebnisse der Analyse ändert sich die interne Struktur des Gesamtsystems in einer Weise, die außerhalb der Kontrolle des Generators steht. Er operiert also in einer sich dynamisch im Lauf des Dialogs verändernden Umgebung. Zudem ist es nicht mehr möglich, unabhängig von anderen Moduln Wissensquellen und Verarbeitungsmethoden so zu wählen, daß diese für die Generierung günstig sind. Die Restriktionen, die aus der Integration folgen, sind also:

- Wissensquellen müssen mit anderen Moduln geteilt werden.
- Kontextuelles Wissen muß in hohem Maß berücksichtigt werden.

Folgende Punkte werden in [Mann, 1987] als Wissensbereiche und Phänomene genannt, auf die sich Analyse und Generierung gemeinsam beziehen:

1. das Lexikon, mit morphologischer, syntaktischer und semantischer Information;
2. die Grammatik;
3. Diskursphänomene, wie Anaphernbildung, thematische Struktur und Fokussierung, und deren Repräsentation im System;
4. Situationsphänomene, wie Benutzermodellierung und die Domänenbeschreibung;

Im XTRA-System sind es genau diese Wissensquellen, die gemeinsam verwendet werden müssen. Einige Gemeinsamkeiten sollen an dem folgenden Beispieldialog demonstriert werden, der in Abbildung ?? (Seite ??) zu sehen ist:

BEN: *Ich fahre täglich von hier ↗ nach Völklingen.*[1]

SYS: *Was kostet die Fahrt?*

BEN: *Das kostet 35 DM im Monat.*

In der Analysephase wird der erste Satz syntaktisch und satzsemantisch analysiert und danach in das konzeptuelle Wissen abgebildet. Sind die Inferenzprozesse von XTRA und dem angeschlossenen Expertensystem beendet und erhält der Generator die Repräsentation des Ergebnisses, muß der gleiche Weg zurück durchlaufen werden. Das heißt, aus der konzeptuellen Repräsentation wird eine satzsemantische und syntaktische Struktur bestimmt, die schließlich zu der Frage an den Benutzer führt.

In den Dialogbeiträgen nehmen die Dialogpartner gegenseitig Bezug auf die Äußerungen des jeweils anderen. So benötigt der Generator für die Erzeugung der Frage Wissen darüber, daß ein *fahren* Ereignis im letzten Satz erwähnt wurde, auf das mit einer definiten Nominalphrase referiert werden kann. Für die Analyse der Antwort des Benutzers muß wiederum bekannt sein, worauf sich *das* beziehen kann. Um diese Entscheidung treffen zu können, muß das Dialogsystem eine gemeinsame Wissensquelle enthalten, die die Struktur des Dialogs enthält, z.B. welche Ausschnitte des Dialogs gerade fokussiert sind und welche Zusammenhänge es zwischen Teilen des konzeptuellen Wissens und den sprachlichen Äußerungen gibt. Dieses *Dialoggedächtnis* muß von beiden Verarbeitungskomponenten benutzt und erweitert werden, damit die Referenzen aufgelöst und generiert werden können.

Neben der gemeinsamen Verwendung der Repräsentationen der verschiedenen Darstellungsebenen ist auch zu überlegen, inwieweit das Transformationswissen zwischen den Ebenen von Analyse und Generierung gemeinsam benutzt werden kann, z.B. das Lexikon als eine Wissensquelle für den Übergang zwischen der syntaktischen und der satzsemantischen Ebene.

Außer dem Wissen über den Dialogkontext, der sich unmittelbar aus den sprachlichen Aktionen der Dialogpartner ergibt, ist in einem Dialogsystem eine Wissensquelle notwendig, die Annahmen über das Wissen und die Ziele der Dialogpartner enthält. Dieses *Benutzermodell* muß im obigen Beispiel ableiten können, daß der Benutzer durch die Angabe

[1] Mit BEN beginnende Zeilen zeigen Eingaben des Benutzers. Die mit SYS beginnenden zeigen die Reaktion des Systems darauf, die in dieser Form von POPEL generiert werden. ↗ steht im Text als Zeichen für eine die zugeordnete Phrase begleitende Zeigegeste.

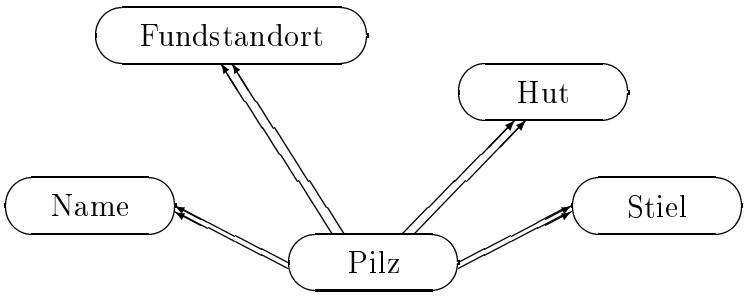


Abbildung 3.2: Ausschnitt aus der terminologischen Hierarchie eines Expertensystems

seiner täglichen Fahrtstrecke – im Kontext der Domäne "Lohnsteuerjahresausgleich" – wissen möchte, inwieweit die dadurch entstehenden Kosten von der Lohnsteuer absetzbar sind. Dieses Benutzerziel muß sich das System zu eigen machen und aus dem Systemziel – Berechnung der absetzbaren Kosten – folgern können, daß Wissen über die Kosten der Fahrt fehlt, und daß es vom Benutzer angefordert werden kann. Außer dieser Repräsentation von Wissen und Zielen des aktuellen Benutzers kann das Benutzermodell Stereotype enthalten, die Standardannahmen über das Wissen bestimmter Benutzerklassen enthalten. Ist der Benutzer beispielsweise als Laie klassifiziert, kann die Erklärungsstrategie des Generators sich von der eines Experten unterscheiden.

3.2.2 Flexibilität

3.2.2.1 Austauschbares konzeptuelles Wissen

Ziel von XTRA ist es, den natürlichsprachlichen Zugang zu jeweils einem Expertensystem zu ermöglichen, das ausgetauscht werden kann. Die Verbindung zwischen XTRA und dem Expertensystem erfolgt über das konzeptuelle Wissen von XTRA. Das terminologische Wissen des Expertensystems wird dabei in das konzeptuelle Wissen von XTRA integriert. Diese Übertragung erfolgt rechnergestützt und ist so konzipiert, daß eine Verdoppelung der Wissensquellen nicht erfolgt [Bolz and Weber, 1990]. Diese Art der Kopplung ist nur dann durchführbar, wenn das Expertensystem eine explizite terminologische Wissensquelle besitzt. Der natürlichsprachliche Zugang zu einem rein regelbasierten System ist damit nicht möglich (siehe auch Abschnitt 3.2.3).

Für die sprachverarbeitenden Teile von XTRA ergibt sich aus der damit folgenden domänenspezifischen Struktur des konzeptuellen Wissens das Problem, daß es naturgemäß nicht dazu vorgesehen ist, für eine eventuelle 'Versprachlichung' verwendet zu werden. Außer der Tatsache, daß das Wissen in einem bestimmten Repräsentationsformalismus vorliegt, können keine weiteren Annahmen gemacht werden. Das heißt zum Beispiel, daß die Generalisierungshierarchie durchaus nicht so aufgebaut ist, wie man es von einer 'klassischen' terminologischen Struktur erwarten könnte. Abbildung 3.2 zeigt einen (vereinfachten) Ausschnitt der Generalisierungshierarchie eines funktionierenden Expertensystems, nämlich eines Systems zur Bestimmung von Pilzen [Müller, 1986]. Die Ovale stehen für Konzepte, die Doppelpfeile bezeichen die *is-a* Relationen. Offensichtlich strukturierte

der Wissensingenieur das Domänenwissen so, wie es unter dem Verarbeitungsaspekt am günstigsten erschienen ist.

Wollte man für diese Wissensquelle das System TEXT zur Verbalisierung einsetzen und die Anfrage `(definition pilz)` an das System stellen, so könnte dieses bei der Abarbeitung des *information* Schemas, wenn es mit dem rhetorischen Prädikat *class* die Generalisierungshierarchie nach oben verbalisiert, den Pilz etwa so beschreiben:

Ein Pilz ist ein Name, ein Hut, ein Stiel und ein Fundstandort.

Diese 'Definition' widerspricht vermutlich dem Vorwissen und den Erwartungen des Benutzers. Um für Erklärungen verwendbar zu sein, müßte die Hierarchie z.B. nach botanischen Kriterien aufgebaut sein. In dem Beispiel sind Beziehungen als Superkonzeptbeziehungen dargestellt, die 'natürlich' als Attribute mit Rollenbeziehungen ausgedrückt würden.

Solche aus der Sicht der Sprachverarbeitung kaum verwendbaren Wissensquellen können nicht direkt als Ausgangs- oder Zielstrukturen des natürlichsprachlichen Zugangs verwendet werden. Deshalb muß die aus der Wissensquelle des Expertensystems gewonnene Wissensquelle unter Verwendung der im System XTRA verwendeten Repräsentations- und Entwicklungswerkzeuge so verändert und erweitert werden, daß zumindest eine partiell nutzbare Wissensquelle zur Verfügung steht [Allgayer *et al.*, 1989b, p.27ff.].

Prinzipiell stellen sich bei der engen Kopplung eine Reihe von Fragen, z.B.: welche Teile einer Wissensquelle können überhaupt in Sprache umgesetzt werden, wie bestimmt man, ob ein repräsentiertes Konzept mit einem Verb oder einem Nomen verbalisiert werden kann, oder welche Teile zusammen als Satz verbalisiert werden können? Eine Lösung dieser Probleme ist nur dann möglich, wenn domänenunabhängiges, sprachorientiertes Wissen, d.h. eine satzsemantische Wissensquelle, bei der Auswahl unterstützend hinzugezogen wird. Diese Interaktion bei der Auswahl spielt beim Entwurf des Generators eine wichtige Rolle.

3.2.2.2 Große sprachliche Bandbreite

Der Generator operiert innerhalb des Systems XTRA in einer Einsatzsituation, in der folgende drei Typen von Reaktionen auftreten können:

1. Fragen, deren Antwort das Expertensystem wissen muß, um weiterarbeiten zu können,

2. Ergebnisse des Expertensystems, die dem Benutzer mitgeteilt werden sollen und

3. Erklärungen des konzeptuellen Wissens.

Die Bandbreite der sprachlichen Ausdrucksmittel, die der Generator beherrschen muß, reicht von dialogtypischen Ellipsen wie

BEN: *Ich fahre täglich von Saarbrücken nach Völklingen zur Arbeit.*

SYS: *Mit dem Bus?*

über einzelne Sätze wie

SYS: *Was kostet die Fahrt?*

bis hin zu Texten wie

SYS: *Steuerhandlungen werden von Personen ausgeführt. Sie verursachen Kosten, die abgesetzt werden können. Die Steuerhandlungen können wiederholt werden.*

die eine Verbalisierung eines Teilausschnitts des terminologischen Wissens sind.

Hier spielen wieder die Kontextwissensquellen eine große Rolle. Die Ellipse kann nur erzeugt werden im Kontext der vorangegangenen Frage und der Text nur dann, wenn aus dem Benutzermodell ableitbar ist, daß der Dialogpartner das Konzept 'steuerlich absetzbare Handlung' nicht kennt. Eine Alternative zu dem Text wäre

SYS: *Diese Fahrt ist eine Steuerhandlung.*

für den Fall, daß die Fahrt zum Arbeitsplatz bekannt ist und das System annimmt, daß an einer genauen Definition kein Interesse besteht.

Um diese Bandbreite an Reaktionen erzeugen zu können, benötigt das Generierungssystem ein flexibles Verfahren zur Bestimmung derjenigen Teile des konzeptuellen Wissens, die in der jeweilig gegebenen Dialogsituation verbalisiert werden sollen, und Wissen darüber, wie diese Teile in sprachliche Strukturen überführt werden sollen. Das Ziel muß es sein, eine Ausgabe zu erreichen, die in der jeweiligen Dialogsituation dem Dialogpartner am verständlichsten ist.

3.2.3 Exkurs: Erklärungen in Expertensystemen

Im letzten Abschnitt wurde die Bandbreite möglicher Ausgaben des Generators umrissen. Das Gebiet, in dem XTRA operiert, erfordert es, genauer zu untersuchen, welche Informationen, die im Expertensystem vorliegen, überhaupt verbalisiert werden können. Die drei Äußerungsarten, die der Generator zu erzeugen in der Lage sein sollte, lassen sich mehr oder weniger direkt auf bestimmte Zustände des Expertensystems abbilden:

- Fragen: es werden Daten benötigt, die vom Benutzer kommen müssen.
- Ergebnisausgaben: eine Lösung wurde gefunden und wird verbalisiert.
- Erklärungen: eine Motivation des Systemverhaltens oder Teile der Wissensbasis sollen an den Benutzer ausgegeben werden.

Besonders der letzte Punkt, die Erklärung des Systemverhaltens und der gefundenen Lösungen, stellt ein Problem dar, sowohl für das Expertensystem als auch für den Generator, der eine natürlichsprachliche Ausgabe erzeugen soll. Fragt ein Benutzer nach dem 'warum' einer bestimmten Lösung, wird ihn die Ausgabe der vom Expertensystem verwendeten Regeln wohl kaum befriedigen. Bei der Entwicklung vieler Expertensysteme wurde zunächst darauf geachtet, daß sie überhaupt in der Lage waren, Lösungen für die

Probleme ihrer Domäne zu liefern. Hingegen sind sie nicht in der Lage, dem Benutzer plausibel zu machen, warum die Anwendung einer bestimmten Regel in diesem Kontext sinnvoll ist [Swartout, 1983].

[Moore, 1989] nennt zusammenfassend als Gründe hierfür, daß diesen Systemen Wissen zur Rechtfertigung ihrer Aktionen, Wissen über allgemeine Problemlösungsstrategien und terminologisches Wissen gefehlt hat. Das Wissen, das der Wissensingenieur im Expertensystem kodiert hat, ist in diesen Systemen nur *implizit* vorhanden und damit für eine Erklärung nicht zugänglich. Das Erklärungswissen muß also zusammen mit dem Problemlösungswissen aufgebaut werden. Sie sind nicht automatisch identisch [Wick, 1989].

Ein Beispiel für ein Expertensystem, das erklären kann, was es warum getan hat, ist das *Explainable Expert System* (EES) [Moore, 1989, p.83ff]. Es ist so aufgebaut, daß ein angeschlossenes Generierungssystem über das Wissen verfügen kann, das es benötigt, um zu erklären, warum das Expertensystem zu einer bestimmten Lösung gelangt ist. Als Wissensquellen enthält das EES ein Modell der Domäne, Problemlösungsstrategien aus der Domäne, Konfliktauflösungsstrategien und domänenübergreifendes terminologisches Wissen. Das Expertensystem selbst wird erzeugt, indem aus diesen Wissensquellen der ablauffähige Code kompiliert wird. In einer Struktur wird die Entwicklungsgeschichte der Entscheidungen festgehalten, die während der Erzeugung des Codes getroffen wurden. Alle diese Wissensquellen und zusätzlich ein Trace, der während des Ablaufs des Expertensystems angelegt wird, stehen dann für Erklärungen zur Verfügung. Das Erklärungssystem ist besonders dazu konzipiert, Folgefragen zu beantworten, da Erklärungen als interaktiver Prozeß zwischen dem System und dem Benutzer angesehen werden.

Betrachtet man die Ankopplung von XTRA an das Expertensystem, ist unmittelbar klar, daß XTRA, und damit der Generator, nur das terminologische Wissen des Expertensystems erklären kann. Inferenzen könnten nur dann erklärt werden, wenn diese explizit in der terminologischen Wissensquelle repräsentiert sind, zum Beispiel jede Regel als ein Frame, der einen Slot für die Vorbedingung und einen für die Aktion der Regel besitzt.

Im System LST-1 [Beiche, 1989] z.B., das Inferenzen teilweise mittels Dämonen durchführt, die bei Zugriff auf Attributwerte aktiviert werden, und teilweise mit Regeln, kann dem Benutzer somit nicht erklärt werden, warum das System zu einem bestimmten Ergebnis kommt.

Das konzeptuellen Wissen von XTRA ist mit dem des Expertensystems verzahnt. In [Wick, 1989] werden die Vor– und Nachteile einer engen bzw. losen Kopplung des Erklärungswissens und des Expertensystemwissens aufgeführt. Wenn eine zu lose Kopplung vorhanden ist, besteht die Gefahr, daß Erklärungswissen und Expertensystemwissen zu wenig miteinander zu tun haben und dem Benutzer eine Erklärung gegeben wird, die mit der Lösung, wie sie das Expertensystem berechnet hat, nichts mehr zu tun hat. Bei einer engen Kopplung besteht, wie in Abschnitt 3.2.2.1 gezeigt, die Gefahr, daß überhaupt keine befriedigende Erklärung geliefert werden kann.

In XTRA wird ein Mittelweg zwischen den beiden Extremen angestrebt. Zwar wird das terminologische Wissen des Systems übernommen, es wird aber während des Übernahmeprozesses dafür gesorgt, daß die resultierende konzeptuelle Wissensquelle erklärbar ist.

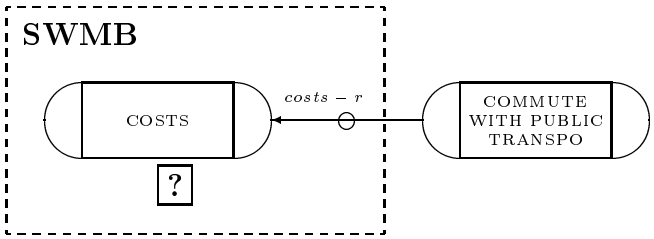


Abbildung 3.3: Ein Beispiel für den Inhalt von SWMB

3.2.4 Multimodalität

Menschliche Dialogpartner tauschen bei räumlicher Kopräsenz zwei Arten von Signalen aus: Sprache und kommunikative Körperbewegungen. Wichtigstes Beispiel dieser *Multimodalität* sind deiktische Akte, die die Elemente eines gemeinsamen visuellen Kontexts durch die Kombination von deiktischen Ausdrücken, wie z.B. "das da", "hier" usw., und außersprachlichen Referenzmitteln, wie z.B. Zeigegesten auf einen gemeinsamen visuellen Kontext, identifizieren [Schmauks and Reithinger, 1988].

Für die Ausgabeseite bietet der Einsatz von Zeigegesten besonders in der Kommunikation mit Expertensystemen den Vorteil, daß ein Objekt nicht mehr ausschließlich sprachlich identifiziert werden muß, wenn eine Referenz auch mittels einer Zeigegeste durchgeführt werden kann. Dadurch können besonders für Laien Begriffe aus unbekannten Domänen, die diesen unverständlich sind und erst umständlich erklärt werden müßten, durch einfach verständliche Gesten ersetzt werden.

Der visuelle Kontext von XTRA ist eine auf einem Bildschirm dargestellte Graphik aus der Expertensystemdomäne, z.B. ein Lohnsteuerformular, auf das der Dialopartner zeigen kann. Auf der Analyseseite steht mit dem System TACTILUS-II [Wille, 1989] eine Komponente zur Verfügung, die Zeigegesten des Benutzers analysieren kann. Die Wissensquelle, auf der die Verarbeitung der Gesten beruht, ist die *Formularhierarchie*, die den inhaltlichen und räumlichen Aufbau der Graphik repräsentiert. Die Kopplung mit dem konzeptuellen Wissen erfolgt mittels eines Systems, das die Korrespondenzbeziehungen zwischen Teilen der Graphik und Konzepten herstellt.

Die theoretischen und praktischen Grundlagen, auf denen die Generierung von Zeigegesten basiert, werden in Kapitel 7 dargestellt.

3.2.5 Die Eingabe

In Abschnitt 3.2.1 wurde gezeigt, daß viele verschiedene Wissensquellen des Gesamtsystems vom Generator benutzt werden müssen, um eine kommunikativ adäquate Ausgabe erzeugen zu können. Die Frage ist, ob die Eingabe, dem Postulat von Ward (siehe Abschnitt 2.1.1 und [Ward, 1989]) gemäß, der ganze interne Zustand des XTRA-Systems ist, aus dem sich der Generator 'ausdenkt', was gesagt werden soll, oder ob der Generator dieses Wissen zwar bei der Generierung berücksichtigen muß, die Eingabe aber enger gefaßt werden kann.

Die Ziele des Benutzers und des Systems werden in dem *Benutzermodell* gespeichert. Der Generator tritt dann in Aktion, wenn eines der Systemziele ist, daß über gewisse Teile des konzeptuellen Wissens eine gemeinsame Überzeugung herrschen sollte. Im engeren Sinn sind diejenigen Teile die Ausgangsstrukturen für POPEL, die zu dem Ziel *system wants that mutual belief exists*, kurz *SWMB*, gehören. Es werden keine weiteren Annahmen darüber gemacht, ob das System glaubt, daß der Dialopartner die entsprechenden Inhalte noch nicht kennt, oder ob sie nochmals verbalisiert werden sollen. Die Aufgabe des Generators ist es, dieses Ziel so in einen Dialogbeitrag umzusetzen, daß dieser sich kohärent in den Dialog einfügt, dem Benutzerwissen und den Systemabsichten angepaßt ist, und das Ziel erreicht wird.

Ein Beispiel für die Eingabestruktur in den Generator ist in Abbildung 3.3[2] zu sehen. Das Ziel ist, daß der Füller der *costs-r*–Rolle, der dem System nicht bekannt ist und hier mit ? markiert ist, dem System vom Dialogpartner mitgeteilt werden soll. Das System möchte erreichen, daß beide Dialogpartner den Füller dieser Rolle kennen. Da das System diesen selbst nicht kennt, kann der Generator als Dialogbeitrag beispielsweise die Frage erzeugen

SYS: *Was kostet die Fahrt?*

3.3 Bidirektionale Verarbeitung in Dialogsystemen

3.3.1 Ein kurzer Überblick über andere Arbeiten

Bei dem Entwurf eines vollständigen natürlichsprachlichen Systems stellt sich unmittelbar die Frage, ob und wenn ja wie Generierung und Analyse gemeinsam Wissensquellen und Verarbeitungsverfahren verwenden können. Bei der gemeinsamen Verwendung zumindest der Wissensquellen verfügt das System über eine redundanzfreie Darstellung seiner Kompetenz, die für Analyse und Generierung identisch wäre.

Unterschiede sieht McDonald [McDonald, 1987, p.200] vor allem in den Verfahren, die dieses Wissen verarbeiten. So könnten beide Verarbeitungsrichtungen zwar das gleiche sprachliche Wissen verwenden, sogar in der identischen Repräsentation, wenn diese deklarativ ist. Der Hauptunterschied ist jedoch die Verarbeitung. Die Generierung muß Entscheidungen treffen, während die Analyse Hypothesen auswerten muß. Sie muß mit mehrdeutigen und unterspezifizierten Daten arbeiten, während der Generator das gesamte System– und Sprachwissen zur Verfügung hat und damit die passende Äußerung bestimmen muß.

Die strikte Trennung von Informationsdarstellung und Informationsverarbeitung sowie eine deklarative Wissensdarstellung nennt auch [Neumann, 1991a]. Er unterscheidet verschiedene Grade der Bidirektionalität:

1. Aus einer gemeinsamen Wissensbasis werden verarbeitungsspezifische Darstellungen berechnet.

[2] Auf die bildliche Darstellung der Elemente der Wissensrepräsentationssprache SB-ONE wird in Abschnitt 4.1 näher eingegangen. Die Ovale in der Abbildung bezeichnen Instanzen, der Pfeil mit dem Kreis eine Rollenbeziehung.

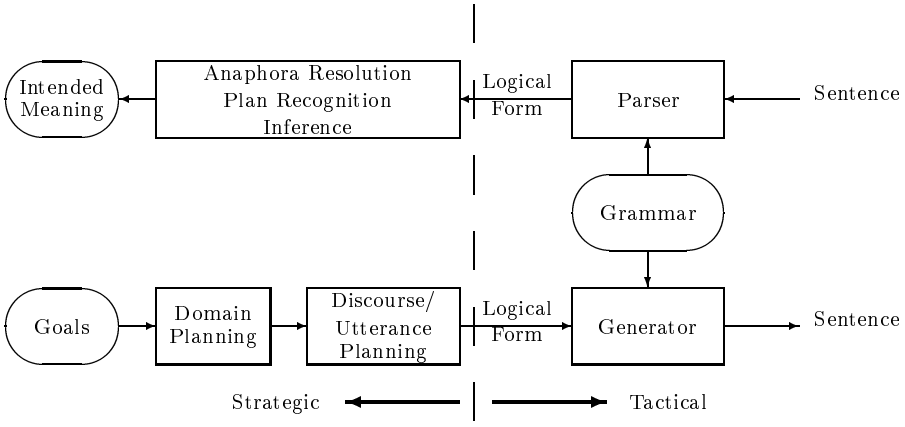


Abbildung 3.4: Die Organisation eines bidirektionalen Systems nach Appelt

2. Die Verarbeitungsprozesse verwenden eine identische Wissensbasis, aber unterschiedliche Verfahren.

3. Eine gemeinsame Wissensbasis und uniforme Basisoperationen werden von unterschiedlichen Prozessen verwendet.

4. Die Wissensbasis und der je nach Richtung parametrisierbare Verarbeitungsprozeß sind identisch.

Eine Architektur für ein System mit einer bidirektional verwendbaren Grammatik stellt [Appelt, 1987] vor (siehe Abbildung 3.4). Gemäß der oben genannten Klassifikation entspricht das System den in Punkt 3 genannten Kriterien, da neben der vollständig deklarativen Grammatik auch die Basisprozesse zur Verarbeitung identisch sein müssen.[3] Unifikationsbasierte Grammatiken bieten sich für diese Architektur an, da die Unifikation von Merkmalsstrukturen richtungsinvariant ist.

Basierend auf der Architektur schlägt [Shieber, 1988] ein syntaktisches Verarbeitungssystem vor, das gemäß Punkt 4 vollständig bidirektional ist, und Deduktion als einheitliches Verarbeitungsverfahren benutzt. Auf der Grundlage der Deduktion soll für das Deutsche im Projekt BILD [Neumann, 1991a] ein System entwickelt werden, das bidirektional lexikalisches und grammatisches Wissen verarbeitet und das in beiden Verarbeitungsrichtungen gleichermaßen effizient ist.

Die Bidirektionalität erstreckt sich in der Architektur von Appelt jedoch nur auf die syntaktische Verarbeitung, ein Gebiet, für das es gut ausgearbeitete, formalisierbare Ansätze gibt, z.B. die Unifikationsgrammatiken (vgl. z.B. [Dymetman *et al.*, 1990] und [Neumann, 1991a] für eine Übersicht). Gemeinsamer Ausgangs– und Endpunkt ist eine logische Form, nach der sich die Verarbeitungsprozesse wieder aufspalten. Damit ist aber wieder eine strenge Trennung der inhaltsfestlegenden von der inhaltsrealisierenden

[3] Appelt nennt hier nicht explizit die morphologische Verarbeitung, für die es aber ebenfalls ein bidirektional verwendbares Verfahren gibt [Koskenniemi, 1984].

Komponente gegeben und es treten, neben den Problemen, die sich aus der Verwendung einer logikbasierten Struktur ergeben,[4] alle diejenigen Probleme auf, die in Abschnitt 2.2 genannt wurden.

Wenn man diese Probleme beiseite läßt, bleibt die Frage, ob oberhalb der syntaktischen Verarbeitung sich die Prozesse von Analyse und Generierung so vollkommen unterscheiden, daß keine bidirektionale Verarbeitung stattfinden kann. Auch die Unterschiede, die in [Mann, 1987] genannt werden, befinden sich im Bereich der Inhalts– und Diskursplanung. Es sei zwar möglich, zu jedem Problem der einen Seite ein korrespondierendes auf der anderen zu finden, Mann schließt seine Überlegungen jedoch mit der Bemerkung [Mann, 1987, p.209]

> "Instead, the lists of problems being adressed by generation and understanding research differ substantially, and will remain different for a long time to come. This is because *the problems that limit the achievable quality of performance, the problems that pace progress, differ strongly between generation and understanding.*[5]"

3.3.2 Bidirektionalität in XTRA

In Abschnitt 3.2.1 wurde die Bidirektionalität in XTRA bereits angeschnitten. Dort wurde herausgestellt, daß vor allem die Repräsentation des satzsemantischen und konzeptuellen Wissens, sowie die kontextuellen Wissensquellen gemeinsam verwendet werden. Unter diesem Aspekt betrachtet, entspricht diese Verwendung Punkt 2 der Liste von Seite 42. Die Bidirektionalität, von der oben gesprochen wurde, bezieht sich jedoch nicht nur auf die Repräsentationsebenen, sondern auch auf das Transformationswissen und die Transformationsverfahren zwischen den Ebenen. In XTRA existieren zwei Transformationsschritte: zwischen der textuellen Ein/Ausgabe und der satzsemantischen Struktur, und zwischen der satzsemantischen Struktur und dem konzeptuellen Wissen.

Für den ersten Transformationsschritt werden morphologisches, grammatisches und lexikalisches Wissen und Verarbeitungmethoden für dieses Wissen benötigt. Die morphologische Verarbeitung stellt kein Problem dar, da das Morphologiepaket MORPHIX gemäß Punkt zwei der Klassifikation von Neumann aufgebaut ist und bidirektional verwendet werden kann. Damit ist auch das morphologische Lexikon nur einmal im System vorhanden. Der Formalismus der unifikationsbasierten syntaktischen Komponente SB-PATR erfüllt zwar die Voraussetzungen, die Appelt für eine bidirektionale Verwendung genannt hat, jedoch ist der in SB-PATR integrierte Chart-Parser speziell für die Analyse ausgelegt, ebenso wie die Grammatik XTRAGRAM. Es bietet sich aber an, den Unifikationsmodul und den Regelformalismus für die Generierung heranzuziehen. Die syntaktische Verarbeitung ist damit teilweise gemäß Punkt drei bidirektional. Die Methoden der Analysekom-

[4] Hier tritt das Problem der logischen Formäquivalenz auf, z.B. der Kommutativität von Operationen. Außerdem kann man nach Belieben Disjunktionen des Typs $\tau \vee \neg\tau$ für alle möglichen Terme τ hinzufügen, ohne daß sich der Wahrheitwert ändert [Appelt, 1987]. Das letztere Problem kann gelöst werden, wenn darauf geachtet wird, daß mit den hinzugefügten Disjunktionen keine zusätzlichen Redeobjekte eingeführt werden [Busemann, 1990, p.13].

[5] Hervorhebung des Autors.

ponente zur Erzeugung von satzsemantischen Strukturen aus der Konstituentenstruktur, die mit XTRAGRAM erzeugt wird, sind speziell auf diese Grammatik zugeschnitten. Allerdings ist das semantische Lexikon, das hierbei das Transformationswissen liefert, so strukturiert, daß es anwendungsneutral aufgebaut ist. Die von der Generierungs– bzw. Analysekomponente benötigte Information wird jeweils aus der gemeinsamen Wissensquelle berechnet. Der Grad der Bidirektionalität entspricht hier dem ersten Punkt der Klassifikation.

Der zweite Transformationsschritt, der Übergang zwischen der satzsemantischen Repräsentation und dem konzeptuellen Wissen, erfolgt regelbasiert, indem Teilstrukturen von einer Ebene in die andere transformiert werden. Diese Regeln können bidirektional verarbeitet werden und ensprechen dem zweiten Punkt der Klassifikation.[6]

Die Schritte vor bzw. nach dieser Transformation sind für Generierung und Analyse unterschiedlich. Während die Generierungskomponente den konzeptuellen Inhalt auswählen muß, ist es die Aufgabe der Analysekomponente, das in der zweiten Transformation erhaltene konzeptuelle Wissen mit dem bereits vorhanden in Beziehung zu setzen. Auch wenn die Verarbeitungsverfahren sich in diesen Schritten unterscheiden, werden doch die kontextuellen Wissensquellen, nämlich das Dialoggedächtnis und das Benutzermodell, gleichermaßen benutzt.

3.4 Die Architektur von POPEL

3.4.1 Systemarchitekturen wissensbasierter Systeme

In Abschnitt 2.2 wurden drei verschiedene Architekturmodelle für Generierungssysteme genannt: sequentiell, interagierend und integriert. Die beiden ersten Modelle gehen von einer Trennung in einen inhaltsfestlegenden und einen inhaltsrealisierenden Teil aus, wobei über den internen Aufbau der einzelnen Module nichts ausgesagt wird. Eine Verfeinerung der internen Struktur der zwei Hauptteile ist jedoch nur dann sinnvoll, wenn überhaupt Komponenten identifiziert werden können, die spezielle, von anderen Komponenten getrennte Aufgaben übernehmen. Im System TEXT sind dies zum Beispiel im inhaltsfestlegenden Teil die Komponente, die das *Schema* und den *Relevant Knowledge Pool* auswählt, und diejenige, die dann das Schema abarbeitet.

Bei der Frage, wie ein Generierungssystem aufgebaut sein kann, ist es hilfreich, allgemeiner die Strukturen von wissensbasierten Systemen zu betrachten. Systeme der Künstlichen Intelligenz setzen sich aus mehreren, oft voneinander unabhängigen Komponenten zusammen, die für eine spezielle Verarbeitungsebene über Problemlösungswissen und –verfahren verfügen. Speziell für natürlichsprachliche Systeme nennt [Görz, 1988] eine *pragmatische Ordung* zwischen den einzelnen Ebenen, die darin besteht, daß die Verarbeitung in einer tieferliegenden Ebene erst dann durchgeführt werden kann, wenn Ergebnisse der Verarbeitung aus einer darüberliegenden Ebene verfügbar sind. In einem Generator kann

[6] Die in [Reinert, 1990] vorgeschlagenen Regeln können in die Struktur transformiert werden, die der in POPEL entwickelten vereinfachten Vorläuferversion entspricht.

beispielsweise eine semantische Struktur erst dann erzeugt werden, wenn in einer darüberliegenden Ebene die konzeptuelle Repräsentation bestimmt wurde.

Die verschiedenen Architekturmodelle für wissensbasierte Systeme, die sich in der Literatur finden (z.B. [Wahlster, 1982, Görz, 1988] und in einer Zusammenfassung in [Bosch, 1988]) legen nicht fest, welche Komponenten existieren, sondern betrachten die Anordnung und die Verbindungen zwischen den Komponenten:

Das Phasenmodell: Die einzelnen Komponenten des Systems sind unidirektional sequentiell hintereinandergeschaltet. Eine Komponente wird erst dann aktiviert, wenn ihr Vorgänger die Berechnung vollständig abgeschlossen hat. In diesem Modell ist also immer nur eine Komponente aktiv. Das sequentielle Generierungsmodell ist eine Ausprägung dieses Modells.

Das Kaskadenmodell: Die einzelnen Komponenten sind, wie im Phasenmodell, sequentiell angeordnet. Außer der ersten Komponente, die keinen Vorgänger, und der letzten, die keinen Nachfolger besitzt, sind alle Komponenten mit genau einer Vorgänger- und einer Nachfolgerkomponente verbunden. In diesem Modell ist eine parallele Verarbeitung möglich: wenn in einer Komponente ein Teilergebnis berechnet wurde, kann damit in der nächsten Komponente weitergearbeitet werden und es können explizite Anforderungen für die weitere Verarbeitung zurückgegeben werden. Das Generierungsmodell mit Rückwirkungen kann mit dieser Architektur implementiert werden.

Das hierarchische Modell: Die Komponenten entsprechen den Knoten eines Baums und können an die Nachfolgerknoten Ergebnisse weiterleiten. Wie im Phasenmodell gibt es keine Rückwirkungsmöglichkeit zu Vorgängerkomponenten.

Das heterarchische Modell: Zwischen allen Komponenten besteht die Möglichkeit einer bidirektionale Kommunikation. Bei einer größeren Anzahl von Komponenten ist in einem solchen Modell der Aufwand für die Kommunikation sehr hoch.

Das Blackboard-Modell: In diesem Modell kommunizieren unabhängige Komponenten über einen gemeinsamen Speicher, die Blackboard. Die Komponenten überprüfen den Inhalt der Blackboard nach relevanten Daten und legen dort ihre Ergebnisse ab. Arbeiten die Komponenten parallel zueinander, kann die Blackboard einen Engpaß darstellen.

Das integrierte Modell: Bei diesem Modell kann man eigentlich nicht von einem Architekturmodell sprechen, da identifizierbare Komponenten mit definierten Schnittstellen gerade nicht vorhanden sind. Das integrierte Generierungsmodell gehört in diese Kategorie.

3.4.2 Der Systemaufbau

Nachdem die Randbedingungen geklärt sind, die beim Design von POPEL berücksichtigt werden müssen, und verschiedene Architekturen vorgestellt wurden, kann im nächsten

Schritt die Architektur ausgewählt werden. Dabei kann man sich an den Fragen orientieren, die in [Butterworth, 1980] – dort für die Modellierung von psychologischen Prozessen – aufgestellt werden:

1. Welche Information wird repräsentiert?

2. Welche Verarbeitungsprozesse regulieren den Transfer und die Transformationen der Informationen im Modell?

3. Welche Kapazitätsbeschränkungen treten bei den Transformationen auf, und wie interagieren die verschiedenen Ebenen?

Anhand der Fragen soll der Aufbau und der Verarbeitungsablauf von POPEL definiert werden, ebenso wie die Interaktionen zwischen den einzelnen Komponenten. Zur besseren Orientierung zeigt Abbildung 3.5 in einer Übersicht den Aufbau des Systems. POPEL folgt der Trennung in einen inhaltsfestlegenden (POPEL-WHAT) und einen inhaltsrealisierenden Teil (POPEL-HOW). Eine zusätzliche Komponente ist der Zeigegestengenerator ZORA[7]. In der Abbildung sind die Bezeichnungen der in POPEL verwendeten Wissensquellen eingetragen.

3.4.2.1 Welche Information wird repräsentiert?

Im Sinne der pragmatischen Ordnung der Verarbeitung können folgende Repräsentationsebenen unterschieden werden, zwischen denen eine Wissenstransformation stattfindet:

- die konzeptuelle Ebene (CKB-Ebene): diese Ebene ist die Eingabeebene für POPEL und fungiert als Schnittstelle zwischen POPEL-WHAT und POPEL-HOW;

- die satzsemantische Ebene (FSS-Ebene);

- die syntaktischen Ebenen: in POPEL wird syntaktisches Wissen zweistufig repräsentiert und zwar einerseits die Dominanzrelationen in der Ebene der *dependenzbasierten Struktur* (DBS) und andererseits die lineare Abfolge in der Ebene der *flektierten und linearisierten Struktur* (ILS)[8]. In dieser Ebene erfolgt auch die Flexion der Wörter mittels MORPHIX. Die Trennung der syntaktischen Verarbeitung in zwei Ebenen erleichert sowohl die Verarbeitung in einem Kaskadenmodell (siehe [DeSmedt and Kempen, 1991] und Abschnitt 3.4.3) als auch die Verarbeitung in einer Sprache wie dem Deutschen mit relativ freier Wortstellung.

Daneben gibt es noch Wissensquellen, die nicht in dieses Ebenenmodell fallen, und Wissen über den Kontext enthalten:

- das Benutzermodell (BGP-MS);

[7] ZORA ist ein Akronym für "**Z***eigegestengenerat***OR***progr***A***mm*."
[8] Das 'I' im Akronym stammt aus der englischen Bezeichnung *inflected linearized structure*.

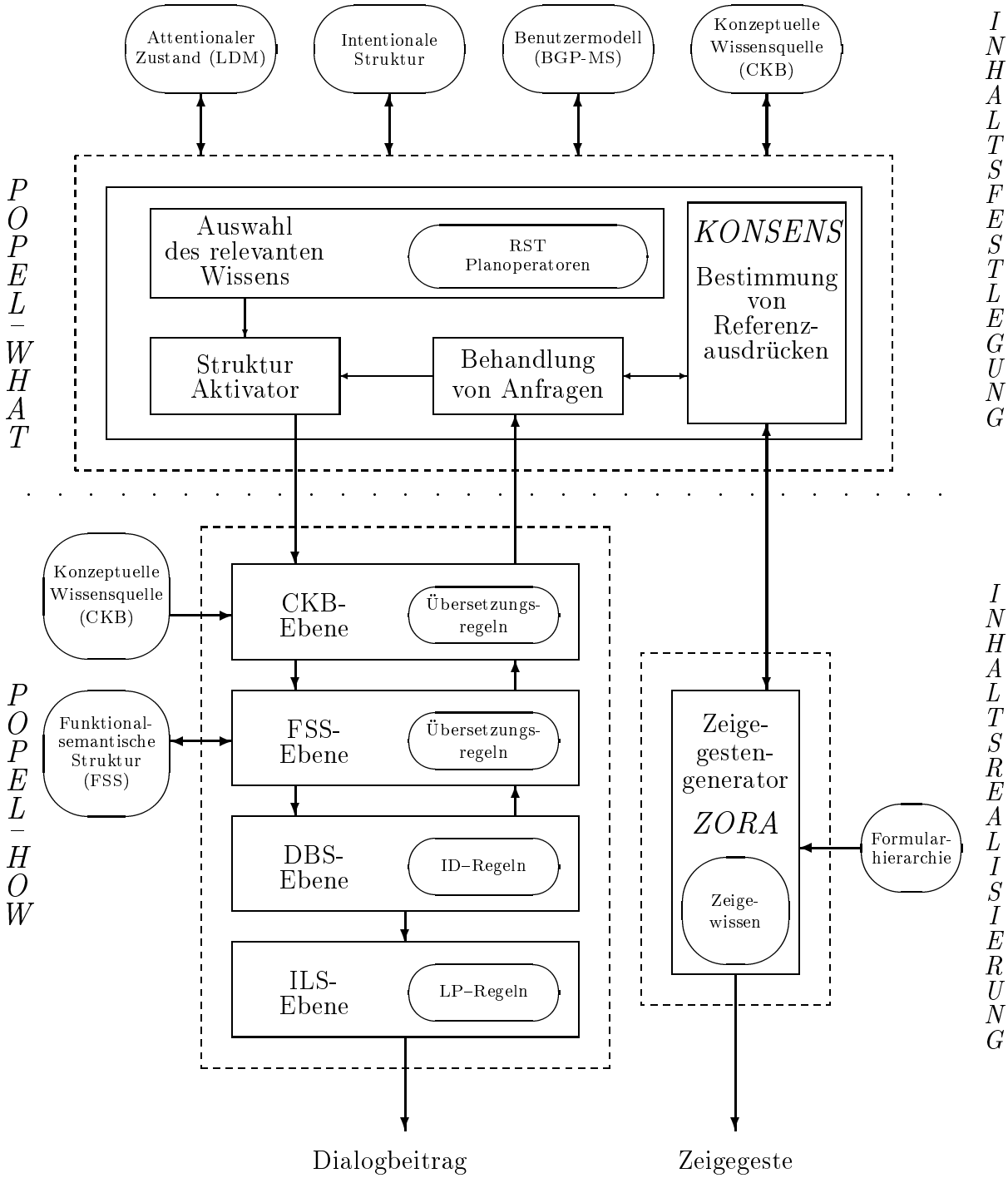


Abbildung 3.5: Die Architektur von POPEL

- das Dialoggedächtnis: die Modellierung des Dialogs in XTRA folgt dem Vorschlag von [Grosz and Sidner, 1986] und besteht aus dem *attentionalen Zustand* im Dialoggedächtnis LDM und der *intentionalen Struktur* (siehe Abschnitt 4.3);
- die Wissensquelle für die Strukturierung des visuellen Kontexts (Formularhierarchie);

3.4.2.2 Welche Verarbeitungsprozesse regulieren den Transfer und die Transformationen der Informationen im Modell?

1. Auswahl des relevanten Wissens

Die Eingabe, die POPEL erhält, kann als ein Ziel aufgefaßt werden, das das System erreichen möchte. Es ist als ein Ausschnitt des konzeptuellen Wissens formuliert. Der erste Schritt der Verarbeitung ist die Auswahl aller derjenigen Teile des konzeptuellen Wissens, die zum Erreichen dieses Ziels in dem jeweilig gegebenen Dialogkontext geäußert werden müssen. Es muß auch sichergestellt werden, daß die Reihenfolge der Auswahl, besonders wenn sie als Text mit mehreren Sätzen realisiert wird, so erfolgt, daß ein kohärenter Dialogbeitrag generiert wird.

Wie bei der Beschreibung der Anforderungen gesagt wurde, muß dieser Schritt so flexibel gehalten sein, daß die ganze Bandbreite von Ellipsen über Einzelsätze bis zu Textabschnitten bearbeitet werden kann. Wenn beispielsweise dem Benutzer die Definition eines Konzepts mitgeteilt werden soll, kann, ähnlich wie im System TEXT, diese Auswahl schemabasiert erfolgen. Für die Auswahlprozesse bei der Generierung einer einfachen Frage ist dieser Ansatz jedoch nicht geeignet. In POPEL basiert die Auswahl auf einer Operationalisierung der *Rhetorical Structure Theory* (RST) (vgl. [Mann and Thompson, 1987a] und Abschnitt 5.1).

2. Übergabe des ausgewählten Wissens

Aufgabe des *Strukturaktivators* ist es, diejenigen Teile des konzeptuellen Wissens, die zur Verbalisierung ausgewählt wurden, an POPEL-HOW zu übergeben. Er steht also an der Schnittstelle zwischen Inhaltsfestlegung und –realisierung und bearbeitet einen Teil dessen, was in Levelts Modell der Sprachgenerierung als *Mikroplanung* bezeichnet wird, z.B. die Bestimmung temporaler Informationen. Ebenfalls hier erfolgt die Strukturierung des Inhalts in einzelne Propositionen und die Entscheidung über die Einbettung einer Proposition in eine andere. Der Strukturaktivator teilt auch den anderen Komponenten des Systems mit, daß die Auswahl des Inhalts beendet ist und terminiert deren Verarbeitung.

3. Behandlung von Anfragen

In Abschnitt 3.2.2.1 wurde das Problem angesprochen, daß aus der Struktur des konzeptuellen Wissen nicht ersichtlich ist, ob und wenn ja in welcher Form eine Teilstruktur überhaupt verbalisiert werden kann. Deshalb müssen Restriktionen, die sich aus der Realisierung ergeben, z.B. daß zusätzliche Informationen der konzeptuellen Ebene zur Verbalisierung benötigt werden, an die inhaltsfestlegende Komponente übergeben werden. Für

Anfragen existiert eine eigene Komponente, die eine 'Annahmestelle' für diese Anfragen ist, und sie entweder selbst bearbeitet oder weiterleitet.

4. Bestimmung von Referenzausdrücken

Ein weiterer Teil der Mikroplanung, nämlich die Bestimmung referentieller Ausdrücke, erfolgt in einer eigenen Komponente (KONSENS). Wenn ein Teil des konzeptuellen Wissens in POPEL-HOW soweit verarbeitet wurde, daß feststeht, daß er als Referenzausdruck realisiert wird, werden hier alle Informationen bestimmt, die notwendig sind, damit der Benutzer das intendierte Referenzobjekt identifizieren kann. Das heißt, es wird unter Verwendung des Benutzermodells und des Diskursmodells festgestellt, ob das Referenzobjekt im impliziten oder expliziten Kontext des Benutzers als bekannt angenommen werden kann und welche Attribute zur Abgrenzung gegen Referenzobjekte der gleichen konzeptuellen Klasse zusätzlich verbalisiert werden müssen. Das schließt auch die Entscheidung ein, ob eine sprachbegleitende Zeigegeste durch ZORA visualisiert werden soll.

5. Transformationen zwischen den Repräsentationsebenen in POPEL-HOW

Ausgehend von der konzeptuellen Repräsentation wird innerhalb der inhaltsrealisierenden Komponente POPEL-HOW in vier Transformationsschritten aus der Eingabestruktur ein Text erzeugt, der schließlich ausgegeben wird. Die Transformationen zwischen konzeptuellen und satzsemantischen Strukturen, und zwischen satzsemantischen und der ersten Stufe der syntaktischen Strukturen erfolgt regelbasiert, indem Teilstrukturen der Ausgangsrepräsentation in Teilstrukturen der Zielrepräsentation abgebildet werden. Dabei wird davon ausgegangen, daß die Abbildung dem *Prinzip der Kompositionalität* folgt, d.h. daß die Abbildung einer komplexen Gesamtstruktur durch die Abbildung von Teilstrukturen möglich ist.

In Abschnitt 3.3.2, in dem die bidirektionale Verwendbarkeit des Transformationswissens in XTRA angesprochen wurde, ist bereits darauf hingewiesen worden, daß die Übergangsregeln für den ersten Schritt gemeinsam mit der Analyse verwendet werden können. Die Regeln für den Übergang von satzsemantischen zu syntaktischen Strukturen können aus dem semantischen Lexikon, das in die satzsemantische Struktur integriert ist, berechnet werden.

Die syntaktische Verarbeitung basiert auf der Trennung syntaktischer Dominanzen von der linearen Abfolge. Im Unterschied zu den beiden oberen Ebenen werden hier keine Transformationsregeln angewandt. Der Übergang zwischen den Ebenen erfolgt dann, wenn vollständige syntaktische Strukturen bestimmt wurden.

Die Wortwahl erfolgt zweistufig während der Abbildungen in POPEL-HOW. Die Inhaltswörter werden während des ersten Transformationsschrittes zwischen der konzeptuellen und der satzsemantischen Ebene gewählt. Die Funktionswörter, die von der syntaktisch-semantischen Struktur der Wörter im Satz abhängen, werden beim Übergang von der satzsemantischen zur ersten syntaktischen Verarbeitungsebene festgelegt.

Ist in einer Verarbeitungsebene keine Abbildung möglich, besteht die Möglichkeit, an die Vorgängerebene Anforderungen nach zusätzlicher Information zu stellen. Hierfür wird untersucht, welche Informationen fehlen, z.B. welche Vorbedingungen einer Regel erfüllt

sein müssen, damit diese anwendbar ist. Diese lokale Rückwirkung kann über die Ebenen propagiert werden, bis sie zu einer Anfrage bei POPEL-WHAT führt.

6. Generierung von Zeigegesten

Bei der Festlegung der Attribute zu einem Referenzausdruck wird festgestellt, ob eine Zeigegeste möglich und sinnvoll ist und es wird gegebenenfalls der Zeigegestengenerator ZORA aktiviert. Das System selbst soll eine benutzerangepaßte Zeigegeste auf einer Graphik erzeugen, die am Bildschirm dargestellt ist. Als externe Wissensquelle steht die Formularhierarchie zur Verfügung, die den geometrischen Aufbau der Graphik repräsentiert. Um die Verarbeitung auch in dieser Komponente flexibel und domänenunabhängig halten zu können, z.B. um neben Formularen auch auf Landkarten zeigen zu können, erfolgt die Verarbeitung in ZORA mit Hilfe von Wissensquellen, die geometrische Eigenschaften der Graphik, Zeigegesten und verschiedene Zeigemittel zueinander in Beziehung setzen, wobei sich die Bewegungen der Zeigegesten aus atomaren Aktionen zusammensetzen.

3.4.2.3 Welche Kapazitätsbeschränkungen treten bei den Transformationen auf, und wie interagieren die verschiedenen Ebenen?

Eine Verarbeitungskomponente ist dadurch charakterisiert, daß sie einen definierten Eingabecode mit dem in der Komponente vorhandenen Wissen verarbeiten kann und ihre Ausgabe in einem ebenfalls definierten Code erzeugt. In einem System, das aus mehreren Modulen besteht, können zwei Komponenten dann interagieren, wenn der Ausgabecode der einen der Eingabecode der anderen Komponente ist. Je mehr Schnittstellen eine Komponente hat, desto größer ist die interne Verarbeitungskomplexität, die benötigt wird, um die verschiedenen Codes miteinander in Beziehung zu setzen.

POPEL ist so konzipiert, daß nur die im Sinne der pragmatischen Ordnung benachbarten Verarbeitungskomponenten miteinander kommunizieren. Sind Interaktionen zwischen nicht benachbarten Komponenten notwendig, müssen diese über die dazwischenliegenden Komponenten laufen, wo sie mit den Repräsentationsmitteln der jeweiligen Ebene reformuliert und weitergegeben werden.

Interaktionen zwischen den Ebenen sind nicht nur von oben nach unten, von der konzeptuellen Repräsentation zur Ausgabe von Dialogbeiträgen möglich und notwendig, sondern auch in der umgekehrten Richtung. Wenn sich in einer Verarbeitungskomponente herausstellen sollte, daß Informationen fehlen, die zur weiteren Verarbeitung notwendig sind, können sie von der nächsthöheren Ebene angefordert werden. Wenn auch diese nicht über das notwendige Wissen verfügt, kann sie die Anfrage weiterleiten, bis sie schließlich an die Komponente zur Behandlung von Anfragen in POPEL-WHAT gelangt. Diese kann, je nach Typ der Anfrage, die verschiedenen Module in POPEL-WHAT aktivieren, damit dort die notwendige Information bereitgestellt wird.

Das System läßt sich als eine doppelte Verarbeitungskaskade interpretieren, wobei parallel zur zweiten Kaskadenstufe der Gestengenerator operiert (siehe Abbildung 3.6). Die äußere Kaskade besteht aus den beiden Komponenten POPEL-WHAT und POPEL-HOW. Intern besteht in den Komponenten nochmals ein kaskadierter Verarbeitungsablauf. In der Inhaltsfestlegung arbeiten Planung und Strukturaktivierung kaskadiert, wobei

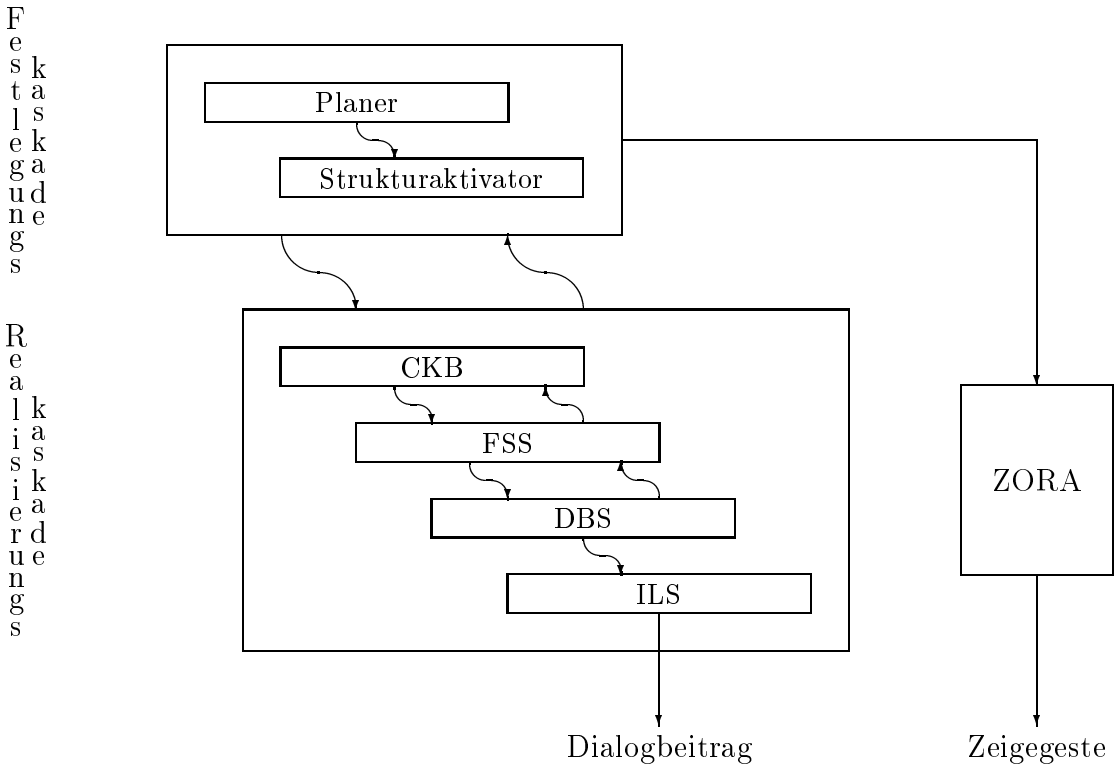


Abbildung 3.6: Die doppelte Verarbeitungskaskade in POPEL

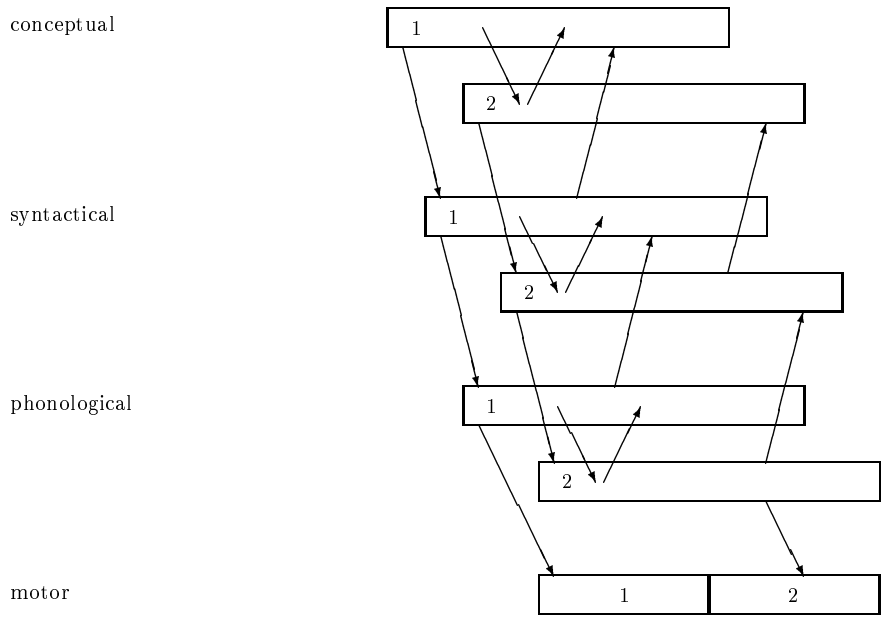


Abbildung 3.7: Jackendoffs Modell des Verlaufs der Sprachproduktion

hier keine direkte Rückwirkung vorhanden ist. Die Kaskadenstufen in der Realisierungskomponente entsprechen den Verarbeitungsebenen. Rückwirkungen sind auf die ersten drei Ebenen beschränkt.

Mit dieser Architektur kann die von Hovy geforderte *präskriptive* oder 'top-down' Auswahl mit der *restriktiven* oder 'bottom-up' gesteuerten Auswahl realisiert werden. Auch die von Hovy (vgl. Abschnitt 2.4.3) genannten Argumente für die Trennung von inhaltsfestlegendem und inhaltsrealisierendem Teil bei gleichzeitiger Interaktion, nämlich Modularität, Sparsamkeit und Angemessenheit, werden erfüllt. Jede Ebene nützt ihr spezielles Wissen aus und fordert zusätzliche Eingaben dann an, wenn Daten fehlen und sie die Verarbeitung nicht weiterführen kann. Anforderungen werden nur an diejenigen Komponenten gestellt, mit denen gemäß der pragmatischen Ordnung sowieso eine Verbindung besteht.

Ein vergleichbares Kaskadenmodell des Ablaufs der Interaktionen bei der menschlichen Sprachproduktion findet sich auch in [Jackendoff, 1987] (vgl. Abbildung 3.7). Anhand zweier Segmente, die verbalisiert werden sollen, werden die Interaktionen zwischen den Segmenten innerhalb einer Verarbeitungsebene und zwischen benachbarten Verarbeitungsebenen dargestellt, wobei die x-Achse dem zeitlichen Verlauf der Produktion entspricht.

Beispielhaft für den möglichen Ablauf der Interaktionen soll hier die Generierung der Ellipse in folgendem Dialogausschnitt stehen:

BEN: *Ich fahre täglich von Saarbrücken nach Völklingen zur Arbeit.*
SYS: *Mit dem Bus?*

Die Eingabe für POPEL besteht, analog zum Inhalt von SWMB in Abbildung 3.3, darin, daß der Benutzer Information zu einem Rollenfüller mitteilen soll. Im Gegensatz zur Abbildung hat das System in diesem Beispiel bereits eine Hypothese für den Rollenfüller, die bestätigt werden soll. Bei der Auswahl der konzeptuellen Repräsentation hat sich POPEL-WHAT entschieden, ausschließlich den Rollenfüller an POPEL-HOW weiterzugeben, da das Ausgangskonzept der Rolle im letzten Satz erwähnt wurde und damit aus dem Kontext erschlossen werden kann.

Für den Rollenfüller wird in POPEL-HOW zunächst eine satzsemantische Beschreibung und dann eine syntaktische Struktur erzeugt, nämlich eine Nominalphrase. Dieser fehlt unter anderem der Kasus, der in einem vollständigen Satz von der Struktur des Verbs bestimmt wird. Da keine weitere Information an POPEL-HOW weitergegeben wird, muß dieses fehlende Merkmal angefordert werden. Die Anfrage von der syntaktischen an die satzsemantische Ebene lautet, daß der Nominalphrase Daten fehlen, die von einer Verbalphrase bereitgestellt werden müßten. In der satzsemantischen Ebene entspricht das dem Fehlen eines Prädikats in der satzsemantischen Struktur, das als ein Argument die satzsemantische Repräsentation der Nominalphrase enthält. Von der satzsemantischen Ebene wird diese Information an die konzeptuelle Ebene weitergeleitet. Dort muß nach einem Element im Kontext des zur Verbalisierung ausgewählten Elements gesucht werden, das diesem Prädikat entspricht. Die Anfrage nach diesem Element wird an POPEL-WHAT gestellt. Aus dem Dialoggedächtnis kann die konzeptuelle Repräsentation des Prädikats des letzten Satzes bestimmt werden, die der Strukturaktivator in die erste Verarbeitungsebene in POPEL-HOW gibt, von wo aus die Information durch die Ebenen propagiert wird. In der satzsemantischen Ebene wird bestimmt, welche Argumentstelle des Prädikats besetzt wird und beim Übergang auf die syntaktischen Ebene, daß die syntaktische Realisierung dieser Argumentstelle einer Präpositionalphrase mit der Präposition *mit* entspricht. Deshalb wird die Nominalphrase zu einer Präpositionalphrase erweitert und kann nun flektiert und ausgegeben werden.

Bei welchen Phänomenen sind Interaktionen überhaupt sinnvoll und notwendig? In PAULINE existieren fünf Interaktionspunkte zwischen dem inhaltsrealisierenden und inhaltsfestlegenden Teil (siehe Abschnitt 2.4.2). Die Architektur von POPEL ist in dieser Hinsicht offen, und legt die Zahl und Art der Interaktionen nicht fest. Wenn neue Wissensquellen in das System integriert werden, wie z.B. über rhetorische Mittel, die zusätzliche Interaktionen notwendig machen sollten, bietet die Architektur einen Rahmen, mit dem diese in dem Modell durchgeführt werden können.

In POPEL wurden die folgenden Interaktionen untersucht:

1. Auswahl konzeptuellen Wissens: wie in Abschnitt 3.2.2.1 gesagt wurde, muß die Auswahl konzeptuellen Wissens durch Restriktionen, die aus der Sprache kommen, unterstützt werden. Wenn beispielsweise zu einem Teil der konzeptuellen Wissensquelle nur zusammen mit einem anderen eine satzsemantische Beschreibung erzeugt werden kann, muß die Auswahlkomponente dies über die Rückwirkung aus dem Abbildungsprozeß erfahren.

2. Generierung von Referenzausdrücken: POPEL-WHAT kann aufgrund der konzeptuellen Struktur des Wissens nicht erkennen, was in der Oberflächenstruktur als

Verb oder Nomen realisiert wird. Andererseits hat POPEL-HOW keinen Zugriff auf die Wissensquellen, die das kontextuelle Wissen speichern, wie z.B. das Benutzermodell. Nur durch die Interaktion von POPEL-WHAT, POPEL-HOW und ZORA kann bestimmt werden, ob ein Teil des konzeptuellen Wissens als eine vollständige Nominalphrase realisiert wird oder nur als Prowort, welche Attribute zur Abgrenzung hinzugefügt werden müssen, welcher Quantor verwendet wird und ob eine Zeigegeste generiert wird.

3. Ellipsen: Wie im obigen Beispiel gezeigt, finden Interaktionen auch dann statt, wenn der Inhalt, der von POPEL-WHAT ausgewählt wird, als Ellipse realisiert wird. Die fehlenden Informationen müssen ebenso wie bei der Bestimmung von Referenzausdrücken mit Hilfe kontextuellen Wissens aus dem Dialoggedächtnis ermittelt werden.

Für diese Punkte wurde eine prototypische Implementation durchgeführt. Zu zwei weiteren möglichen Interaktionen wurden Vorarbeiten durchgeführt, die jedoch noch nicht im Gesamtsystem integriert wurden.

1. Unterstützung der Wortwahl bei der Generierung von Zeigegesten: Ein Referenzausdruck wird von einer Geste begleitet, wenn erwartet wird, daß damit die Referentenidentifikation erfolgversprechender ist. Im Text muß zu der Geste ein begleitendes Wort, z.B. ein Lokaladverb oder ein Demonstrativpronomen, vorhanden sein, das die Verbindung mit der Geste herstellt. Bei der Auswahl dieses Worts kann ZORA helfen, da es Wissen über die räumlichen Relationen der Graphik in Form der Formularhierarchie besitzt. Derzeit bestimmt ZORA mögliche begleitende Wörter, diese Information wird jedoch noch nicht weiterverwendet.

2. Reihenfolge der Auswahl: das Deutsche ist eine Sprache mit relativ freier Abfolge der Argumente eines Verbs. So stehen im unmarkierten Fall Phrasen, die auf bereits bekanntes referieren, z.B. Pronomen, direkt hinter dem finiten Verb, während nominale Phrasen, die neue oder fokussierte Information enthalten, direkt am Anfang des Satzes oder nahe am Ende stehen [Engel, 1988]. Wenn in den Kontextwissensquellen Informationen darüber vorhanden sind, was wichtig und was weniger wichtig ist, kann das von POPEL-WHAT bei der Bestimmung der Reihenfolge berücksichtigt werden. Dazu muß aber von POPEL-HOW mitgeteilt werden, welche Restriktionen vom gerade generierten Satz ausgehen, d.h. wie sich dessen dynamische Struktur entwickelt.

3.4.3 Der Ablauf der Verarbeitung

3.4.3.1 Die inkrementelle und parallele Verarbeitung

Aus den Überlegungen im vorhergehenden Abschnitt folgt, daß die Verarbeitung nicht *sequentiell* zwischen POPEL-WHAT, POPEL-HOW und ZORA erfolgen kann. Durch die Interaktionen zwischen den Komponenten, aber auch durch die Interaktionen innerhalb der verschiedenen Repräsentationsebenen von POPEL-HOW und in den Komponenten von POPEL-WHAT, muß eine *verschränkte* Verarbeitung möglich sein, bei der

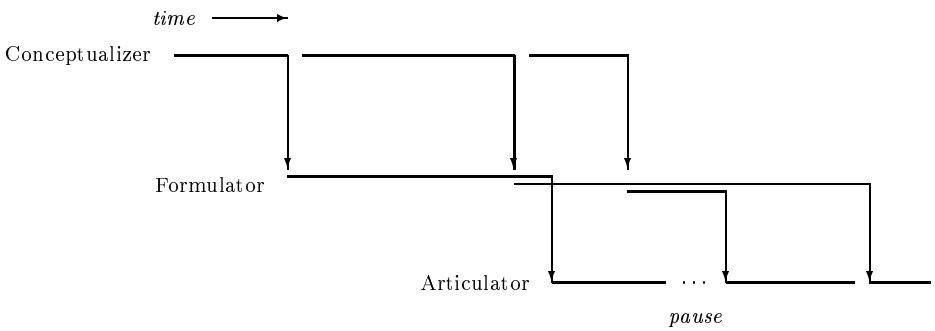


Abbildung 3.8: Das Kaskadenmodell der inkrementellen Generierung

die Kontrolle abwechselnd auf die einzelnen Komponenten übergeht. Es kann also nicht wie im System TEXT eine vollständig spezifizierte *Message* zwischen den Komponenten ausgetauscht werden, da Rückwirkungen zwischen Komponenten die Arbeit gegenseitig beeinflussen können.

Der Aufbau und die Interaktionen zwischen den Komponenten von POPEL legen nahe, die Verarbeitung *inkrementell* ablaufen zu lassen. Das heißt, daß POPEL-WHAT einzelne Teile des konzeptuellen Wissens zur Verbalisierung auswählt und diese an POPEL-HOW übergibt. Während sie dort durch die verschiedenen Ebenen abgebildet werden, können bereits nachfolgende Teile bestimmt werden. Wenn eine Interaktion zwischen den Komponenten notwendig ist, kann diese durchgeführt werden, ohne daß die Verarbeitung der anderen Teile unterbrochen werden muß. Parallel dazu kann ZORA eine Zeigegeste generieren.

Abbildung 3.8 zeigt den zeitlichen Ablauf der Generierung in einem inkrementellen psycholinguistischen Verarbeitungsmodell (nach [DeSmedt, 1990, p.13]). Sobald der *Conceptualizer* einen Teil der Ausgabe bestimmt hat, übergibt er diesen an die nächsttiefere Verarbeitungsstufe. Es ist zudem zu sehen, daß die Reihenfolge, in der das konzeptuelle Wissen ausgewählt wird, in einer Verarbeitungsebene geändert werden kann, falls dies aufgrund der Restriktionen dieser Ebene notwendig ist [Levelt, 1989, p.26]. Aus der Abbildung folgt unmittelbar, daß die Verarbeitung in den verschiedenen Stufen *parallel* erfolgen muß, damit in den verschiedenen Stufen der Kaskade gleichzeitig verschiedene Segmente der zu verbalisierenden Äußerung bearbeitet werden können.

In Abbildung 3.6 wurde die doppelte Verarbeitungskaskade vom POPEL vorgestellt. Dem *Conceptualizer* in Abbildung 3.8 entsprechen in POPEL die Komponenten zur Auswahl und Aktivierung konzeptueller Strukturen, die parallel arbeiten können. Der *Formulator* besteht aus den vier Ebenen von POPEL-HOW, die eigenständige Kaskadenstufen bilden. Parallel zu POPEL-HOW arbeitet ZORA.

3.4.3.2 Die Dekomposition in Teilaufgaben

Inkrementelle Verarbeitung in einer Kaskade setzt voraus, daß die Information in Segmenten von einer Stufe an die nächste weitergegeben wird, damit diese sobald als möglich mit der Bearbeitung beginnen kann. Mit einem solchen Vorgehen kann bereits frühzeitig eine Ausgabe auf der untersten Kaskadenstufe erhalten werden, während nachfolgende Teile noch in der Kaskade bearbeitet werden. Zur Granularität, d.h. zur Frage, welchen Umfang solch ein Segment haben muß, damit es von der nächsten Stufe weiterverarbeitet werden kann, definiert Levelt [Levelt, 1989, p.26]

> "Each processing component will be triggered into activity by a minimal amount of its characteristic input."

Bei jeder Kaskadenstufe muß demgemäß festgelegt werden, welche *characteristische Eingabe* sie verlangt. Da, wie in Abschnitt 3.4.2.2 gesagt, die Abbildungen zwischen den Stufen dem *Prinzip der Kompositionalität* folgen, basiert er auf den kleinsten Einheiten, aus denen sich die Abbildungen zusammensetzen. Auf POPEL bezogen bedeutet das, daß diese kleinsten Einheiten von der Strukturierung der Wissensquellen abhängen, zwischen denen Transformationen erfolgen.

3.4.3.3 Parallelität in POPEL

Die Festlegung des *characteristic input* und damit die Kompositionalität sind Bedingungen dafür, daß die Verarbeitung parallel ablaufen kann. Parallelverarbeitung ist für solche Probleme geeignet, für die eine Strategie vorhanden ist, die es erlaubt, das Problem in voneinander unabhängige Subprobleme aufzuteilen, aus deren Teillösungen die Lösung des Gesamtproblems zusammengesetzt werden kann [Krishnamurthy, 1989]. Diese *Teile–und–Herrsche* Strategie ('divide and conquer') erfordert es, das Problem auf die folgenden drei Aspekte hin zu untersuchen:

- Dekomponierbarkeit: Kann das Problem in Teilaufgaben zerlegt werden?
- Komplexität: Wie effizient ist die Dekomposition, d.h. welchen Gewinn an Geschwindigkeit kann man erhalten?
- Kommunikation: Wie hoch ist der Kommunikationsaufwand?

Eine erste Dekomposition des Gesamtproblems wurde bei der Festlegung der Kaskadenstufen in POPEL bereits durchgeführt. Abgesehen davon kann aber auch in einzelnen Stufen von POPEL-HOW die Verarbeitung parallel ablaufen. In Abbildung 3.8, die die inkrementelle Verarbeitung charakterisiert, sind z.B. zwei Bereiche zu sehen, in denen innerhalb des Formulators parallel an verschiedenen Segmenten gearbeitet wird. Daraus folgt, daß die Verarbeitung in einer Kaskadenstufe nicht von einem einzigen Prozeß durchführt werden kann. Weist man jedem Segment innerhalb einer Stufe einen eigenen Prozeß zu, kann die Verarbeitung innerhalb einer Stufe parallelisiert werden. Da auch die Informationen, die zu einer Rückwirkung zwischen den Stufen der Kaskade und zwischen dem inhaltsrealisierenden und dem inhaltsfestlegenden Teil führen, immer lokal bezüglich

einzelner Segmente sind, können diese Rückwirkungen durchgeführt werden, ohne daß diese die Verarbeitung von Nachbarsegmenten beeinflussen. Die Rückwirkungen stellen somit ebenfalls keinen Engpaß der Verarbeitung da.

Das parallele Verarbeitungsmodell bietet damit auch hinsichtlich des zweiten Aspektes Vorteile, nämlich der Komplexität. Das Gesamtproblem ist aufgrund der Struktur der zugrundegelegten Verarbeitungsstrategien und Wissensquellen von vornherein zur Parallelverarbeitung geeignet und problemlos in Teilaufgaben dekomponierbar. Es fällt also kein zusätzlicher Aufwand an.

Schwieriger ist die Frage nach der Kommunikation zwischen den einzelnen Prozessen zu beantworten. Der Kommunikationsaufwand zwischen den Prozessen hängt davon ab, welche Interaktionen bei der Bearbeitung einzelner Segmente notwendig sind. Hier muß der Grundsatz der Lokalität beachtet werden, daß nämlich Interaktionen, und damit Kommunikation, nur zwischen benachbarten Prozessen stattfinden soll, um die Kommunikationskosten gering zu halten. Besonders solche Interaktionen, die über mehrere Kaskadenstufen hinweggehen und damit nicht lokal sind, müssen so gestaltet werden, daß kein überhöhter Kommunikationsaufwand entsteht (siehe Abschnitt 3.4.2.3). Daneben muß darauf geachtet werden, daß *Deadlocks* vermieden werden, d.h. ein Blockieren von Prozessen, zwischen deren Verarbeitungsreihenfolge komplexe Wiedersprüche existieren.

Kapitel 4

Die externen Wissensquellen für POPEL

Da POPEL innerhalb eines Dialogsystems entwickelt wurde, kann auf die Wissensquellen und Wissensrepräsentationsformalismen des Gesamtsystems zurückgegriffen werden. Dieses Kapitel führt das Wissensrepräsentationssystem SB-ONE ein, in dem das konzeptuelle und satzsemantische Wissen dargestellt wird, sowie das Dialoggedächtnis LDM und das Benutzermodell BGP-MS. Diese beiden Systeme enthalten das kontextuelle Wissen, das z.B. bei der Generierung von Referenzausdrücken benötigt wird.

4.1 SB-ONE und seine Erweiterungen

SB-ONE ist ein Repräsentationssystem aus der Familie der KL-ONE ähnlichen Formalismen und dient zum Aufbau konzeptueller Wissensquellen [Schmolze and Brachman, 1982, Kobsa, 1989, Profitlich, 1990]. Bei der Entwicklung wurden besonders Aspekte berücksichtigt, die sich aus dem Einsatz in einem System zur Verarbeitung natürlicher Sprache ergeben.

4.1.1 Die Repräsentationssprache SB-ONE

SB-ONE ermöglicht es, Wissen in zwei Repräsentationsebenen darzustellen. Es besteht zum einen aus der terminologischen Ebene, der sog. *T-Box*, in der Begriffe und Beziehungen zwischen den Begriffen dargestellt werden, und zum anderen aus einer individualisierten Ebene, der sog. *A-Box*, in der Einzelaussagen über die Welt formuliert werden.

Informell definiert, werden generelle Konzepte der T-Box mit Attributbeschreibungen charakterisiert, die die Relationen zu anderen Konzepten angeben. Die Konzepte sind in einer Subsumptionshierarchie angeordnet, in der ein untergeordnetes Konzept die Attributbeschreibungen des übergeordneten Konzepts erbt. Individualisierte Konzepte der A-Box repräsentieren eine Ausprägung eines generellen Konzepts und seiner Attributbeschreibungen in einer aktuellen Situation. Abbildung 4.1 zeigt eine Beispielswissensquelle, anhand derer die im Zusammenhang dieser Arbeit wichtigen Sprachelemente eingeführt werden sollen.

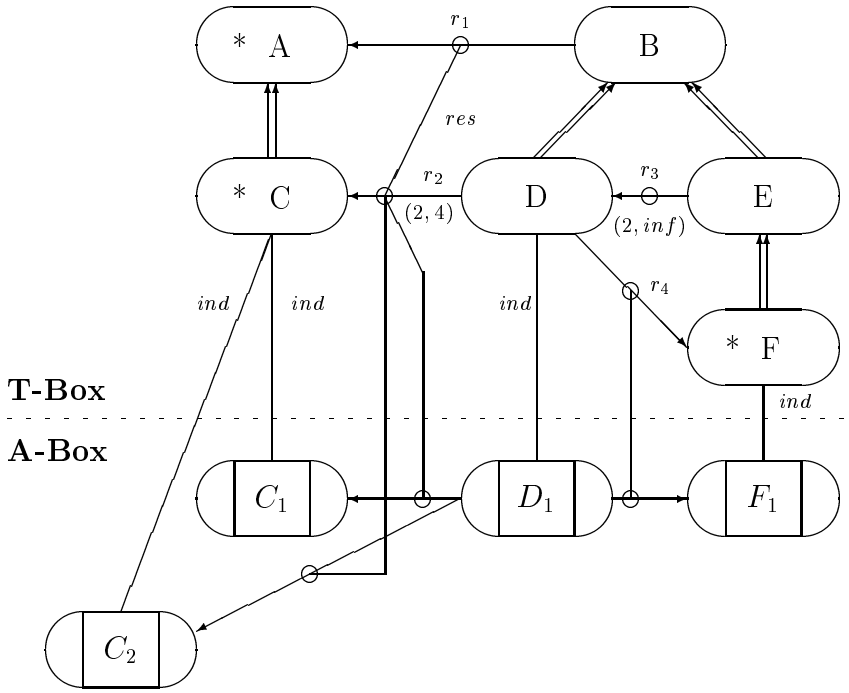


Abbildung 4.1: Die graphische Darstellung der Sprachelemente von SB-ONE

Im einzelnen verfügt SB-ONE über die folgenden Sprachelemente, die auch in anderen an KL-ONE angelehnten Sprachen vorhanden sind [Allgayer and Reddig, 1990, p.3f]:

- *Primitive Konzepte*: Die bei dem Konzept angegebenen Attribute sind nicht hinreichend zu seiner Definition, d.h. die Definition ist unvollständig. Primitive Konzepte können als semantische Primitiva aufgefaßt werden, mit Hilfe derer das Wissen modelliert wird. In Abbildung 4.1 sind Konzepte als Ovale dargestellt, primitive Konzepte zusätzlich mit einem Stern gekennzeichnet.

- *Definierte Konzepte*: Alle notwendigen und hinreichenden Bedingungen zur Definition des Konzepts sind angegeben.

- *Rollen*, *Werteinschränkung* und *Kardinalität*: Attributbeschreibungen werden mit Rollen ausgedrückt. Sie sind Relationen zwischen zwei Konzepten, die den Definitions– und Wertebereich angeben. Die Kardinalität einer Rolle beschränkt die minimale und maximale Anzahl der Rollenfüller. Rollen sind in Abbildung 4.1 mit kleinen Kreisen und r_i gekennzeichnet. (n, m) gibt die Kardinalität an.

- *Individualisierungen*: Individualisierungen von Konzepten und Rollen repräsentieren eine Existenzaussage eines Objekts, das einer generellen Beschreibung genügt. Sie sind als Ovale mit vertikalen Linien dargestellt. Die Individualisierungsbeziehungen zwischen T– und A-Box sind durch *ind*-Kanten abgebildet.

Daneben verfügt SB-ONE zusätzlich über Sprachelemente, die in anderen Systemen nicht vorkommen:

- *'Singleton Concepts'*: Diese Konzepte dürfen nur einmal individualisiert werden, da sie eine Klasse mit genau einem Element repräsentieren, wie z.B. "PAPST".

- *Rollenmodalitäten*: An einer Rolle kann angegeben werden, ob sie bei einem individualisierten Konzept ebenfalls individualisiert werden muß oder nicht, d.h. ob sie notwendig oder optional ist. Der Vorteil gegenüber der Angabe einer Kardinalität $(0, n)$ für optionale Rollen liegt darin, daß die Rollenkardinalität auch bei optionalen Rollen beliebig gewählt werden kann. In Abbildung 4.1 ist z.B. die Rolle r_3, die von dem Konzept F geerbt wird, als optional definiert und wurde nicht individualisiert.

- *Voreinstellungen*: Für das Werteinschränkungskonzept einer Rolle und die Kardinalität können Voreinstellungen (*defaults*) angegeben werden. Wird ein Konzept individualisiert und liegen über die Attributbeschreibungen keine Informationen vor, können auf diese Weise Annahmen über die Existenz weiterer Individualisierungen getroffen werden.

- *Definition von benutzerdefinierten Attributen an Konzepten und Rollen der A– und T-Box*: An den genannten Sprachelementen können Zusatzinformationen annotiert werden, die vom SB-ONE System gespeichert, aber bei der Verarbeitung nicht berücksichtigt werden.

Für die T-Box existiert ein Klassifikationsalgorithmus (*classifier*), der dazu benutzt werden kann, neue Konzepte so in die Subsumptionshierarchie einzufügen, daß sie über allen spezielleren Konzepten und unter allen allgemeineren Konzepten stehen (siehe [Profitlich, 1990, p.18ff]). Auch manuell aufgebaute Wissensquellen können so auf die Korrektheit der Subsumptionsbeziehungen überprüft werden. Das Pendant in der A-Box ist der *realizer*, der zu einer Struktur in der A-Box die speziellsten generellen Konzepte bestimmt, deren Ausprägungen die individualisierten Konzepte sind.

Für SB-ONE existieren weitere Werkzeugsysteme, wie ein leistungsfähiger *Matcher* [Aue *et al.*, 1989], der es erlaubt, nur teilweise spezifizierte SB-ONE Strukturen, die Variablen als Platzhalter für Konzepte und Rollen enthalten, mit einer bestehenden Wissensquelle zu vergleichen, sowie das *Spreading-Activation System* SPREDIAC [Schäfer, 1990] zur intelligenten Pfadsuche in SB-ONE Wissensquellen. Neben der funktionalen Schnittstelle zu SB-ONE wurde eine *graphische Benutzerschnittstelle* für den Wissensingenieur entwickelt, die es erlaubt, Wissensquellen interaktiv am Bildschirm zu erstellen und zu editieren [Kalmes, 1990].

Den Konzepten und Rollen von SB-ONE können zwar vom Entwickler einer T-Box externe Bezeichner zugewiesen werden, diese sind aber für das System völlig bedeutungslos. Die Interpretation der Konzepte, z.B. bei der Klassifikation, hängt alleine von deren

interner Struktur ab. Alle Konzepte und Rollen erhalten vom System einen eindeutigen Bezeichner. Konzepte der T- bzw. A-Box werden mit `GC/IC`, Rollen mit `GR/IR` und einer fortlaufenden Nummer bezeichnet.

Der in Abbildung 3.3 (Seite 41) graphisch dargestellte partielle Ausschnitt einer A-Box sieht als formatierter Text ausgegeben und in einer anderen Individualisierung z.B. folgendermaßen aus:

```
IC2 (IC2) is-a "COMMUTE-WITH-PUBLIC-TRANSPO"(GC58)
  IR3 (IR3 {"OBJECT" . GR132}): IC5 (IC5 {"PHYSICAL-OBJECT" . GC21})
  IR2 (IR2 {"AGENS-TO" . GR133}): IC4 (IC4 {"PERSON" . GC29})
  IR1 (IR1 {"COSTS-R" . GR135}): IC3 (IC3 {"COSTS" . GC14})
```

Hier wurden bei der Individualisierung keine Konzept– oder Rollenbezeichner angegeben. Deshalb verwendet SB-ONE die internen Bezeichner (in Klammern) auch als externe. In geschweiften Klammern stehen Informationen über die Namen und internen Bezeichner der generellen Konzepte und Rollen, nach dem Doppelpunkt die Rollenfüller.

4.1.2 Erweiterungen für SB-ONE: SB-ONE$^+$ und $\mathcal{L}_{\mathcal{SB-ONE}^+}$

Mit SB-ONE können nur Klassen von Objekten (in der T-Box) oder einzelne Elemente dieser Klassen (in der A-Box) beschrieben werden. Um auch Mengen von Objekten adäquat repräsentieren und verarbeiten zu können, wurde SB-ONE$^+$ entwickelt. SB-ONE$^+$ erweitert die T-Box um ein Sprachkonstrukt, mit dem festgelegt wird, ob der Füller einer Rolle als Menge oder als Element interpretiert werden soll. Zudem gibt es einen Rollentyp, der es erlaubt, die Eigenschaften des Rollenfüllers als mengenwertiges Objekt zu beschreiben, z.B. als Teilmenge [Allgayer, 1990].

Alternativ zu dem Zugang zu SB-ONE mittels einer funktionalen oder graphischen Schnittstelle ist in SB-ONE$^+$ eine Termdefinitionssprache integriert, die sog. 'Linearisierte Form' $\mathcal{L}_{\mathcal{SB-ONE}^+}$, mit der Einträge in eine SB-ONE Wissensquelle erstellt und Anfragen an diese formuliert werden können [Allgayer and Schmitt, 1991]. Sie kann von den Verarbeitungskomponenten, die das System verwenden, auch dazu benutzt werden, in einer klaren Syntax Zwischenergebnisse oder Anfragen zu speichern. $\mathcal{L}_{\mathcal{SB-ONE}^+}$ ist auf dem Horn-Klausel Compiler *HC2LC* aufgebaut, der gemäß der prozeduralen Interpretation von PROLOG Horn-Klauseln in Lisp-Funktionen übersetzt [Jansen-Winkeln, 1990].

Die Syntax der Anfragen mittels $\mathcal{L}_{\mathcal{SB-ONE}^+}$ bzw. HC2LC soll anhand des A-Box Operators `irole` demonstriert werden, mit dem auf die Rollenbeziehungen in der A-Box zugegriffen werden kann. `irole` hat drei Argumente, nämlich die Rolle, das Ausgangs– und das Zielkonzept der Rolle. Die Anfrage selbst wird mit dem Operator `ask` durchgeführt. Ist eine A-Box gegeben, die das oben gezeigte individualisierte Konzept `ic2` enthält, sind folgende Anfragen möglich:

Anfrage	Ergebnis
`(ask ()(irole ir3 ic2 ic5))`	`yes`
`(ask ()(irole ir3 ic2 ic45))`	`no`
`(ask ()(irole ir3 ?x ic5))`	`((?x = ic2))`
`(ask ()(irole ?r ic2 ic5))`	`((?r = ir3))`

Wie zu sehen ist, werden Variablen in den Anfragen mit einem Fragezeichen präfigiert. Auf die Anfrage

```
(ask ()(irole ?r ic2 ?ran))
```

sind für `?r` und `?ran` mehrere Variablenbindungen möglich. Der Interpreter liefert zuerst eine Lösung. Mit der Funktion `(next-solution)` wird ein *Backtracking* durchgeführt und liefert bei jedem Aufruf eine weitere Lösung, falls eine solche existiert, ansonsten den Wert `no`.

Es ist möglich, $\mathcal{L}_{\mathcal{SB}-\mathcal{ONE}^+}$ um selbstdefinierte HC2LC-Prädikate zu erweitern. Die Anfrage nach dem generellen Konzept des Füllers einer individualisierten Rolle kann beispielsweise folgendermaßen definiert werden:

```
(defpred GENERAL-CONCEPT-OF-FILLER
   ((GENERAL-CONCEPT-OF-FILLER ?ind-role ?general-concept)
       (irole ?ind-role ?idomain ?irange)
       (isa ?irange ?general-concept)))
```

`isa` überprüft die Individualisierungsbeziehung. Wiederum mit obiger Individualisierung ergibt sich

```
(ask ()(GENERAL-CONCEPT-OF-FILLER ir2 ?gen-concept))
⇒ ((?gen-concept = CG29))
```

4.2 Das konzeptuelle und satzsemantische Wissen

Wie bei der Übersicht über das Dialogsystem XTRA bereits erwähnt wurde, sind zwei mit SB-ONE implementierte Wissensquellen in dem System vorhanden, die *Conceptual-Knowledge Base* (CKB) und die *Functional-Semantic Structure* (FSS).

Die Unterscheidung der konzeptuellen von der satzsemantischen Repräsentation erfolgte, um domänenabhängiges von domänenunabhängigem Wissen zu trennen. Die interne Information der CKB muß in irgendeiner Weise mit sprachlichem Wissen in Beziehung gebracht werden. Ohne die Zwischenstufe der FSS wäre der Übergang zwischen der domänenabhängigen, internen Repräsentation zu einer domänenunabhängigen, sprachnahen Struktur sehr aufwendig.[1] Die FSS dient deshalb als Vermittler zwischen diesen beiden, indem eine Taxonomie von Aktionen, Objekten und Eigenschaften definiert wird, sowie die Strukturen, die aus diesen zusammengesetzt werden können. Bei der Verarbeitung wird auf der Basis dieses Wissens die satzsemantische Zwischenrepräsentation aufgebaut, indem Individualisierungen in der A-Box der FSS erzeugt werden. Die Transformationen zwischen CKB und FSS bei der Generierung und Analyse sind somit unabhängig von grammatischem Wissen möglich. Ebenso kann die syntaktisch-semantische Verarbeitung domänenunabhängig durchgeführt werden. Lediglich das semantische Lexikon muß um den jeweils benötigten Domänenwortschatz erweitert werden.

[1] Beispielsweise wurde der Generator MUMBLE-86 gemäß dieser Philosophie entwickelt. Inzwischen wurde jedoch auch für diesen Generator eine satzsemantische Zwischenrepräsentation entwickelt [Meteer, 1990a]

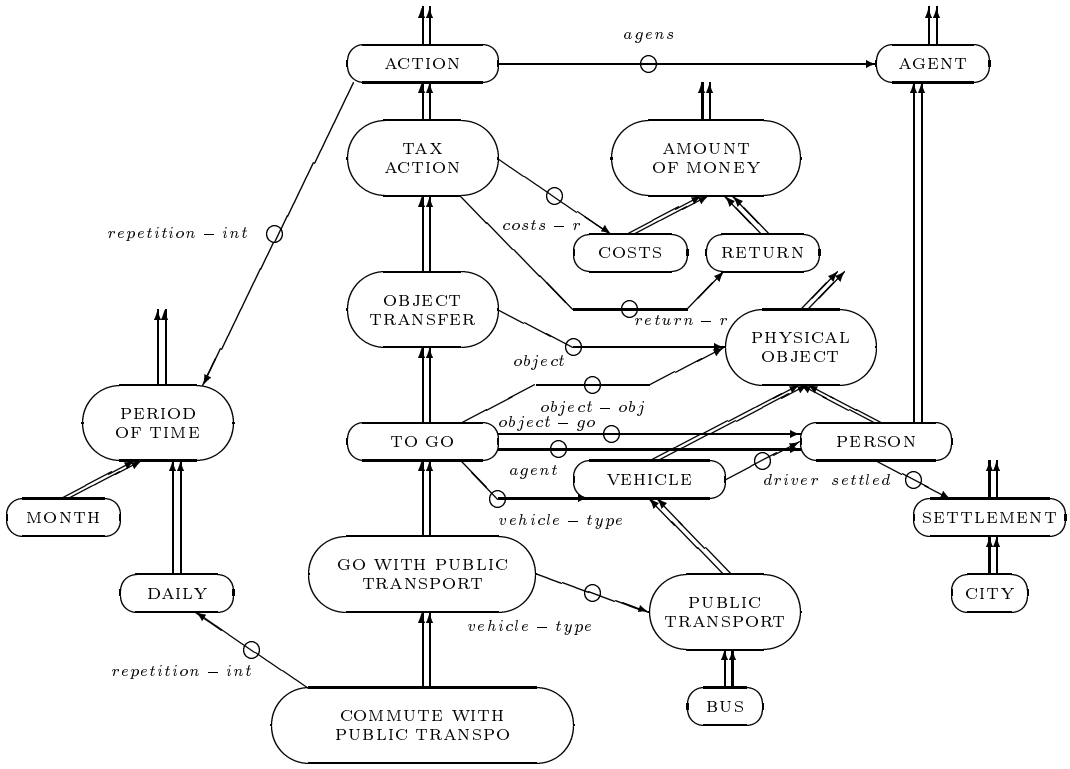


Abbildung 4.2: Ein Ausschnitt aus der konzeptuellen Wissensquelle (vereinfacht)

Eine satzsemantische Zwischenebene findet sich inzwischen in verschiedenen natürlichsprachlichen Systemen, wobei das *Upper Model* des PENMAN-Systems diejenige sein dürfte, die am weitesten entwickelt ist (vgl. [Bateman, 1990], auch für eine Übersicht über andere Systeme).

4.2.1 Die konzeptuelle Wissensquelle

Die CKB, die mit der terminologischen Wissensquelle des angeschlossenen Expertensystems eng verbunden ist, basiert primär auf der Struktur des Expertenwissens (siehe Abschnitt 3.2.2.1). Abbildung 4.2 zeigt einen vereinfachten Ausschnitt der T-Box, die bei der Kopplung mit dem System LST-1 verwendet wird. Die Konzept– und Rollennamen spiegeln wider, daß die Strukturierung unter dem Gesichtspunkt des Steuerrechts erfolgt ist.

Ein Beispiel aus dieser Wissensquelle ist das Konzept `TAX-ACTION`, das steuerlich absetzbare Aktionen beschreibt, und das zwei Rollen besitzt, für die Kosten der Aktion `costs-r` und den Rückerstattungsbetrag `return-r`. Zudem erbt es die Rollen für die ausführende Person `agens` und ein Wiederholungsintervall `repetition-int` von dem Konzept `ACTION`.

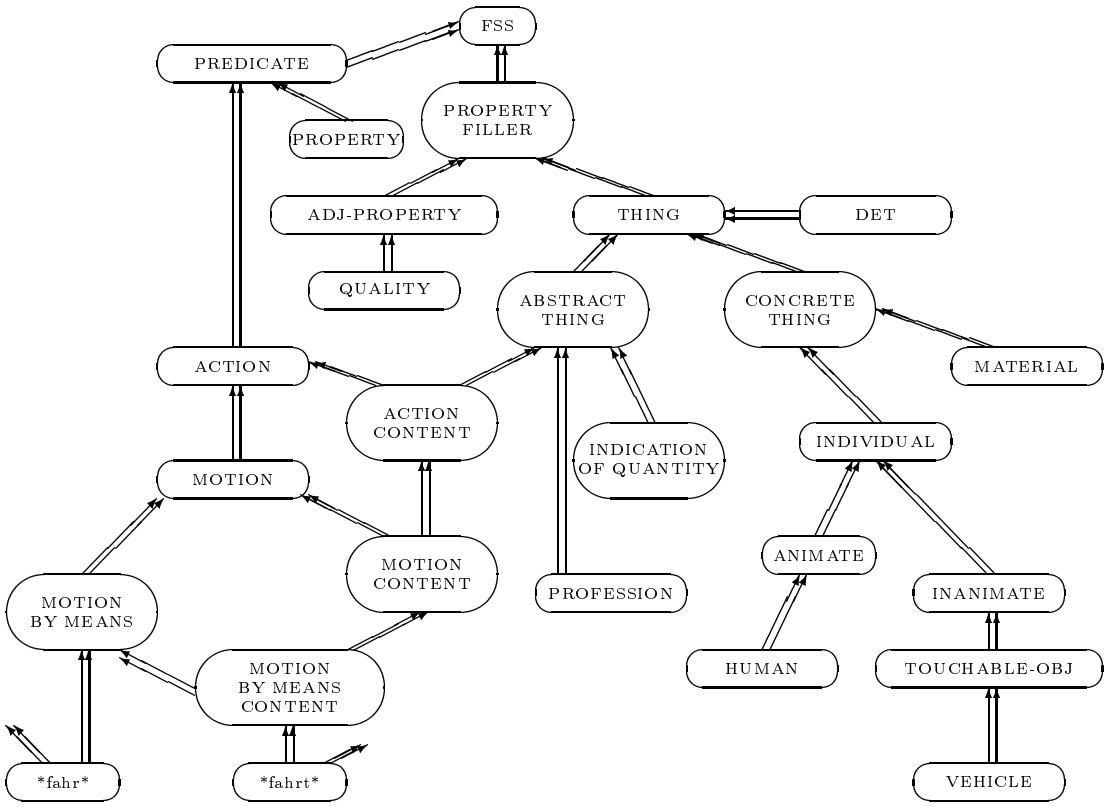


Abbildung 4.3: Ein Ausschnitt aus der Taxonomie der FSS

4.2.2 Die funktional-semantische Struktur und das semantische Lexikon

Als Zwischenstufe bei der Abbildung des konzeptuellen Wissens auf natürliche Sprache wird die domänenunabhängige satzsemantische Repräsentation der FSS verwendet. Die FSS repräsentiert in ihrer T-Box, wie sprachlich wohlgeformte Prädikat/Argument Strukturen aufgebaut sein müssen, damit sie verbalisiert bzw. analysiert werden können (siehe Abbildung 4.3). Die FSS unterscheidet zwischen Konzepten, die als Nomen (Subkonzepte von `THING`), Verben (Subkonzepte von `PREDICATE`) und Adjektive (Subkonzepte von `ADJ-PROPERTY`) verbalisiert werden können. Diese Klassifikation ist jedoch kein Baum, z.B. haben Nominalisierungen von Verben sowohl nominale als auch verbale Superkonzepte. Der Vorrat an Rollen ist im Gegensatz zu der CKB ebenfalls domänenunabhängig. Die Rollen beschreiben entweder Tiefenkasusbeziehungen zwischen den Konzepten (11 verschiedene Rollen, vgl. [von Polenz, 1985]), Modifikationen und Determination bei nominalen Konzepten (13 verschiedene Rollen), oder den Sprechakttyp bei Verb-Konzepten.

Das *semantische Lexikon* ist eine Erweiterung der FSS und verwendet deren domänenunabhängige Terminologie, um die Semantik der Wörter zu definieren, die sowohl zum domänenunabhängigen 'Grundwortschatz' als auch zum Wortschatz der Anwendungs-

domäne gehören. Hierfür wird zu einem Wort ein Konzept definiert und als Subkonzept unter das FSS-Konzept eingefügt, dessen semantische Struktur mit der des Wortes übereinstimmt. Zudem enthält das Lexikon Wissen darüber, welchen syntaktischen Strukturen die Rollen der FSS entsprechen können. Wird eine Äußerung generiert oder analysiert, individualisieren die Verarbeitungskomponenten für die Wörter der Äußerung die entsprechenden Konzepte des semantischen Lexikons in der A-Box der FSS.

Für den Aufbau und die Erweiterung des semantischen Lexikons wurde ein Wissensakquisitionssystem entwickelt, das mittels interaktiver Klassifikation für eine konsistente Struktur sorgt [Ndiaye, 1990].

Ein Beispiel für einen Eintrag im semantischen Lexikon ist z.B. das Verb *kosten*:

```
"*KOST*" (GC104) is-a (("*LEXICON*" (GC79)) ("COST" (GC50)))
  "MEASURE" (GR420): "INDICATION-OF-QUANTITY" (GC42) [(1 1 1), NEC]
  "SUBJECT" (GR416): "THING" (GC20)  [(1 1 1), NEC]
  "ILLOC" (GR422): "SPEECH ACT" (GC58)  [(1 1 1), NEC]
  "TIME" (GR417): "TIME" (GC2)  [(1 1 1), OPT]
  "CAUSE" (GR421): "PREDICATE" (GC46)  [(1 1 1), OPT]
  "LOCATION" (GR418): "THING" (GC20)  [(1 1 1), OPT]
  "RESULT" (GR415): "THING" (GC20)  [(1 1 1), OPT]
  "PURPOSE" (GR419): "FSS" (GC1)  [(1 1 1), OPT]
```

Durch die Angabe des Superkonzepts `*LEXICON*` ist es als Eintrag des semantischen Lexikons ausgewiesen. Die Rollen werden von dem Konzept `COST` geerbt. Nicht dargestellt sind die Attribute an den Rollen, die angeben, mit welcher syntaktischen Funktion eine Rolle bzw. deren Füller realisiert werden kann, z.B. der Füller der Rolle `subject` als Nominalphrase im Nominativ.

4.3 Die Diskursstrukturierung

Ebenso wie die Teile eines Satzes, die in vielen Sprachen nicht beliebig angeordnet werden können, unterliegen auch Folgen von Äußerungen bestimmten Regularitäten, die dafür sorgen, daß aus den Einzeläußerungen ein kohärenter Text oder Dialogbeitrag wird. Kohärenzstiftende Mittel sind beispielsweise die Verwendung definiter oder pronominaler Referenzen auf vorerwähnte Diskursobjekte oder die Verwendung von Partikeln, die die Struktur des Textes anzeigen. Auch die rhetorischen und inhaltlichen Beziehungen zwischen aufeinanderfolgenden Äußerungen machen aus diesen ein zusammengehörendes Ganzes. Grundlage für die Verarbeitung dieser Regularitäten in XTRA ist eine Wissensquelle, die den Diskurs strukturiert [Reithinger, 1989].

Der Aufbau der Wissensquelle zur Diskursstrukturierung basiert im wesentlichen auf [Grosz and Sidner, 1986] und [Reichman, 1985]. Die Verwendung der Diskursstrukturierung bei der Generierung von Referenzausdrücken wird in Kapitel 7 beschrieben, eine Übersicht der Anwendung bei der Analyse findet sich in [Reithinger, 1989, p.12f].

4.3.1 Die Grundlagen der Diskursstruktur

Grosz & Sidner

Die bekannteste Theorie auf dem Gebiet Diskursstrukturierung ist die in [Grosz and Sidner, 1986] beschriebene. Danach läßt sich die Struktur eines Diskurses mit folgenden drei Komponenten darstellen:

- der textuellen Beschreibung;
- der intentionalen Struktur;
- dem attentionalen Zustand;

Die *intentionale Struktur* basiert auf dem Zweck eines Diskurses, dem *Discourse Purpose* (DP), der sich rekursiv in Segmente aufteilen läßt, die *Discourse Segment Purposes* (DSP). Zwischen den DSPs bestehen zwei fundamentale Beziehungen, nämlich

- Dominanz: ein DSP_1 dominiert DSP_2, wenn DSP_2 zum Erreichen des Ziels von DSP_1 beiträgt;
- "Satisfaction Precedence": DSP_1 geht DSP_2 voraus ('satisfaction-precedes'), wenn DSP_1 vor DSP_2 erfüllt werden muß;

Aus der Dominanz-Relation folgt, daß die DSPs hierarchisch aufgebaut sind, und aus der Satisfaction-Precedence, daß es zwischen den Knoten derselben Hierarchiestufe eine Reihenfolge gibt. Die beiden Beziehungen lassen sich beispielsweise bei der Analyse einer Äußerung aus der Verwendung von Partikeln ableiten.

Der *attentionale Zustand* besteht in einem strukturierten Gedächtnis von Diskursbeiträgen. Seine Struktur entspricht einem Kellerspeicher, dessen oberstes Element zu einem DSP gehört. Kommt ein neuer DSP, wird dessen Repräsentation zuoberst auf den Speicher gelegt, wenn der alte DSP in Dominanz-Relation zum neuen DSP steht, ansonsten wird vorher das oberste Element aus dem Speicher entfernt.

Verwendet werden kann dieser Speicher z.B. zur Analyse und Generierung von Referenzausdrücken. So kann mit einem Pronomen nur auf Diskursbeiträge referiert werden, die zu dem obersten Kellerelement gehören, und damit zum derzeitigen DSP, oder zu einem diesen dominierenden DSP.

Reichman

Reichman [Reichman, 1985] betrachtet in ihren empirischen Untersuchungen primär Diskussionen, deren Abschnitte in sog. *Contextspaces* gegliedert sind. In einem Contextspace befinden sich die Repräsentanten aller zu einem Diskursabschnitt gehörenden Propositionen. Zudem wird z.B. gespeichert, welches Ziel mit dem Diskursabschnitt verfolgt wird und welchen Status er besitzt. Der Status eines Contextspace gibt an, ob die Inhalte seiner Propositionen sich derzeit im Vordergrund der Diskussion befinden, unterbrochen oder abgehandelt sind. Er stellt damit Informationen bereit, mit denen entschieden werden kann, mit welchen Mitteln auf seine Elemente referiert werden kann.

Zwischen den Contextspaces können Relationen existieren, die angeben, wie die einzelnen Abschnitte zusammenhängen. Eine *support* Relation beispielsweise besteht zwischen zwei Contextspaces, von denen einer unterstützende Argumente zu dem vorher angesprochenen Thema des anderen enthält. Der Übergang zwischen den Contextspaces erfolgt durch sog. *Conversational Moves*. Ähnlich wie im Modell von Grosz & Sidner kann der Übergang durch Partikeln oder feststehende Redewendungen markiert werden. Für neun *Moves* werden sprachliche Markierungen und Diskursstrukturen an Beispieldialogen demonstriert.

Kritik an den Ansätzen

Das Modell von Grosz & Sidner trennt, im Gegensatz zu Reichman, die Intentionen, die einem Diskurs zugrundeliegen, von der Repräsentation der sprachlichen Struktur des Dialogs. In [Grosz and Sidner, 1986, p.202] wird zudem die Behauptung aufgestellt, daß rhetorische Relationen, wie die *support* Relation von Reichmann, aber auch die im System TEXT oder in [Mann and Thompson, 1987a] definierten, weniger allgemein seien als die beiden Beziehungen ihres Modells, da diese nicht an Sprache gebunden seien und nur von der Planstruktur der Aufgabe abhängen, die zu lösen ist.

Diese Analyse, die bei einem Dialog richtig sein kann, in dem es darum geht, kooperativ eine Aufgabe zu lösen, vermag aber nur einen Teil des sprachlichen Verhaltens adäquat darzustellen. Wenn ein Dialogpartner von der gemeinsam zu lösenden Aufgabe abschweift, z.B. um eine Bemerkung zu einem bereits abgeschlossenen DSP zu machen, oder wenn Parallelen zwischen zwei DSPs hergestellt werden sollen, können gegenseitige Bezüge nicht dargestellt werden, da DSPs nur hierarchisch dargestellt sind und Querverbindungen zwischen DSPs nicht möglich sind.

4.3.2 Das Linguistic Dialog Memory (LDM)

In XTRA existiert mit dem *Linguistic Dialog Memory* (LDM) eine Wissensquelle, mit der der attentionale Zustand repräsentiert und verarbeitet werden kann (siehe Abbildung 4.4). Ziel des Entwurfs ist es, den Verarbeitungskomponenten eine Wissensquelle zur Verfügung zu stellen, die den Zusammenhang von sprachlicher Struktur, konzeptueller Darstellung und thematischem Umfeld speichert.

Im LDM ist in den *Referentiellen Objekten* (ROs) der Zusammenhang zwischen sprachlicher und konzeptueller Struktur von Objekten und Vorgängen gespeichert. Jedes RO enthält einen oder mehrere Verweise in die konzeptuelle Wissensquelle der CKB (A- oder T-Box) und zudem eine Liste von Einträgen, die angeben, an welcher Stelle des Dialogs die konzeptuellen Einheiten wie angesprochen wurden. Ein solcher Eintrag besteht aus zwei Verweisen. Der eine geht zur FSS und gibt die satzsemantische Struktur der Äußerung an, der andere geht in das *Dialog Sequence Memory* (DSM) und gibt die Position im Dialogverlauf an.

Das DSM speichert die Abfolge, in der die Elemente der ROs geäußert wurden. Jedes DSM-Elemente besteht aus einem Verweis auf einen Eintrag in das zugehörige RO, einer Markierung, welcher Dialogpartner gesprochen hat, einem Fokuswert, sowie einem Verweis

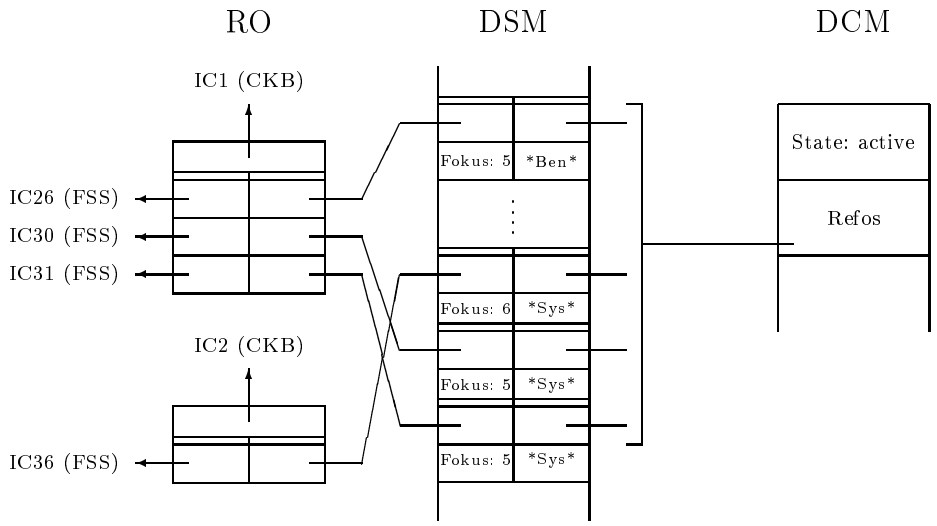


Abbildung 4.4: Ein Ausschnitt aus dem LDM

auf die kontextuelle Struktur des DCM (s.u.), zu dem diese Äußerung gehört.

Die Fokuswerte der Einträge des DSM dienen dazu, dem System Wissen an die Hand zu geben, mit dem es Referenzen verarbeiten kann. Für den Generator bedeutet dies, daß er Referenzausdrücke so erzeugen kann, daß der Dialogpartner den intendierten Referenten unmittelbar identifizieren kann. Die Skala der Werte basiert auf [Reichman, 1985, p.75f]. Dort werden vier verschiedene (symbolische) Fokuswerte und Übergangsregeln zwischen diesen Werten definiert. Den höchsten Fokuswert erhält z.B. ein neu eingeführtes Element, oder ein Element, das im Vorfeld eines Satzes steht und somit topikalisiert ist. Die Fokuswerte werden im DCM im Gegensatz zu Reichman als Zahlen gespeichert.

Die attentionale Struktur der Dialogsegmente wird in den Kontexträumen des *Dialog Context Memories* (DCM) und deren gegenseitigen Abhängigkeiten repräsentiert. Jeder Kontextraum enthält einen Eintrag, der angibt, zu welchem Abschnitt des Diskurses er gehört (vergleichbar dem DSP), und Verweise auf diejenigen Elemente des DSM, die zu ihm gehören.

Wenn der DSP wechselt, wird ein *Move* auf dem DCM durchgeführt, der dazu führt, daß der bisherige Kontextraum in einen anderen *Zustand* übergeht. Die Struktur der Kontexträume im DCM repräsentiert somit den 'attentional state' des Dialogs. Beispiele für Zustände eines Kontextraums sind z.B. *closed*, wenn der DSP abgearbeitet ist, oder *active*, wenn über den zum Kontextraum gehörigen DSP gerade der Dialog geführt wird.

Die Struktur des DCM und der Zustand der Kontexträume wird durch die *Moves* auf dem DCM bestimmt. Diese sind von der Veränderung der intentionalen Struktur oder der rhetorischen Struktur des Diskurses abhängig und müssen von außen gesteuert werden. In [Reithinger, 1989] wird ein prototypisches System vorgestellt, das diese Leistung erbringen kann.

Ein Beispiel für einen *Move* ist *open*, bei dem ein neuer Kontextraum geöffnet wird. Der Knoten der intentionalen Struktur, der zu dieser Äußerung gehört, hat noch keinen dominierenden Knoten, und es ist kein Kontextraum mit dem Zustand *active* vorhanden. Dieser *Move* wird am Anfang des Dialogs durchgeführt und dann, wenn der vorherige Dialogabschnitt mit *close* beendet wurde. Ein *Move* kann entweder explizit in den Dialogbeiträgen markiert sein, oder aus dem Inhalt in der Äußerung abgeleitet werden. Ein *interrupt* kann beispielsweise dadurch festgestellt werden, daß das Ziel, das mit dieser Äußerung verbunden ist, nicht zu den Zielen des gerade bearbeiteten Diskursabschnitts paßt.

4.4 Das Benutzermodell BGP-MS

Wie in Abschnitt 3.2.1 bereits gesagt wurde, benötigt ein Dialogsystem ein Benutzermodell, das Annahmen über das Wissen und die Ziele des Benutzer verwaltet (siehe [Kobsa and Wahlster, 1988] für einen Überblick über den Stand der Forschung in diesem Bereich). Besonders für ein Generierungssystem ist es von großer Bedeutung, da z.B. die Entscheidung, wie ein bestimmter Sachverhalt dem Benutzer mitgeteilt werden soll, davon abhängt, ob der Benutzer mit dem Problemkreis vertraut oder ein Laie ist [Kass and Finin, 1987].

XTRA verfügt über das Benutzermodellierungssystem BGP-MS[2] [Kobsa, 1990], das Annahmen über das konzeptuelle Wissen des Benutzers und des Systems repräsentiert. Grundlage der Modellierung ist die Partitionierung des konzeptuellen Wissens in Bereiche, von deren Inhalt das System annimmt, daß darüber eine gewisse Überzeugung ('belief') besteht. So umfaßt die Partition mit dem Namen *SBMB*, was für *System Believes Mutual Belief exists* steht, diejenigen Teile der CKB, von denen, vereinfacht gesagt, das System annimmt, daß eine gemeinsame Überzeugung bei den Dialogpartnern vorhanden ist, daß beide sie kennen (siehe [Kobsa, 1985] für eine eingehende Behandlung der Problematik geschachtelter und gegenseitiger Überzeugungen). Eine etwas komplexere Überzeugung ist in der Partition *SBUBMBSB* repräsentiert, die den CKB-Teil enthält, von dem das System annimmt, daß der Benutzer annimmt, daß eine gemeinsame Überzeugung besteht, daß das System selbst über dieses Wissen verfügt. Insgesamt sind derzeit 42 solcher Partitionen über der CKB definiert.

Auch die Ziele ('wants') des Benutzers und des Systems werden in dieser Weise repräsentiert. Die CKB-Repräsentation der Eingabe für POPEL ist in einem der Kontexte für die Repräsentation der Systemziele enthalten, in dem Kontext *SWMB* (siehe Abschnitt 3.2.5). Dort wurde auch bildlich dargestellt, daß eine Partition nicht unbedingt wohlgeformte SB-ONE Ausdrücke enthalten muß. So kann in einer Partition nur eine Rolle selbst enthalten sein, nicht aber deren Definitions– oder Werteinschränkungskonzepte, die in einer anderen Partition stehen können.

Aufbauend auf der Partitionierung bietet BGP-MS die Möglichkeit, Benutzerstereotype zu definieren und in einem aktuellen Dialog zu aktivieren. Aus der Beobachtung der

[2]BGP-MS ist ein Akronym für "**B**elief, **G**oal and **P**lan **M**aintenance **S**ystem".

Benutzereingabe können so Annahmen über das wahrscheinliche Wissen eines Dialogpartners gewonnen werden.

Um das relativ komplexe System von verschachtelten Kontexten effizient verwalten zu können, wurde SB-ONE um das Partitionsverwaltungssystem SB-PART erweitert [Scherer, 1990], das es erlaubt, die Partitionen in einer Hierarchie redundanzfrei zu verwalten.

Durch die enge Anbindung des Benutzermodells an SB-ONE können Wissensquellen, die nicht in dieser Sprache formuliert sind, z.B. die Grammatik oder das syntaktische Lexikon, nur mittels Abbildung in SB-ONE mit abgedeckt werden. Diese Duplizierung könnte zu Inkonsistenzen führen und die Abbildungsprozesse unnötig verkomplizieren. Für einen Generator ist jedoch das Wissen über den Sprach- und Wortgebrauch eines Dialogpartners ein wichtiges Entscheidungskriterium, wenn es darum geht, die sprachliche Struktur der Äußerung so auszuwählen, daß sie die Sprachkompetenz des Dialogpartners berücksichtigt.

Kapitel 5

Die Auswahl und Strukturierung eines Dialogbeitrags in POPEL

Die erste Aufgabe, die POPEL erfüllen muß, ist die Auswahl und Strukturierung der konzeptuellen Information, die in dem jeweilig aktuellen Dialogkontext geäußert werden soll. In Kapitel 2 wurden der planungsbasierte Ansatz von KAMP für die Generierung von Sätzen, sowie der schemabasierte Ansatz von TEXT für die Generierung von kurzen Texten vorgestellt. Die in POPEL auftretende Bandbreite von Äußerungen, die erzeugt werden sollen, erfordert ein Verfahren, das, ausgehend von den Intentionen, die es als Eingabe erhält, flexibel und der Situation angemessen arbeiten kann.

Dabei soll das Prinzip beachtet werden, daß in POPEL wenn möglich keine Spezialösungen verwendet werden, sondern dem Stand der Forschung entsprechende, auch in anderen Systemen verwendete Methoden und Repräsentationen. Zur Beschreibung der Struktur von kohärenten Texten wird hierzu in verschiedenen Systemen die *Rhetorical Structure Theory* (RST) [Mann and Thompson, 1987a] verwendet. Die Auswahl und Strukturierung des Inhalts in POPEL basiert auf einer Operationalisierung dieser Theorie. Zunächst wird die ursprünglich zur Analyse von kohärenten Texten vorgesehene Theorie vorgestellt, dann eine Operationalisierung, die nur zur Strukturierung einer bereits vollständig zur Verbalisierung ausgewählten Wissensquelle dient. Danach wird eine Operationalisierung behandelt, die die RST auch zur Auswahl des Inhalts verwendet, und die Ausgangspunkt für die Auswahl und Strukturierung in POPEL ist. Abschließend wird die RST-Version vorgestellt, die für POPEL entwickelt wurde, sowie die Realisierung der Verfahren zur Auswahl und Strukturierung des Inhalts.

5.1 Die Rhetorical Structure Theory

5.1.1 Die deskriptive Version von Mann & Thompson

RST ist in der Version, die in [Mann and Thompson, 1987a] vorgestellt wird, eine deskriptive Theorie, die angibt, welche Beziehungen zwischen Sätzen und Gruppierungen von Sätzen bestehen müssen, damit diese von einem Leser als *kohärent* und damit als Text aufgefaßt werden. Zwischen zusammengehörenden Textteilen muß eine der rhetori-

Circumstance	Antithesis
Solutionhood	Concession
Elaboration	Condition
Enablement	Otherwise
Motivation	Interpretation
Evidence	Evaluation
Justify	Restatement
Volitional Cause	Non-Volitional Cause
Volitional Result	Non-Volitional Result
Summary	Sequence
Background	Contrast
Purpose	

Abbildung 5.1: Die RST-Relationen

schen Relationen identifiziert werden können, die aus der Analyse einiger hundert Texte abgeleitet wurden. Derzeit sind 23 Relationen definiert (siehe Abbildung 5.1). Diese Relationen treten rekursiv zwischen immer kleineren benachbarten Textteilen auf, bis hinunter zur Satzteilebene. Die Beschreibung eines Gesamttextes mit Hilfe der RST-Relationen liefert eine hierarchische Struktur, deren Blätter den Satzteilen entsprechen. Nur Texte, für die eine solche Beschreibung gefunden werden konnte, gelten als kohärent.

Obgleich allgemein anerkannt wird, daß zwischen Sätzen, die kohärent einen Text bilden, Relationen bestehen, die diese Kohärenz vermitteln, ist die genaue Menge solcher Relationen und ihre Definition umstritten. Besonders [Grosz and Sidner, 1986] wenden sich dagegen, eine feste Menge zu definieren. In [Hovy, 1990a] wurde der Versuch unternommen, 350 Relationen von 25 verschiedenen Ansätzen in eine gemeinsame hierarchische Struktur zu ordnen, darunter auch die Relationen der RST. Es war möglich, drei Haupttypen zu identifizieren, sowie 16 Kernrelationen, die sich nicht auf die inhaltliche Bedeutung des Textes beziehen, sondern sich auf die rhetorischen Mittel beschränken, wie die Inhalte vermittelt werden. Je spezifischer jedoch die Relationen werden, d.h. je tiefer sie in der Hierarchie stehen, desto inhaltsbezogener werden sie. Die 16 Kernrelationen werden mit RST abgedeckt.

Eine RST-Relation besteht aus einem *Nukleus* und den *Satelliten*. Dem Nukleus entspricht der inhaltliche Schwerpunkt, der mitgeteilt werden soll. Satelliten geben zusätzliche Informationen, z.B. die Motivation oder Hintergründe zu dem Nukleusteil der Relation. Während ein Satellit weggelassen werden kann, ist er ohne den zugehörigen Nukleus nicht verständlich.

In Abbildung 5.2 ist die Definition der Relation EVIDENCE zu sehen. Sie umfaßt Bedingungen für den Nukleus (N), den Satelliten (S) und die Kombination der beiden, sowie den Effekt, den die Verwendung auf die Überzeugungen bewirkt. Auch die Anwendbarkeitsbedingungen hängen von den Intentionen der beiden Diskurspartner ab. In den folgenden zwei Beispielsätzen besteht eine solche EVIDENCE Relation (übersetzt aus [Mann and Thompson, 1987a, p.10]):

"Das Programm für das Kalenderjahr 1980 funktioniert wirklich. In wenigen

Name der Relation:	EVIDENCE
Bedingung für N:	Der Leser glaubt N nicht in einem für den Autor ausreichenden Maß.
Bedingung für S:	Der Leser glaubt S oder findet es glaubwürdig.
Bedingung für N + S:	Dadurch, daß der Leser S versteht, erhöht sich die Akzeptanz von N.
Effekt:	Der Leser ist in verstärkten Maß bereit, N zu glauben.
Ort des Effektes:	N

Abbildung 5.2: Die Definition der RST-Relation EVIDENCE

Minuten habe ich alle Zahlen meiner Steuererklärung eingetragen und habe genau die Ergebnisse erhalten, die ich manuell ausgerechnet habe."

Im ersten Satz wird eine Behauptung aufgestellt, die mit dem folgenden untermauert wird. Der Satellit in dem kurzen Text ist nur verständlich, wenn der Nukleus vorhanden ist. Umgekehrt ist der Nukleussatz für sich alleine zwar verständlich, er wird aber z.B. einen skeptischen Kunden nicht überzeugen.

Die deskriptive Version der RST bietet lediglich eine informelle Definition der Relationen und wurde dazu benutzt, Texte manuell zu *analysieren* und zu untersuchen, welche rhetorischen Strukturen in diesen vorhanden sind. Sie wurde jedoch so entwickelt, daß sie auch als Basis für einen Textplaner in einem Generierungssystem eingesetzt werden kann. In [Mann and Thompson, 1987b] werden informell Ideen für eine *konstruktive Version* der RST sowie das grobe Ablaufschema eines RST-Planers vorgestellt. Die Eingabe besteht aus dem Ziel des Autors und beschreibt den beabsichtigten Effekt des zu erzeugenden Texts auf den Leser. Dieser Effekt wird mit dem in der Definition der RST-Relationen verglichen, wobei die Bedingungen in der Definition als Vorbedingungen für die Anwendung dienen. Ausgabe des Planers ist eine baumförmige RST-Struktur, deren Blätter die Propositionen repräsentieren, aus denen der Text besteht.

5.1.2 Der RST-Strukturierer von Hovy

Ein System, das auf der Basis einer operationalisierten Version von RST arbeitet, wird in [Hovy, 1991] vorgestellt. Dem Strukturierer ist ein Ausschnitt einer Wissensquelle vorgegeben, dessen Inhalt mit Hilfe von RST-Operatoren so geordnet wird, daß ein kohärenter Text erzeugt werden kann. Die Operationalisierung folgt dem Vorschlag von Mann und Thompson, d.h. zu einem Ziel, das der Planer erfüllen soll, werden Operatoren gesucht, deren Vorbedingungen mit dem gegebenen Wissen erfüllt werden können. Nukleus und Satellit der Relation werden ebenfalls wieder als Ziele aufgefaßt. Der Planungsalgorithmus terminiert, wenn alle Teile der Wissensquelle konsumiert sind, oder keine anwendbaren Operatoren mehr gefunden werden.

Abbildung 5.3 zeigt einen Operator für die Relation SEQUENCE. Variablen sind mit

```
Name: SEQUENCE
Results:
  ((BMB SPEAKER HEARER (POSITION-OF ?PART ?NEXT)))
Nucleus+Satellite requirements/subgoals:
  ((BMB SPEAKER HEARER (NEXT-ACTION.R ?PART ?NEXT)))
Nucleus requirements/subgoals:
  ((BMB SPEAKER HEARER (TOPIC ?PART)))
Nucleus growth points:
  ((BMB SPEAKER HEARER (CIRCUMSTANCE-OF ?PART ?CIR))
   (BMB SPEAKER HEARER (ATTRIBUTE-OF ?PART ?VAL ))
   (BMB SPEAKER HEARER (PURPOSE-OF ?PART ?PURP)))
Satellite requirements/subgoals:
  ((BMB SPEAKER HEARER (TOPIC ?NEXT)))
Satellite growth points:
  ((BMB SPEAKER HEARER (ATTRIBUTE-OF ?NEXT ?VAL))
   (BMB SPEAKER HEARER (DETAILS-OF ?NEXT ?DETS))
   (BMB SPEAKER HEARER (POSITION-OF ?NEXT ?FOLL)))
```

Abbildung 5.3: Die Relation SEQUENCE in Hovys RST-Operationalisierung (Ausschnitt)

einem Fragezeichen präfigiert und werden bei der Instantiierung eines Operators an Elemente der Wissensquelle gebunden. Die Notation `(BMB SPEAKER HEARER (...))` bedeutet, daß ein Zustand erreicht werden soll, daß der Hörer glaubt, daß es die Intention des Sprechers ist, daß der Ausdruck, der in der Klammer steht, wahr ist.

Die Auswahl dessen, *was* gesagt werden soll, wird in dieser Operationalisierung getrennt von der Bestimmung der *Reihenfolge* (vgl. [Moore, 1989, p.67]). Der Prozeß der Auswahl des Wissens hat kein Wissen darüber zur Verfügung, ob das, was er auswählt, in einem Text resultiert, der die Intentionen überzeugend vermittelt und der kohärent ist. Da der Strukturierer z.B. kein zusätzliches Wissen erschließen kann, um ein neues Argument in den Text einzubauen, das den Hörer zu überzeugen vermag, ist gerade seine rhetorische Mächtigkeit eingeschränkt. Zusätzlich wird angenommen, daß das Wissen in einer Form vorliegt, in der sich von vorneherein (Teil–) Sätze identifizieren lassen.

Die Operatoren sind sehr domänenspezifisch und hängen unmittelbar von der inhaltlichen Struktur der Wissensquelle ab. Der gezeigte Operator arbeitet in einer Domäne, in der es um Schiffe der US-Navy, deren Positionen und Auftrag geht. Eine Übertragung in eine andere Domäne und/oder einen anderen Formalismus ist nicht ohne weiteres möglich. Da RST eine Theorie ist, die nicht über den Inhalt, sondern die Struktur eines Textes Aussagen macht, ist eine solch enge Bindung problematisch.

5.1.3 Der RST-Planer von Moore

RST als Basis für die *Auswahl* und *Strukturierung* eines Textes wurde in dem EES-System verwendet, das Erklärungen eines Expertensystems generiert [Moore, 1989]. Das System benutzt einen hierarchischen Planungsmechanismus [Sacerdoti, 1978], der als Eingabe ein

Formal:

```
EFFECT:       (MOTIVATION ?act ?goal)
CONSTRAINTS: (AND  (GOAL ?speaker ?goal)
                   (STEP ?act ?goal)
                   (GOAL ?hearer ?goal))
NUCLEUS:      (BEL ?hearer (STEP ?act ?goal))
SATELLITES:   nil
```

Natürlichsprachlich:

Um die Handlung *act* hinsichtlich *goal* zu motivieren,
WENN *act* ein Schritt ist, um *goal* zu erreichen
und der Hörer teilt *goal*
DANN überzeuge den Hörer, daß *act* ein Schritt ist,
um *goal* zu erreichen

Abbildung 5.4: Ein Planoperator für die Relation MOTIVATION

Diskursziel erhält, das mit der Generierung eines Textes erreicht werden soll.

Die Planoperatoren werden dazu benutzt, dasjenige Wissen aus den Wissensquellen des Systems zu bestimmen und kohärent zu ordnen, das verbalisert werden muß, um das Diskursziel der Eingabe zu erfüllen. Die Operatoren und deren Notation basieren auf den RST-Relationen und bestehen aus folgenden Teilen (vgl. [Moore, 1989, p.129] und Abbildung 5.4):

- einem *Effekt*, der das Ziel angibt, das mit diesem Operator erreicht werden soll. Er kann ein *intentionales Ziel* sein, z.B. daß ein Zustand erreicht werden soll, in dem der Hörer überzeugt ist, daß er eine bestimmte Handlung ausführen sollte. Der Effekt kann auch eine RST-Relation sein, z.B. die MOTIVATION für eine Handlung.

- einem *Bedingungsteil*, dessen Bedingungen erfüllt sein müssen, damit der Operator angewendet werden kann. Die Bedingungen können sich auf Wissensquellen wie z.B. das Systemwissen oder das Benutzermodell beziehen.

- einem *Nukleus*, den alle Operatoren besitzen müssen.

- *Satelliten*, die zusätzliche Information enthalten, die benötigt wird, um den Effekt des Operators zu erreichen.

Im Gegensatz zur ursprünglichen Definition von RST werden nicht verschiedene Bedingungen an den Nukleus und die Satelliten gestellt. Da die Planoperatoren dazu verwendet werden, Wissen aus einer Wissensquelle auszuwählen, mußte der Bedingungsteil enger gefaßt werden, um den Suchaufwand zu verringern.

Vorteilhaft an dieser Version der RST ist, daß die intentionalen Ziele und die rhetorischen Mittel, um sie kohärent zu verbalisieren, in einer einzigen Struktur repräsentiert und verarbeitet werden. So ist gewährleistet, daß die Inhalte nicht in einer Form ausgewählt werden, die eine kohärente Realisierung nicht zuläßt.

Der Planungsalgorithmus nimmt das Diskursziel der Eingabe als obersten Knoten in der Planungshierarchie und sucht einen Operator, dessen *effect* dem Ziel entspricht und dessen Bedingungen erfüllt werden können. Falls mehrere Operatoren anwendbar sind, werden Selektionsheuristiken angewendet. Diese berücksichtigen, in welchem Ausmaß verschiedene Wissensquellen wie z.B. das Benutzerwissen und das Dialoggedächtnis von einem Operator angesprochen werden. Bei der Instantiierung eines Operators werden für den Nukleus und die Satelliten neue Unterziel-Knoten in den Plan eingefügt. Der Planungsprozeß terminiert, wenn alle Unterziele primitive Sprechakte wie z.B. INFORM sind, die an das System PENMAN [Bateman *et al.*, 1989] zur Verbalisierung übergeben werden können.

Beim Aufbau des Textplans wird außer den Bedingungen des Planoperators ebenfalls geprüft, ob die Expansion mit einem Operator zu einem Plan führt, der eine kohärente RST-Struktur ergibt. Bezüglich des Plans bedeutet das, daß zwischen jeweils zwei Knoten, deren Effekte primitive Sprechakte sind, ein RST-Knoten im Plan vorhanden sein muß, wenn man diesen von links nach rechts und von oben nach unten durchläuft.

5.2 Die Auswahl des Inhalts in POPEL-WHAT

Die oben genannten RST-basierten Systeme haben gezeigt, daß RST eine gute Grundlage bietet, die auch bei der Inhaltsfestlegung in POPEL-WHAT genutzt werden soll.

Die Auswahl des relevanten Wissens folgt dem Ansatz der RST-Implementierung von Moore und arbeitet als System, das als Eingabe das *intentionale Ziel* erhält, das durch die Generierung eines Dialogbeitrags beim Hörer erreicht werden soll. Dazu muß der Inhalt aus der konzeptuellen Wissensquelle, der verbalisiert werden soll, ausgewählt und die Reihenfolge der einzelnen Teilstrukturen festgelegt werden. Als Basis dient die Implementierung von Moore, die entsprechend an die Bedürfnisse von POPEL angepaßt wurde. Erweiterungen waren nötig, da die Bandbreite der Äußerungen sich nicht auf Texte mit mehreren Sätzen beschränkt, sondern alle in Dialogen auftretenden Äußerungstypen umfaßt.

5.2.1 Die Notation der Ziele

Im Planer existieren drei verschiedene Arten von Zielen (vgl. [Moore, 1989, p.132]; dort werden nur die beiden erstgenannten explizit aufgeführt):

- *intentionale Ziele*, mit denen die Intentionen notiert werden, die mit einem Plan oder Planausschnitt verfolgt werden;
- *rhetorische Ziele*, die rhetorische Relationen zwischen den einzelnen Textteilen angeben;

- *primitive Ziele*, die die Schnittstelle zum Strukturaktivator bilden und diejenigen Teile der konzeptuellen Wissensquelle enthalten, die verbalisiert werden sollen.

Mit einem *intentionalen Ziel* soll der Wissensstand ausgedrückt werden, der nach der Verbalisierung eines Dialogbeitrags oder eines Teils davon beim Dialogpartner erreicht werden soll. Die Notation der intentionalen Ziele beruht auf der Struktur des Benutzermodells BGP-MS, das neben den Überzeugungen der Dialogpartner auch deren Ziele enthält. Ausgangspunkt für POPEL ist, wie in Abschnitt 3.2.5 dargestellt, die SWMB-Partition des Benutzermodells. Sie enthält alle diejenigen Teile der konzeptuellen Wissensquelle, von denen das System möchte, daß sie den Dialogpartnern gegenseitig bekannt sind, und ist damit Eingabe in POPEL.

Der Planer arbeitet mit einer linearisierten Notation der Ziele und nicht mit den Partitionen selbst. Ein intentionales Ziel wird als Liste geschrieben, deren erstes Element den Partitionsnamen des Ziels im Benutzermodell angibt. Die restliche Liste gibt an, welche Überzeugung bei wem und worüber erreicht werden soll. In der Steuerdomäne, in der die Intentionen sich auf den Austausch von Informationen beschränken, wurde nur modelliert, daß der Dialogpartner `H`[1] oder der Sprecher `S` etwas wissen sollen. Der Inhalt der Partition, die das konzeptuelle Wissen des Ziels enthält, wird als einzelnes Element oder als Liste dargestellt.

Beispiele für intentionale Ziele mit den Einträgen in der T- und A-Box aus Abbildung 5.5 sind[2]

```
(SWMB (KNOW H TAX-ACTION))
(SWMB (KNOW H (DRIVE BUY)))
(SWMB (KNOW S AGENS-TO))
```

Das erste Ziel kann POPEL als eine Erklärung des Konzepts oder als ein Beispiel für eine steuerlich absetzbare Handlung verbalisieren, das zweite als Text, der die beiden Aktionen verbalisiert, und das dritte als eine Frage an den Benutzer, ob der Mann gefahren ist.

Rhetorische Ziele bestehen aus einer rhetorischen Relation und denjenigen Teilen der konzeptuellen Wissensquelle, mit deren Verbalisierung das Ziel erreicht wird. Ein Beispiel für ein rhetorisches Ziel ist

```
(ELABORATION RETURN-R)
```

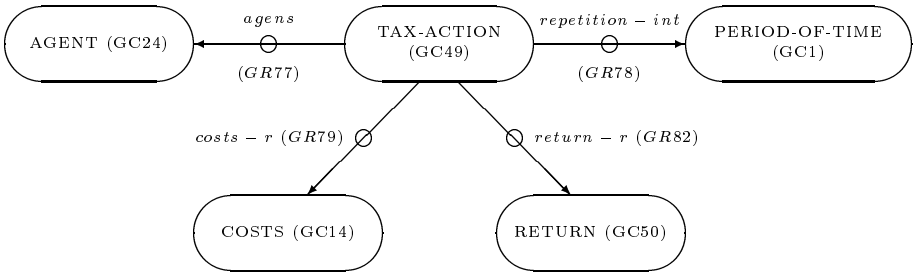
das bei der Generierung der Erklärung von `TAX-ACTION` als Unterziel auftreten kann. Dieses Ziel führt dazu, daß die Wissensquelle daraufhin untersucht wird, was zusätzlich zu dem Füller der Rolle ausgewählt werden soll.

Primitive Ziele bilden die Blätter des Planes. Sie enthalten Konzepte und Relationen zwischen Konzepten, die im Laufe der Planung zur Verbalisierung ausgewählt wurden und die an den Strukturaktivator übergeben werden. Zugleich legen sie den illokutionären Effekt fest. Es sind zwei Effekte definiert, nämlich `INFORM`, d.h. diese Teile sollen dem

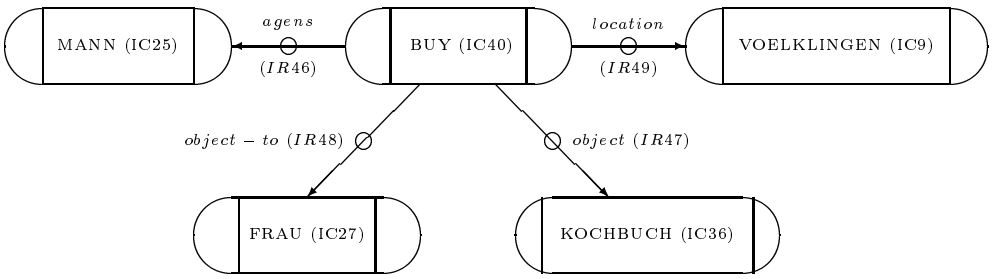
[1] `H` steht für *hearer*.

[2] Um die Lesbarkeit der Beispiele zu verbessern, werden statt der internen Bezeichner von SB-ONE deren Namen verwendet.

a) Die Definition einer steuerlich absetzbaren Handlung



b) Zwei Individualisierungen



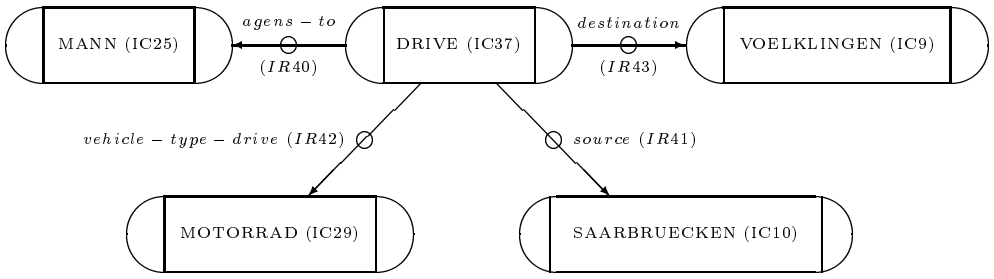


Abbildung 5.5: Beispieleinträge aus der T-Box (a) und A-Box (b, vereinfacht) der konzeptuellen Wissensquelle

Dialogpartner mitgeteilt werden, und `REQUEST`, d.h. danach soll gefragt werden. Welche Art von Frage, z.B. Entscheidungs– oder Ergänzungsfrage, wird erst im Strukturaktivator festgelegt. Ebenso wird die Entscheidung, was in einem Satz zusammengefaßt wird und wo Satzgrenzen liegen, erst im Strukturaktivator getroffen. Wie in Abschnitt 3.2.2.1 bereits gesagt wurde, ist es nicht möglich, aus der Struktur des konzeptuellen Wissens diese Entscheidung zu treffen.

Als Elemente der Wissensquelle enthält ein primitives Ziel Ausdrücke der CKB, wobei zwischen Objekten und Verbindungen zwischen diesen unterschieden wird. Objekte sind Konzepte der T- und A-Box, Verbindungen sind Rollen zwischen den Konzepten sowie drei zusätzliche Relationen: `ind` steht für die Individualisierungsbeziehung zwischen einem individualisierten und einem generellen Konzept, `super` für die Superkonzeptbeziehung, und `isa-supers` für einen Pfad zwischen einem individualisierten Konzept und dessen generellem Konzept oder einem seiner Superkonzepte. Verbindungsangaben müssen immer nach dem Konzept angegeben werden, von dem sie ausgehen. Nach einem Konzept können gleichzeitig mehrere Verbindungen in Listenform angegeben werden (vgl. die Syntaxdefinition in Abbildung 5.6).

Primitive Ziele beginnen immer mit dem Schlüsselwort `activate`, um zu signalisieren, daß sie das Interface zum Strukturaktivator herstellen. In POPEL ist nur ein Planoperator definiert, der dieses Ziel erfüllen kann. Beispiele für primitive Ziele sind:

```
(activate (INFORM TAX-ACTION AGENS AGENT))
(activate (REQUEST DRIVE AGENS-TO MANN))
(activate (INFORM DRIVE (AGENS-TO VEHICLE-TYPE-DRIVE SOURCE DESTINATION)))
```

Das erste Ziel kann als Aussage

Eine Steuerhandlung verursacht Kosten.

das zweite als Frage

Fährt der Mann?

realisiert werden. Bei *request*-Zielen muß das Konzept oder die Rolle, die der Benutzer dem System mitteilen soll, nach dem Effekt `REQUEST` stehen. Das dritte Ziel besagt, daß `DRIVE` verbalisiert werden soll und die Füller der angegebenen Rollen als dessen Kontext im weiteren Planungsprozeß ebenfalls verbalisiert werden könnten.

Im Gegensatz zu der Zielnotation von Moore [Moore, 1989, p.125ff] gibt es in POPEL keine Repräsentationsprimitive wie `GOAL`, `COMPETENT` oder `PERSUADED`, die sich auf Absichten oder Möglichkeiten des Dialogpartners beziehen. Der Planer arbeitet nur über den Strukturen des konzeptuellen Wissens sowie des Benutzermodells. In diesem Benutzermodell, und nicht im Planer, müssen Kompetenzen und Ziele des Benutzers modelliert werden, da dieses Wissen bei der Verarbeitung im Gesamtsystem benötigt wird, und nicht nur bei der Generierung.

5.2.2 Die Syntax der Planoperatoren

In Abbildung 5.6 ist die Syntax der Planoperatoren in einer BNF-Notation definiert. Die in Schreibmaschinenschrift angegebenen Symbole stehen für Elemente des jeweiligen Typs,

$$
\begin{array}{lll}
planoperator & ::== & \text{:NAME} \quad \texttt{Zeichenkette} \\
 & & \text{:EFFECT} \quad zielausdruck \\
 & & \text{:CONSTRAINT} \quad \texttt{HC2lC-Prädikat} \\
 & & \text{:NUCLEUS} \quad zielausdruck \\
 & & \text{:SATELLITES} \quad (\ sat - spezifikation^{0..n}\ ) \\
sat - spezifikation & ::== & (sat - zielausdruck\ \texttt{HC2lC-Prädikat}) \\
sat - zielausdruck & ::== & schleifenausdruck \mid zielausdruck \\
schleifenausdruck & ::== & (\text{FORALL}\ \texttt{Variablenliste}\ zielausdruck) \\
zielausdruck & ::== & intentionales - ziel \mid rhetorisches - ziel \mid primitives - ziel \\
intentionales - ziel & ::== & (\texttt{UM-Partition}\ (\text{KNOW}\ adressat\ sb - one - id^{1..n})) \\
adressat & ::== & \text{S} \mid \text{H} \\
rhetorisches - ziel & ::== & (rst - relation\ sb - one - id^{1..n}) \\
rst - relation & ::== & \text{CIRCUMSTANCE} \mid \text{ANTITHESIS} \mid \text{SOLUTIONHOOD} \mid \\
 & & \text{CONCESSION} \mid \text{ELABORATION} \mid \text{CONDITION} \mid \\
 & & \text{OTHERWISE} \mid \text{MOTIVATION} \mid \text{INTERPRETATION} \mid \\
 & & \text{EVIDENCE} \mid \text{EVALUATION} \mid \text{JUSTIFY} \mid \\
 & & \text{VOLITIONAL CAUSE} \mid \text{NON-VOLITIONAL CAUSE} \mid \\
 & & \text{VOLITIONAL RESULT} \mid \text{NON-VOLITIONAL RESULT} \mid \\
 & & \text{RESTATEMENT} \mid \text{SUMMARY} \mid \text{SEQUENCE} \mid \\
 & & \text{BACKGROUND} \mid \text{CONTRAST} \mid \text{PURPOSE} \mid \\
 & & \text{ENABLEMENT} \\
primitives - ziel & ::== & (\text{ACTIVATE}\ sprechakt) \\
sprechakt & ::== & (\text{INFORM}\ \texttt{SB-ONE-Konzept}\ sb - one - relation^{2..n}) \mid \\
 & & (\text{INFORM}\ \texttt{SB-ONE-Konzept}\ sb - one - relation \\
 & & \qquad \texttt{SB-ONE-Konzept}) \mid \\
 & & (\text{REQUEST}\ sb - one - element\ sb - one - element \\
 & & \qquad sb - one - element) \\
sb - one - relation & ::== & \texttt{SB-ONE-Rolle} \mid \text{IND} \mid \text{SUPER} \mid \text{ISA-SUPERS} \\
sb - one - element & ::== & \texttt{SB-ONE-Konzept} \mid sb - one - relation \\
sb - one - id & ::== & \texttt{SB-ONE-Konzept} \mid \texttt{SB-ONE-Rolle}
\end{array}
$$

Abbildung 5.6: Die Definition der Operatorensyntax

```
(define-text-plan-operator
  :NAME describe-action-concept
  :EFFECT (SWMB (KNOW H ?action-concept))
  :CONSTRAINTS (p-and (action-concept-test ?action-concept
                                           ?agent-role
                                           ?remaining-roles)
                      (role ?agent-role ?action-concept ?agent))
  :NUCLEUS (ACTIVATE (INFORM ?action-concept ?agent-role ?agent))
  :SATELLITES (((SWMB (KNOW H ?agent)) (*optional*))
               ((FORALL ?remaining-roles
                 (SWMB (KNOW H ?remaining-roles))))))
```

Abbildung 5.7: Ein Planoperator für die Erklärung eines Konzepts

Variablen, an die Elemente dieses Typs gebunden werden, oder HC2LC-Prädikate, mit denen diese Variablen gebunden werden. Sonstige terminale Symbole sind groß geschrieben, nichtterminale klein und kursiv.

Ein Planoperator besteht aus einem Namen, einem Effekt, einem Bedingungsteil, und der Angabe der Unterziele des Nukleus und der Satelliten. Die Bedingungen werden mit HC2LC-Prädikaten formuliert, wobei der Zugriff auf die in SB-ONE aufgebauten Wissensquellen über $\mathcal{L}_{\mathcal{SB}-\mathcal{ONE}^+}$ möglich ist. Über selbstdefinierte Prädikate können auch andere Wissensquellen verwendet werden.

Im RST-Planer von POPEL wurden die Bedingungen für Satelliten beibehalten, wie sie in der deskriptiven Version der RST und bei Hovy vorhanden sind. Auf gesonderte Bedingungen für den Nukleus wurde verzichtet. Diese sind Teil der Bedingungen, die für den gesamten Operator gelten. Da angenommen wird, daß der Nukleus den inhaltlichen Schwerpunkt des Textabschnitts trägt, sind sie ein Teil der Gesamtbedingungen. Der `SATELLITES`–Slots eines Planoperators besteht aus einer Liste von Zielausdrücken, die wiederum aus dem eigentlichen Ziel und der in $\mathcal{L}_{\mathcal{SB}-\mathcal{ONE}^+}$ formulierten Bedingung besteht. Bei der Instantiierung eines Planoperators wird ein Unterziel im `SATELLITES`–Teil nur dann in den Plan als neues Unterziel hinzugenommen, wenn diese Bedingung wahr ist oder keine Bedingung vorhanden ist.

Abbildung 5.7 zeigt einen Beispielsoperator, dessen Effekt das intentionale Ziel ist, daß der Dialogpartner ein Aktionskonzept aus der T-Box des konzeptuellen Wissens kennt. Das Prädikat `action-concept-test` überprüft, ob das an die Variable `?action-concept` gebundene Konzept ein Aktionskonzept ist. Als Nukleus des Erklärungstextes wird die `?agent`–Rolle ausgewählt, d.h. das Wissen, daß eine Handlung von einer Person ausgeführt werden muß. Als Information zu diesem Nukleus wird zunächst untersucht, ob dem Dialogpartner zusätzlich mehr über den `?agent` mitgeteilt werden muß. Dieses Ziel ist `*optional*` und kann, falls keine Planexpansion gefunden wird, ignoriert werden (siehe den nächsten Abschnitt). Der zweite Ausdruck in der Liste der Satelliten ist der spezielle Ausdruck `FORALL`, der Iterationen über Variablen erlaubt, an die mehrere Werte gebunden sind. Für jeden Wert der Variablen wird ein Unterziel in den Plan eingetragen.

```
 (1)   *subgoals-to-achieve* := list (*text-plan-top-node*)
 (2)   while *subgoals-to-achieve*
 (3)         subgoal := first (*subgoals-to-achieve*)
 (4)         if    plan-node-untried-alternatives (subgoal) = NIL
 (5)         then find-candidate-plan-operators (subgoal)
 (6)         endif
 (7)         if    plan-node-plan-operator (subgoal)
 (8)         then instantiate-plan-operator (subgoal)
 (9)         else
(10)              selected-operator := select-plan-operator (subgoal)
(11)              if    selected-operator = NIL
(12)              then
(13)                    if     subgoal = *text-plan-top-node*
(14)                    then  return FAIL
(15)                    elseif not optional (subgoal)
(16)                    then  re-plan-node (plan-node-superior (subgoal))
(17)                    endif
(18)              else
(19)                    if    not primitive (subgoal)
(20)                    then instantiate-plan-operator (subgoal)
(21)                    endif
(22)              endif
(23)        endif
(24)  endwhile
(25)  return SUCCESS
```

Abbildung 5.8: Der Algorithmus des Planers

5.2.3 Die Verarbeitung der Operatoren

5.2.3.1 Der Ablauf des Planers

Die Verarbeitung der Planoperatoren wird von einem einfachen hierarchischen Planer durchgeführt (vgl. NOAH [Sacerdoti, 1978], jedoch ohne *critics*). Abbildung 5.8 zeigt den Programmablauf in einer vereinfachten Form.

Das intentionale Ziel, das der Planer als Eingabe erhält, wird in der Variablen `*text-plan-top-node*` gespeichert (1)[3]. Solange es Unterziele gibt, die noch nicht erfüllt wurden, arbeitet der Planer (2). Falls zu dem offenen Unterziel, das gerade bearbeitet wird, noch keine Planoperatoren ausgewählt wurden, werden zunächst alle Operatoren, deren `effect` dem Ziel entspricht, aus der Operatorendatenbasis gesucht (3-6). Ist zu dem Ziel bereits ein Operator bestimmt worden, dann wurde es bereits einmal bearbeitet, jedoch konnten keine Operatoren für eines der Unterziele gefunden werden. Wenn es alternative Variablenbelegungen gibt, wurde der Planknoten mit dieser Belegung erneut in die Liste der zu erreichenden Unterziele aufgenommen (7-8). Ist noch kein Operator zu dem Ziel

[3]Die Zahlen in Klammern beziehen sich auf die Zeilennummern in Abbildung 5.8.

```
(define-text-plan-operator
  :NAME describe-concept-with-instance
  :EFFECT (SWMB (KNOW H ?concept))
  :CONSTRAINTS  (p-and (isa-supers ?ic ?concept)
                       (UM (SBUB ?ic)))
  :NUCLEUS (activate (INFORM ?ic isa-supers ?concept))
  :SATELLITES nil)
```

Abbildung 5.9: Ein Planoperator für eine Erklärung durch eine Beispielsinstanz

bestimmt, wird aus der Menge der möglichen Operatoren einer ausgewählt (10). Dieses erfolgt unter Verwendung von Heuristiken, die in Abschnitt 5.2.3.2 behandelt werden. Wenn kein Operator ausgewählt wurde, und das Ziel das Eingabeziel ist, kann kein kohärenter Text für dieses Ziel erzeugt werden (13-14). Ist das Ziel optional, kann es ignoriert werden, wenn dafür keine Planexpansion gefunden werden kann (15). Ansonsten wird in `re-plan-node` der übergeordnete Planknoten für dieses Ziel mit neuen Variablenbindungen oder einem anderen Planoperator wieder in die Liste der zu expandierenden Ziele aufgenommen (16). Ist ein Planoperator anwendbar und nicht primitiv, d.h. kein Blatt des Planes, wird der Planoperator instantiiert, die Variablen werden gebunden und für Nukleus und Satelliten neue Ziele in den Plan aufgenommen (20).

Die Reihenfolge, in der die Unterziele in den Plan aufgenommen werden, hängt ab von dem Ziel des Operators. Ist es ein intentionales, wird der Nukleus links von den Satelliten eingefügt und damit auch vor den Satelliten verbalisiert. Ist der Effekt eine RST-Relation, hängt die Reihenfolge von der sog. *canonical order* der Relation ab [Mann and Thompson, 1987a, p.16f]. Bei Operatoren für *Antithesis* z.B. kommen die Satelliten vor dem Nukleus. Da der Planer von links nach rechts und zuerst in die Tiefe arbeitet, wird der Inhalt auch in der Reihenfolge, in der er geäußert werden soll, festgelegt und kann inkrementell weiter verarbeitet werden. Die Eingabe für den Strukturaktivator besteht in den primitiven Zielen des Planes, die, sobald sie festgelegt wurden, weitergegeben werden.

Die inkrementelle Verarbeitung hat zur Folge, daß der Planer keinen vollständigen Plan erzeugt, der dann an die Realisierungskomponente POPEL-HOW übergeben wird. Da damit die Realisierung und die daraus resultierende Veränderung der Wissensbasis des XTRA-Systems parallel zur Auswahl abläuft, ändern sich die Anwendbarkeitsbedingungen für die Planoperatoren noch während der Planungsphase. Bei der Planung eines Teiles eines Dialogbeitrags werden also unmittelbar bereits diejenigen Teile berücksichtigt, die in dem gleichen Beitrag ausgewählt wurden, jedoch bereits realisiert sind.

Die parallele Implementation basiert auf der objektorientierten Simulation eines Parallelprozessorsystems, mit dem auch das gesamte restliche System realisiert wurde. Es wird im Abschnitt 8.1 (Seite 151) beschrieben.

5.2.3.2 Die Auswahl der Operatoren

Es gibt verschiedene Reaktionsmöglichkeiten für einen Generator, um ein intentionales Ziel zu erreichen. Beispielsweise kann alternativ zur Verbalisierung des terminologischen Wissens, wie es der Operator aus Abbildung 5.7 durchführt, auch eine Beispielsinstanz verbalisiert werden (vgl. Abbildung 5.9). An diesem Operator ist auch zu sehen, daß der Planer, wenn kein Text, sondern nur ein Einzelsatz erzeugt wird, keine Textstruktur gemäß der RST aufbauen muß. Die Operationalisierung der RST in POPEL erlaubt demnach die Flexiblität, die in der Spezifikation gefordert wurde.

Das Prädikat `isa-supers` im Beispiel bindet an die Variable `?ic` ein instantiiertes Konzept des generellen Konzepts `?concept` oder von Subkonzepten dieses Konzepts.[4] Das Prädikat `UM` greift auf das Benutzermodell zu und testet, ob `?ic` in der Partition `SBUB` steht, d.h. ob das System glaubt, daß der Benutzer das individualisierte Konzept kennt.

Der Planoperator `describe-concept-with-instance` verwendet kontextuelles Wissen, um das Ziel zu erreichen, im Gegensatz zu dem Planoperator `describe-action-concept`, der die T-Box Definition verbalisiert. Wenn der Planer beide Operatoren verwenden kann, erhöht sich die Flexibilität erheblich, mit der er reagieren kann. Es tritt jedoch das Problem auf, daß entschieden werden muß, wann welcher Operator zum Einsatz kommt. Daß diese Entscheidung notwendig ist, ist ein Folge des *Qualifikationsproblems* ('qualification problem'), das darin besteht, daß man bei den Operatoren nicht alle Bedingungen angeben kann und will, bei denen der Operator *nicht* anwendbar ist (vgl. z.B. [Hertzberg, 1988, p.27ff]). Eine vollständige Definition ist beispielsweise nur dann erwünscht, wenn in den Kontextwissensquellen keine dem Dialogpartner bekannte Beispielsinstanz gefunden werden kann, oder wenn er eine explizite Definition wünscht.

Moore führt zur Auswahl der möglichen Operatoren Heuristiken ein, die u.a. berücksichtigen, welche Annahmen über das Benutzerwissen in einem Operator getroffen werden und ob bereits bekannte Konzepte verwendet werden. Mit dem Einsatz dieser Heuristik wird angenommen, daß das System kooperativ und möglichst benutzerangepaßt arbeiten soll. In einem interessensbasierten System, das andere Ziele verfolgt, z.B. das, möglichst viel Steuern zu erhalten, dürfte diese Heuristik nicht verwendet werden.

Der Ansatz von Moore wurde übernommen, jedoch nur derjenige Teil realisiert, der Zugriffe auf das Benutzermodell in Erwägung zieht. BGP-MS ist mächtiger als das Benutzermodell im System von Moore und deckt Teile des Wissens ab, das sich dort in anderen Wissensquellen befindet.

Der Zugriff zum Benutzermodell erfolgt mit dem Prädikat `UM`, wobei jeder Zugriff gezählt wird und bei dem Operator in jeder alternativen Bindungsmöglichkeit in der Variablen `UM-ACCESS` gespeichert wird. Die Zugriffshäufigkeit wird für alle Operatoren eines Planknotens auf das Intervall 0 bis 1 normiert, wobei der Operator mit den meisten Zugriffen den Wert 1 erhält. Die Operatoren werden dann nach ihrer Bewertung sortiert und in dieser Reihenfolge verwendet.

Die Arbeitsweise der Heuristik soll anhand des Planknotens in Abbildung 5.10 darge-

[4] Genaugenommen müßte dieses Prädikat überprüfen, ob `?concept` ein *Basic Concept* des generellen Konzepts von `?ic` ist (siehe z.B. [Lakoff, 1987]).

```
#S(PLAN-NODE GOAL (SWMB (KNOW H TAX-ACTION))
      PLAN-OPERATOR
      #S(PLAN-OPERATOR NAME DESCRIBE-CONCEPT-WITH-INSTANCE
         EFFECT (SWMB (KNOW H ?CONCEPT))
         CONSTRAINTS (P-AND (ISA-SUPERS ?IC ?CONCEPT) (UM (SBUB ?IC)))
         NUCLEUS (ACTIVATE (INFORM ?IC ISA-SUPERS ?CONCEPT))
         BINDINGS
         (#S(VARIABLE-BINDING VARIABLE-NAME UM-ACCESS
             VARIABLE-VALUE 8 ASSUMPTIONS NIL)
          #S(VARIABLE-BINDING VARIABLE-NAME ?IC
             VARIABLE-VALUE COMMUTE-WITH-PUBLIC-TRANSPO ASSUMPTIONS NIL)
          #S(VARIABLE-BINDING VARIABLE-NAME ?CONCEPT
             VARIABLE-VALUE TAX-ACTION ASSUMPTIONS NIL))
         UNTRIED-BINDINGS NIL
         SELECTION-HEURISTICS-ALIST ((USER-MODEL-SCORE 1)) SCORE 1)
      TRIED-ALTERNATIVES NIL
      UNTRIED-ALTERNATIVES
      (#S(PLAN-OPERATOR NAME DESCRIBE-ACTION-CONCEPT-WITHOUT-KNOWLEDGE
         EFFECT (SWMB (KNOW H ?ACTION-CONCEPT))
         CONSTRAINTS (P-AND (ACTION-CONCEPT-TEST
                               ?ACTION-CONCEPT ?AGENT-ROLE ?REMAINING-ROLES)
                            (ROLE ?AGENT-ROLE ?ACTION-CONCEPT ?AGENT))
         NUCLEUS
         (ACTIVATE (INFORM ?ACTION-CONCEPT ?AGENT-ROLE ?AGENT))
         SATELLITES
         (((ELABORATION ?AGENT) (*OPTIONAL*))
          ((FORALL ?REMAINING-ROLES
                   (SWMB (KNOW H ?REMAINING-ROLES)))))
         BINDINGS
         (#S(VARIABLE-BINDING VARIABLE-NAME UM-ACCESS
             VARIABLE-VALUE 0 ASSUMPTIONS NIL)
          #S(VARIABLE-BINDING VARIABLE-NAME ?AGENT
             VARIABLE-VALUE AGENT ASSUMPTIONS NIL)
          #S(VARIABLE-BINDING VARIABLE-NAME ?REMAINING-ROLES
             VARIABLE-VALUE (COSTS-R RETURN-R REPETITION-INT)
             ASSUMPTIONS NIL)
          #S(VARIABLE-BINDING VARIABLE-NAME ?AGENT-ROLE
             VARIABLE-VALUE AGENS ASSUMPTIONS NIL)
          #S(VARIABLE-BINDING VARIABLE-NAME ?ACTION-CONCEPT
             VARIABLE-VALUE TAX-ACTION ASSUMPTIONS NIL))
         UNTRIED-BINDINGS NIL
         SELECTION-HEURISTICS-ALIST ((USER-MODEL-SCORE 0)) SCORE 0.0))
      ASSUMPTIONS NIL STATUS ACTIVE OPTIONAL NIL)
```

Abbildung 5.10: Ein Planknoten mit zwei Operatoren und deren Bewertungen

stellt werden. Wenn der Planer das intentionale Ziel `(SWMB (KNOW H TAX-ACTION))` als Eingabe erhält, und in der Operatorendatenbasis die Operatoren aus den Abbildungen 5.7 und 5.9 stehen, wird der gezeigte Planknoten erzeugt. In der A-Box ist das individualisierte Konzept `COMMUTE-WITH-PUBLIC-TRANSPO` vorhanden, das den Bedingungsteil des Operators `DESCRIBE-CONCEPT-WITH-INSTANCE` erfüllt und auch dem Benutzer bekannt ist. Bei der Auswertung des Bedingungsteils mußte achtmal auf das Benutzermodell zugegriffen werden, während bei dem alternativ möglichen Operator kein Zugriff erfolgte. Deshalb wird das System z.B.

Diese Fahrt ist eine Steuerhandlung.

generieren, wobei die genaue Form der Nominalphrasen von kontextuellen Bedingungen abhängt, die bei der Festlegung der Referenzausdrücke ausgewertet werden.

5.3 Die Schnittstelle zur Realisierung

An der Schnittstelle zwischen dem Auswahlprozeß und der Realisierung steht der *Strukturaktivator* (siehe auch Abschnitt 3.4.2.2). Nachdem die Auswahl der Teile erfolgt ist, die verbalisiert werden sollen, übernimmt der Strukturaktivator teilweise die Aufgaben, die in Levelts Modell als Mikroplanung bezeichnet werden [Levelt, 1989, p.144ff]. Die aus dem Planer kommende Information wird dahingehend untersucht, welche Teile zusammen als Satz realisiert werden können und es wird entschieden, wie einzelne Teile zusammengefaßt werden können. Daneben wird noch der Typ der Äußerung, z.B. Frage– oder Aussagesatz, festgelegt.

Ebenfalls in dieser Komponente müßten *Tempus* und *Modus* bestimmt werden (vgl. z.B. [Ehrich, 1987]). Da in der zur Verfügung stehenden Repräsentationssprache temporale Relationen nicht adäquat modelliert werden können, wurden in dieser Richtung keine weiteren Untersuchungen und Implementationsarbeiten durchgeführt. Das gleiche gilt für die Festlegung des Modus. Versuchsweise wird derzeit das Modalverb "können" und der Konjunktiv verwendet, wenn eine generelle Rolle der T-Box verbalisiert wird, die optional ist.

5.3.1 Die Festlegung von Satzstrukturen

Der Auswahlprozeß arbeitet lediglich über den Konzepten der CKB und Relationen zwischen den Konzepten, ohne daß entschieden werden kann, welche Strukturen zusammen als Satz realisiert werden. Viele andere Generierungssysteme gehen im Gegensatz dazu davon aus, daß die Strukturierung des Wissens verbzentriert ist. Der Planer von Hovy [Hovy, 1991] z.B. erwartet, daß die Eingabe in einzelnen Propositionen vorliegt, die in jeweils einem Satz geäußert werden können. Ist der Inhalt einmal nach Sätzen gegliedert, stellt sich weiterhin das Problem, wie diese strukturiert werden sollen, und was als Haupt– und Nebensatz zusammengefaßt bzw. was besser in mehreren Sätzen realisiert werden soll. Diese Fragestellung ist eines der Schwerpunktthemen im Zusammenhang mit RST-basierter Generierung (vgl. [Hovy, 1990c, Scott and de Souza, 1990]) und wurde in POPEL für einige Fälle untersucht.

5.3.1.1 Die Bestimmung der Satzstruktur

Mit den Beispieleinträgen aus der CKB in Abbildung 5.5 läßt sich zeigen, wie diese realisiert werden, und welche Strukturen sich dabei ergeben. Bei der Verbalisierung des T-Box Eintrags können die Rollenbeziehungen das Prädikat eines Satzes bilden, z.B. die Rolle `agens` des Konzepts `TAX-ACTION` in dem Satz

Eine Person führt eine Steuerhandlung aus.

Im Gegensatz dazu kann das individualisierte Konzept `BUY` als Verb realisiert werden, z.B. zusammen mit den Rollen `agens` und `object` als Satz

Der Mann kauft ein Kochbuch.

Auch bei der Verbalisierung des generellen Konzepts von BUY kann dieses als Prädikat realisiert werden, obgleich `TAX-ACTION` eines seiner Superkonzepte ist. Alleine aus der Struktur der CKB läßt sich also die Entscheidung nicht treffen, was als Satz realisiert wird und was nicht.

Der Strukturaktivator benötigt zur Entscheidung, wie die Gruppierung in Teilsätze erfolgt, das Wissen, wie die CKB-Elemente in der FSS realisiert werden. Wie oben gesagt wurde, werden die aus der CKB ausgewählten Teile inkrementell an den Strukturaktivator weitergegeben. Dieser muß sie ebenfalls inkrementell der Realisierung übergeben, und dann auf die entstehenden FSS-Strukturen zugreifen, damit ihm diese Information zur Verfügung steht. Ist für ein Element der CKB die Realisierung als *Prädikat* in der FSS erfolgt, kann dieses Element als das verbale Zentrum angesehen werden, das zusammen mit weiteren CKB-Elementen einen Satz bildet. Dieser Satz wird gebildet unter Hinzufügung solcher CKB-Elemente, deren Realisierungen auf der FSS-Ebene Füller einer Rolle der FSS-Struktur des Prädikats sind, und die in der RST-Struktur zusätzliche Information zu einer Rolle oder einem Rollenfüller des CKB-Elements geben, das als Zentrum eines Satzes identifiziert wurde. Die zusätzliche Information wird in der RST-Struktur durch die Relation ELABORATION geliefert.

Z.B. wird bei der Generierung des Ziels

```
(SWMB (KNOW H DRIVE))
```

folgender Plan aufgebaut:

```
Nukleus: (ACTIVATE (INFORM DRIVE
                     (AGENS-TO VEHICLE-TYPE-DRIVE SOURCE DESTINATION)))
Satelliten:
    (SWMB (KNOW H AGENS-TO))
      Nukleus: (ELABORATION MANN)
                  Nukleus: (ACTIVATE (INFORM MANN))
     (SWMB (KNOW H VEHICLE-TYPE-DRIVE))
      Nukleus: (ELABORATION MOTORRAD)
                  Nukleus: (ACTIVATE (INFORM MOTORRAD))
     (SWMB (KNOW H SOURCE))
      Nukleus: (ELABORATION SAARBRUECKEN)
```

```
                Nukleus: (ACTIVATE (INFORM SAARBRUECKEN))
(SWMB (KNOW H DESTINATION))
 Nukleus: (ELABORATION VOELKLINGEN)
                Nukleus: (ACTIVATE (INFORM VOELKLINGEN))
```

`DRIVE` wird als Prädikat in die FSS-Ebene abgebildet, dessen Argumente die Abbildungen der Rollenfüller aus der CKB sind. Verbalisiert werden kann das Konzept z.B. als Verb des Satzes

Der Mann fährt mit dem Motorrad von Saarbrücken nach Völklingen.

Ein ganz anderes Problem tritt auf, wenn sich in den ausgewählten CKB-Elementen keines befindet, das auf ein Prädikat abgebildet werden kann. In dem Dialogausschnitt

BEN: *Wer fährt mit dem Motorrad?.*
SYS: *Der Mann.*

wählt POPEL als Antwort nur denjenigen Inhalt aus, von dem angenommen wird, daß er dem Benutzer unbekannt ist, nämlich das individualisierte Konzept `MANN`. In Fällen wie diesen, in denen zu CKB-Elementen, die zu verbalisieren sind, kein Prädikat in der FSS vorhanden ist, versucht der Strukturaktivator *nicht*, dieses zu ergänzen und beispielsweise ein zusätzliches CKB-Element auszuwählen, das auf ein Prädikat abgebildet wird.

Die Untersuchung auf Satzstrukturen erfolgt nur, damit eine Strukturierung der Äußerung durchgeführt werden kann. Wenn die Zuordnung von CKB-Elementen zu einem Prädikat nicht möglich ist, werden diese ebenfalls an POPEL-HOW übergeben. Wird dort festgestellt, daß Informationen fehlen, so werden Anfragen gestellt, die bei der Behandlung zur Suche der fehlenden Information z.B. in den Kontextwissensquellen oder in der CKB führen. POPEL kann damit Äußerungen im Sinne von Engel [Engel, 1988, p.179] erzeugen, die einen kommunikativen Zweck erfüllen, aber nicht unbedingt grammatisch vollständig sein müssen. In dem obigen Beispiel wird der fehlende Kasus aus der satzsemantischen Struktur des vorangehenden Satzes erschlossen (siehe auch Abschnitt 5.5).

5.3.1.2 Die Festlegung von Einbettungen

Damit liegt die Strukturierung in einzelne Sätze vor. Diese sollten aber ineinander eingebettet werden, wenn es zu einem besser strukturierten und kohärenteren Text führt. Die Verbalisierung der Rollen `costs-r`, `return-r` und der jeweiligen dominierenden Konzepte und der Rollenfüller aus Abbildung 5.5 als

Die Steuerhandlung verursacht Kosten, die abgesetzt werden können.

ist beispielsweise besser verständlich als zwei Einzelsätze. Die Einbettung kann hier erfolgen, weil die FSS-Realisierungen von `COSTS` und `RETURN` identisch sind. Der zweite Satz kann also in den ersten eingebettet werden.

Die Strukturierung des Textes in einzelne Sätze und deren Organisation gehört zu den "sieben ungelösten Problemen" RST-basierter Textplanung, die in [Hovy, 1990c] angesprochen werden. In [Scott and de Souza, 1990, p.56ff] werden Heuristiken vorgestellt,

die helfen, Einbettungen von Sätzen ineinander zu bestimmen. Die Autorinnen stellen die These auf, daß Einbettungen eines Satzes in einen anderen nur dann möglich sind, wenn diese mit einer `ELABORATION` Relation verbunden sind.

Diese Heuristik wird modifiziert ebenfalls im Strukturaktivator angewendet. Stehen zwei aufeinanderfolgende Sätze in `ELABORATION`-Relation, oder sind beide eine `ELABORATION` eines gemeinsamen Nukleus, dann werden in den CKB-Elementen beider Sätze solche Elemente gesucht, die in der FSS nominal mit dem gleichen Konzept realisiert werden.

Das CKB-Element des zweiten Satzes erhält dann bei der Übergabe in die Realisierungskomponente eine Markierung, die besagt, daß dieser Satz eingebettet werden soll, falls aus syntaktisch-semantischer Sicht nichts dagegen spricht. Ebenso wird bei der Realisierung des CKB-Elements, das den Anschluß zwischen beiden Sätzen herstellt, die Information übergeben, daß es für die Einbettung verantwortlich ist. Die Prozesse der Realisierung werten diese Informationen aus und können, falls möglich, damit z.B. in der syntaktischen Realisierung einen Relativsatz und das Relativpronomen festlegen.

5.3.2 Die Schnittstellenfunktionen

Bei dem Überblick über den Ablauf der Verarbeitung in POPEL in Abschnitt 3.4.3 wurde der Begriff der charakteristischen Eingabe jeder Verarbeitungsstufe genannt, die festlegt, mit welcher Granularität die Parallelverarbeitung in einer Stufe durchgeführt werden kann. An der Schnittstelle zu POPEL-HOW legt der Strukturaktivator diese Granularität für die erste Stufe der Realisierung fest.

Es wurde gezeigt, daß der Strukturaktivator bei der Strukturierung der primitiven Ziele des RST-Plans in Sätze bereits während der Bearbeitung eines Satzes die Zwischenergebnisse aus der Realisierung benötigt, um feststellen zu können, welche CKB-Strukturen zusammen einen (Teil-)Satz bilden können. Deshalb muß die Realisierung eines Elements der CKB erfolgen, sobald es ausgewählt wurde und es aus der Sicht des Strukturaktivators zu dem derzeit bearbeiteten Satz hinzugefügt werden kann.

In den primitiven Zielen sind neben der Angabe von Illokutionen nur Elemente der epistemologischen Beschreibungsebene von SB-ONE enthalten. Diese bilden die charakteristische Eingabe in die Realisierung und legen fest, welches die kleinsten Einheiten sind, die an POPEL-HOW übergeben werden können. Eine andere Strukturierung ist nicht möglich: weder können kleinere Segmente als die SB-ONE Strukturen gefunden werden, noch verfügt der Strukturaktivator in seinen Wissensquellen über Daten, die eine Strukturierung in größere Einheiten erlauben würden.

Die Elemente in den Zielen bestehen aus generellen und individualisierten Konzepten, sowie aus Relationen zwischen ihnen. Dementsprechend bilden zwei Funktionen die Schnittstelle zur Realisierung. Die Funktion

`(verbalize-ckb-object` *concept specials* `)`

initiiert die Realisierung von *concept*, wobei in *specials* zusätzliche Daten mitgegeben werden, die z.B. Einbettungs– oder Satztypinformation enthalten. Der Rückgabewert der

```
(1)   repeat
(2)         repeat
(3)               /* Phase 1 */
(4)               loop for primitive-goal in new-objects
(5)               do   prepare-one (primitive-goal)
(6)               endloop
(7)               /* Phase 2 */
(8)               loop for object in objects-to-activate
(9)               do   activate (object)
(10)              endloop
(11)        until not (member (first (new-objects), ELABORATION-RELATIONS)
(12)                    and embedding-possible (new-objects))
(13)        /* Phase 3 */
(14)        terminator-go
(15)        wait-for-last-sentence (activator, LDM)
(16)  until planning-terminated
(17)  processor-stop
```

Abbildung 5.11: Der Ablauf des Strukturaktivators

Funktion ist eine Datenstruktur, die den Zugriff auf die Realisierung dieses Konzeptes erlaubt (siehe Abschnitt 6.1). Relationen zwischen Konzepten werden mit

`(verbalize-ckb-link` *from-concept relation to-concept* `)`

angegeben. Neben der Realisierung einzelner Elemente der CKB benötigt der Strukturaktivator zusätzlich noch die Funktion

```
(terminator-go)
```

mit der POPEL-HOW mitgeteilt wird, daß aus der Sicht der Inhaltsfestlegung die Konzepte, die in einem Satz verbalisiert werden sollen, ausgewählt wurden und die Realisierung dieses Satzes beendet werden soll.

5.3.3 Der Ablauf des Strukturaktivators

Vom parallel laufenden Planer erhält der Strukturaktivator die primitiven Ziele in der Reihenfolge, in der sie geäußert werden sollen, sowie die RST-Relationen zwischen den primitiven Zielen. In der Reihenfolge, in der die Ziele ausgewählt wurden, werden sie auch aufbereitet und an POPEL-HOW weitergegeben.

Der Algorithmus zerfällt in drei Phasen (siehe Abbildung 5.11). In der ersten (Zeilen 3-6) werden die Konzepte und Relationen, die in einem primitiven Ziel stehen, analysiert und aufbereitet. Für jedes Konzept des Ziels wird eine Datenstruktur des Typs `ACTIVE-ITEM` erzeugt, in der neben dem Konzept selbst alle in dem Ziel zu diesem Konzept angegebenen Relationen sowie Informationen darüber gespeichert werden, ob das

```
#<ACTIVATOR 45260360> is an instance
                      of class #<Standard-Class ACTIVATOR 37314344>:
 The following slots have :INSTANCE allocation:
                          ...
 SAID-THINGS              NIL
 ACTIVATED-ITEM-HISTORY   ((INFORM MOTORRAD)
                           (RST-RELATION ELABORATION MOTORRAD)
                           (INFORM MANN)
                           (RST-RELATION ELABORATION MANN)
                           (INFORM DRIVE (AGENS-TO VEHICLE-TYPE-DRIVE
                                          SOURCE DESTINATION))
                           (RST-RELATION SEQUENCE (DRIVE BUY)))
 NEW-OBJECTS              NIL
 DSM-START-INDEX          0
 OBJECTS-TO-ACTIVATE      (#S(ACTIVE-ITEM NET-ID MOTORRAD OBJECT NIL
                                 ORDER NIL))
 ACTIVE-OBJECTS           (#S(ACTIVE-ITEM NET-ID MANN OBJECT
                                 #<POPEL-HOW::CKB-OBJECT 42237454>
                                 ORDER 2)
                           #S(ACTIVE-ITEM NET-ID DRIVE OBJECT
                                 #<POPEL-HOW::CKB-OBJECT 44575444>
                                 ORDER 1))
                          ...
```

Abbildung 5.12: Ein Ausschnitt aus der Datenstruktur des Strukturaktivators

Element oder eine seiner Relationen vom Benutzer erfragt werden sollen. Diese Daten werden in `OBJECTS-TO-ACTIVATE` gespeichert.

In der zweiten Phase (7-10) werden die in dieser Variablen gespeicherten Informationen mit den oben genannten Schnittstellenfunktionen an POPEL-HOW übergeben. Der Rückgabewert der Funktion und die Position dieses Elements in der gerade in Bearbeitung befindlichen Äußerung werden in die Datenstruktur geschrieben und diese selbst wird in die Liste der `ACTIVE-OBJECTS` aufgenommen (9). Sind keine aufbereiteten Elemente der CKB mehr vorhanden, wird mit den Heuristiken aus Abschnitt 5.3.1 entschieden, ob weitere primitiven Ziele zu der in Produktion befindlichen Äußerung hinzugefügt werden können. In diesem Fall wird wieder die erste Phase durchlaufen (11-12).

Soll die Äußerung abgeschlossen werden, wird in der dritten Phase (13-15) der Terminator für POPEL-HOW aufgerufen (14) und darauf gewartet, daß zu jedem aktivierten Element ein Eintrag in das Dialoggedächtnis erfolgt ist, d.h. daß die Verbalisierung erfolgreich war (15). Die Verarbeitung wird mit Phase 1 fortgesetzt, solange die Äußerung nicht fertiggeplant ist (16). Ist dies der Fall, wird vom Strukturaktivator das gesamte System angehalten (17).

In Abbildung 5.12 ist ein Auschnitt der Datenstrukturen des Strukturaktivators während der Verbalisierung des Eingabeziels

```
(SWMB (KNOW H (BUY DRIVE)))
```

dargestellt.[5] In der Variablen SAID-THINGS werden die geäußerten Sätze, nicht aber die inkrementell erzeugten Zwischenergebnisse gespeichert, und in ACTIVATED-ITEM-HISTORY die bereits bearbeiteten Ziele und RST-Relationen. NEW-OBJECTS ist die Schnittstelle zum Planer, in der dieser die neu festgelegten Relationen und Ziele schreibt und von der aus sie der Strukturaktivator weiterverarbeitet. DSM-START-INDEX enthält den Index des LDM am Beginn der Verbalisierung. Die zwei Variablen OBJECTS-TO-ACTIVATE und ACTIVE-OBJECTS wurden oben beschrieben. In diesem Fall wurden MANN und DRIVE bereits an POPEL-HOW übergeben, während MOTORRAD noch aktiviert werden muß.

5.4 Die Synchronisation von Planer und Strukturaktivator

Die Parallelität des Planers und des Strukturaktivators erfordert es, beide Prozesse zu synchronisieren. Die Planung des Inhalts darf nicht unabhängig von der Geschwindigkeit der Realisierung sein: durch die inkrementelle Realisierung verändern sich die Wissensquellen des Systems und damit die Grundlage, auf der die Planoperatoren und die Variablenbindungen ausgewählt werden. Der Planer muß deshalb warten, wenn die Realisierung der an den Strukturaktivator übergebenen Ziele nicht so schnell erfolgt, wie die Planung weiterer Ziele durchgeführt werden kann.

Die Synchronisationsbedingungen müssen es aber erlauben, daß die *strategische* Planung, d.h. diejenige, die die generellen Strukturen festlegt, durchgeführt werden kann, ohne daß die Realisierung eines gerade in Bearbeitung befindlichen Satzes dies behindert. Erst bei der *taktischen* Planung, d.h. dann wenn diese zu einem primitiven Ziel führt und von der aktuellen Äußerung beeinflußt werden kann, muß gewartet werden, wenn der vorher geplante Inhalt noch nicht realisiert ist. Werden die beiden Stufen der Kaskade so synchronisiert, kann das System bereits mit der Realisierung einer Äußerung beginnen, bevor festgelegt ist, wie deren gesamter Inhalt aussehen wird.

Die Synchronisation erfolgt zum einen über die Anzahl der Ziele, die vom Planer an den Strukturaktivator übergeben werden und zum anderen über die Strukturierung in Sätze. Wenn der Planer Ziele in die Variable NEW-OBJECTS schreibt, überprüft er, ob dort bereits ein Ziel steht. Wenn ja, wartet der Planungsprozeß solange, bis die Liste der Ziele abgearbeitet ist, und sich der Strukturaktivator in der Phase befindet, in der er die ausgewählten CKB-Strukturen an POPEL-HOW übergibt. Wurde im Strukturaktivator festgestellt, daß ein neu eintreffendes Ziel nicht mehr zu dem gerade verbalisierten Satz gehört, wird der Planer ebenfalls angehalten, bis der Satz in POPEL-HOW realisiert wurde.

Planer und Strukturaktivator müssen auch dann interagieren, wenn der Planer für ein Ziel keinen anwendbaren Operator findet und Planknoten neu bearbeitet werden müssen.

[5] Die Implementation ist objektorientiert durchgeführt. Die Objektklasse AKTIVATOR umfaßt die Daten und Funktionen des Strukturaktivators. Es existiert während des Ablauf von POPEL eine Instanz dieser Klasse.

Diese Neuplanung kann auch die bereits an den Strukturaktivator übergebenen primitiven Ziele betreffen und deren unter Umständen bereits in der Realisierung befindlichen Inhalte.

Sind die Ziele noch nicht an POPEL-HOW übergeben, können sie einfach aus dem Strukturaktivator gelöscht werden. Sind sie in der Realisierungskaskade, aber noch nicht geäußert, muß POPEL-HOW es ermöglichen, sie aus der Verarbeitung zu nehmen. In Abschnitt 6.1 wird gezeigt, daß die inkrementelle Verarbeitung Mechanismen bereitstellen muß, um mit Mehrdeutigkeiten umgehen zu können. Diese Mechanismen können auch für das Löschen von Objekten benutzt werden.

5.5 Die Bearbeitung von Anfragen

Das Pendant zum Strukturaktivator, der die zur Verbalisierung ausgewählten Teile der konzeptuellen Wissensquelle an POPEL-HOW übergibt, ist die Behandlung der Anfragen, die von POPEL-HOW an POPEL-WHAT gestellt werden. Der Funktionsumfang hängt von den Interaktionen ab, die in POPEL zwischen den beiden Hauptkomponenten erfolgen sollen. In Abschnitt 3.4.2.3 wurden einige Interaktionsmöglichkeiten vorgestellt, von denen drei implementiert wurden.

- Die Auswahl konzeptuellen Wissens: Bei der Realisierung des konzeptuellen Wissens mit POPEL-HOW kann es dort zu der Situation kommen, daß die Realisierung nicht fortgeführt werden kann, da zusätzliches Wissen benötigt wird. Dieses Wissen wird, in Form von Konzepten und Rollen der CKB, von POPEL-HOW an POPEL-WHAT übergeben (siehe Abschnitt 6.3). Der Entscheidungsprozeß, der bestimmt, ob angefragte Elemente aktiviert werden sollen, wurde in einer einfachen Implementation realisiert. Es werden nur Anfragen nach Konzepten und Rollen für Objekte der CKB A-Box behandelt, da angenommen wird, daß die Erklärungsplanung der T-Box alle notwendigen Teile selektiert. Diese führen zur Aktivierung der angeforderten Konzepte und Rollen, falls nicht für das angeforderte Konzept im LDM ein referentielles Objekt existiert, das den höchsten Fokuswert besitzt. Dann wird angenommen, daß das Konzept aus dem Kontext erschlossen werden kann. Diese Bedingung gilt nicht, wenn die Anforderung aus der Verarbeitung eines anderen Konzepts herrührt, das nur zusammen mit dem angeforderten weiterverarbeitet werden kann.

- Generierung von Referenzausdrücken: Anfragen, die im Zusammenhang mit der Generierung von Referenzausdrücken stehen, werden an die Komponente für deren Behandlung weitergeleitet (siehe Kapitel 7).

- Ellipsen: Kann aus den ausgewählten konzeptuellen Strukturen kein vollständiger Satz gebildet werden, werden bei der Generierung von Ellipsen Daten über den Kontext benötigt, aus denen die fehlende Information rekonstruiert werden kann. Da Ellipsen bei der inkrementellen Generierung nur dann als solche erkannt werden, wenn das Terminierungssignal vom Strukturaktivator an POPEL-HOW übermittelt wird, erfolgen die Anfragen, wenn sich POPEL-HOW in der Terminierungsphase befindet (siehe Abschnitt 6.4).

Kapitel 6

Die Verarbeitung in der Realisierungskaskade

Nach der *Auswahl* des Inhalts muß für diesen in der inhaltsrealisierenden Komponente von POPEL eine *satzsemantische* und *syntaktische Beschreibung* erzeugt werden, bevor dem Dialogpartner die korrespondierende Äußerung mitgeteilt werden kann. Die Anforderungen an POPEL-HOW, die die Architektur von POPEL stellt, wurden in den Abschnitten 3.4.2 und 3.4.3 dargelegt. Das vorangehende Kapitel legte Bedingungen fest, die sich aus dem Aufbau und Ablauf der inhaltsfestlegenden Komponente ergeben haben. Insbesondere sind dies, daß die inkrementelle Realisierung von nacheinander ausgewählten konzeptuellen Strukturen möglich sein muß, damit die Strukturierungskomponente die Satzstruktur festlegen kann. Zudem muß POPEL-HOW in der Lage sein, fehlende Informationen anzufordern, die aus der Sicht des realisierungsspezifischen Wissens notwendig sind.

Zunächst wird die operationale Grundlage des Modells vorgestellt. Die Realisierungskomponente besteht aus einem verteilten, parallelen System unabhängiger, kooperierender Prozesse, die Ausschnitte der Wissensquellen beschreiben und die Transformationen zwischen den Ebenen durchführen. Besondere Anforderungen entstehen dabei in der syntaktischen Verarbeitung, insbesondere in der Verteilung des syntaktischen Wissens. Schließlich wird dargestellt, wie Anforderungen erzeugt und bearbeitet werden, und wie die Terminierung der Verarbeitung in POPEL-HOW erfolgt.[1]

[1]Der erste Prototyp von POPEL-HOW ist in [Finkler, 1989, Neumann, 1989] beschrieben. Die hier vorgestellte Version unterscheidet sich vor allem durch eine stark erweiterte Funktionalität, u.a. den automatischen Regelaufbau zwischen der FSS– und der DBS-Ebene, die Struktur der Grammatik, die Generierung von Ellipsen und die Behandlung von Anfragen. Zudem ist die Generierung von Referenzausdrücken (vgl. Kapitel 7) völlig neu.

6.1 Das parallele Basismodell

6.1.1 Eine Übersicht über das Modell

Aus der Gesamtarchitektur des POPEL-Systems ergeben sich die folgenden Charakteristika im Aufbau der Realisierungskomponente (vgl. die Abschnitte 3.4.2.2 und 3.4.3):

- Das System besteht aus einer vierstufigen Kaskade, deren Eingabe aus Elementen der konzeptuellen Wissensquelle besteht.
- Diese CKB-Elemente werden inkrementell mit Hilfe von Regeln auf funktional-semantische Strukturen abgebildet, aus denen wiederum regelbasiert syntaktische Strukturen erzeugt werden.
- Die einzelnen Elemente werden weitgehend autonom auf die nächste Ebene abgebildet, damit eine inkrementelle Arbeitsweise gewährleistet ist und möglichst rasch Restriktionen, die sich aus den sprachlichen Wissensquellen ergeben, die Arbeit in POPEL-WHAT unterstützen können.
- Die Elemente einer Verarbeitungsebene bilden im Sinne der darunterliegenden Wissensquelle wohlgeformte Strukturen, müssen jedoch auch Verbindungen zu den Elementen haben, aus denen sie entstanden sind, damit der Datenfluß für eine Kommunikation zwischen den Ebenen möglich ist.

Grundlage für die kaskadierte Verarbeitung in POPEL-HOW bildet ein einheitliches Modell, das es erlaubt, die Verarbeitung innerhalb der einzelnen Repräsentationsebenen möglichst unabhängig voneinander durchzuführen. Für jedes Element, das autonom bearbeitet werden kann, wird innerhalb dieses *verteilten, parallelen Modells* ein eigener Prozeß (oder ein *Objekt*) erzeugt, der/das die Verarbeitung dieses Elements übernimmt.

Die Bezeichnung *Objekt* wurde in Anlehnung an die Terminologie der objektorientierten Programmierung gewählt und gibt bereits einen Hinweis auf die spätere Implementierung. Die Realisierungskaskade ist im Sinne des *object-oriented concurrent programming* aufgebaut [Yonezawa and Tokoro, 1987]. Bei dieser Methode werden objektorientierte Programmierung und Parallelverarbeitung kombiniert, indem Objekte in einer objektorientierten Programmiersprache als unabhängige Prozesse definiert werden.

Die Objekte einer Ebene kooperieren, wenn die Verarbeitung nicht autonom von den einzelnen Objekten durchgeführt werden kann, z.B., wenn die Transformationsregeln, die die Abbildung zwischen den Repräsentationsebenen steuern, Abhängigkeiten zwischen mehreren Elementen einer Wissensquelle definieren. Die Anzahl der Objekte in einer Ebene ist nicht begrenzt, da durch die inkrementelle Verarbeitung jederzeit weitere Eingaben in die Ebene erfolgen können.

Verbindungen zwischen den Objekten einer Ebene bestehen dann, wenn gemäß der zugrundeliegenden Wissensquelle, von der sie einen Ausschnitt repräsentieren, Relationen zwischen ihnen bestehen. Die Repräsentation dieser Relationen definiert im parallelen Modell die Verbindungen, über die die Objekte miteinander kommunizieren können. Zwischen Objekten benachbarter Ebenen werden Nachfolger– bzw. Vorgängerrelationen definiert,

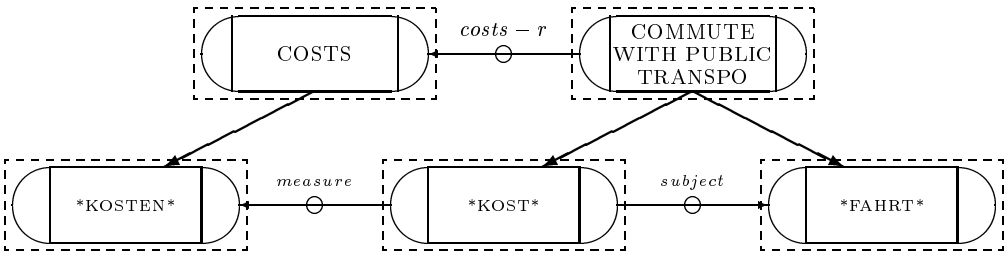


Abbildung 6.1: Die Segmentierung und Abbildung in den SB-ONE basierten Ebenen

über die bidirektional Daten ausgetauscht werden können, wenn die Objekte der einen Ebene aus einer Abbildung entstanden sind, die die Objekte der darüberliegenden Ebene durchgeführt haben. Die Kommunikation zwischen den Objekten kann sowohl *synchron* erfolgen, d.h. die Anfrage eines anderen Objekts wird sofort bearbeitet, wobei die Verarbeitung des befragten Objekts unterbrochen wird, als auch *asynchron*, d.h. die Anfrage wird in einem Puffer zwischengespeichert und zu einem späteren Zeitpunkt bearbeitet.

Das Modell definiert zunächst nur die grundlegende Vorgehensweise in der Kaskade aus verteilten Prozessen, die in allen Stufen einheitlich sein soll. Festzulegen bleibt, was die charakteristische Eingabe für die FSS–Ebene und die Ebenen der syntaktischen Verarbeitung ist, wie die spezielle Ausprägung der Verarbeitung für ein Objekt einer Ebene aussieht, wie die regelbasierte Abbildung durchgeführt wird, und wie die Probleme der Synchronisation und der Kommunikation innerhalb des verteilten Systems gelöst werden.

6.1.2 Die Verteilung der Wissensquellen

Die inkrementelle Verbalisierung in einem Kaskadenmodell setzt voraus, daß zu jeder Stufe die charakteristische Eingabe definiert werden muß, die innerhalb dieser Stufe eine Verarbeitung auslösen kann (vgl. Abschnitt 3.4.3.1). Diese minimalen, aktivitätsauslösenden Eingabestrukturen legen damit auch fest, welche Segmente der zugrundeliegenden Wissensquelle innerhalb der Stufe parallel verarbeitet werden können (vgl. Abschnitt 3.4.3.3). Den Segmenten können dann Objekte des Verarbeitungsmodells zugeordnet werden, die die Verarbeitung übernehmen. Bei der Segmentierung muß darauf geachtet werden, daß der Aufwand der Dekomposition und der Kommunikation zwischen den Prozessen nicht zu groß wird.

6.1.2.1 Die Verteilung in den SB-ONE basierten Ebenen

Die Eingabe in die erste Kaskadenstufe, die CKB-Ebene, liegt durch die Ausgabe der inhaltsfestlegenden Komponente bereits fest. Diese übergibt an POPEL-HOW Konzepte der T– und A-Box, sowie Relationen zwischen diesen Konzepten. Da die Wissensquelle der FSS ebenfalls mit SB-ONE formuliert wird, muß auch die charakteristische Eingabe in diese Ebene aus Elementen der Wissensrepräsentationssprache bestehen. Auch hier gilt, wie bei der Festlegung der charakteristischen Eingabe in die CKB-Ebene (vgl. Abschnitt 5.3.2), daß weder eine feinere noch eine gröbere Strukturierung möglich und sinnvoll ist.

In SB-ONE ist das *Konzept* das prinzipielle bedeutungstragende Element, dessen Bedeutung durch seine Relationen zu anderen Konzepten definiert wird. Wie später gezeigt wird (vgl. Abschnitt 6.1.5), gehen auch die Transformationsregeln zwischen den Ebenen von dem Konzept als zentralem Element der Abbildung aus, sowohl in der Ausgangsstruktur der Abbildung als auch in der Zielstruktur. Diese *konzeptzentrierte Sichtweise* legt es nahe, innerhalb der durch die zugrundeliegende SB-ONE Wissensquelle definierten Ebenen der CKB und FSS jedes Konzept, das verarbeitet werden soll, als Objekt aufzufassen. Dieses ist autonom und bildet sich parallel auf die nächste Verarbeitungsebene ab. Die Relationen zwischen den Konzepten werden auf die Kommunikationsverbindungen abgebildet, über die Nachrichten zwischen den Objekten ausgetauscht werden können.

Ein Beispiel für die Segmentierung der Strukturen der konzeptuellen und satzsemantischen Verarbeitungsebenen, die auf SB-ONE basieren, wird in Abbildung 6.1 gezeigt. Die Eingabe in POPEL-HOW kann mit dem Ziel

```
(SWMB (KNOW S (COSTS-R COSTS)))
```

vom Planer erzeugt werden. Den individualisierten Konzepten werden Objekte des Verarbeitungsmodells zugeordnet, die in der Abbildung mit der gestrichelten Umrahmung dargestellt sind. Die Pfeile zwischen den Ebenen deuten die Transformationen zwischen den beiden Repräsentationsebenen an.

6.1.2.2 Die Verteilung in der syntaktischen Ebene

Die Abbildung zwischen den SB-ONE basierten Wissensquellen ist eine homogene Abbildung, bei der in der Ausgangs– und Zielebene das Segmentierungsprinzip identisch ist. Anders ist dies beim heterogenen Übergang von der FSS– in die syntaktische Ebene. Die charakteristische Eingabe, die die Zielstruktur der Abbildung aus der FSS-Ebene ist, liegt noch nicht fest und hängt wesentlich von der dieser Ebene zugrundeliegenden syntaktischen Wissensquelle ab.

Damit eine bestimmte Grammatik in POPEL-HOW verwendet werden kann, muß sie die Bedingungen erfüllen, die Levelt für das menschliche Sprachproduktionssystem aufgestellt hat, nämlich sie muß lexikonzentriert sein und Fragmente in beliebiger Reihenfolge akzeptieren, um eine inkrementelle Verarbeitung zuzulassen (vgl. Abschnitt 2.6.2 und [Kempen, 1987a, Neumann and Finkler, 1990]). Besonders der Aufbau der syntaktischen Struktur von einem lexikalischen Element hin zu einer Phrase ist wichtig, da es in einem inkrementellen System nicht plausibel ist, die syntaktische Struktur von oben nach unten, ausgehend von einem initialen Wurzelknoten S aufzubauen.

In der FSS-Ebene beschreiben die Konzepte und die ihnen zugeordneten Verarbeitungsobjekte Wörter des semantischen Lexikons und deren Argumentrahmen. Annotiert an den Rollen sind die grammatischen Funktionen, die diesen Rollen entsprechen. Die Struktur der FSS-Objekte kann dann identisch auf die syntaktische Ebene abgebildet werden, wenn die Grammatik wortzentriert arbeitet, wie es z.B. die Dependenztheorie vorschlägt [Engel, 1988]. In diesem Fall besteht die charakteristische Eingabe in die syntaktische Verarbeitungsebene aus den Wörtern und den syntaktischen Beziehungen zwischen diesen Wörtern. Den Wörtern werden Objekte zugeordnet, die syntaktischen Beziehungen

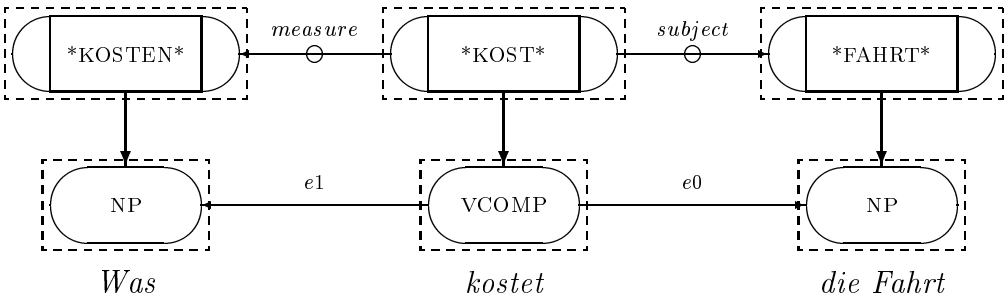


Abbildung 6.2: Die Segmentierung und Abbildung in der FSS– und der DBS-Ebene

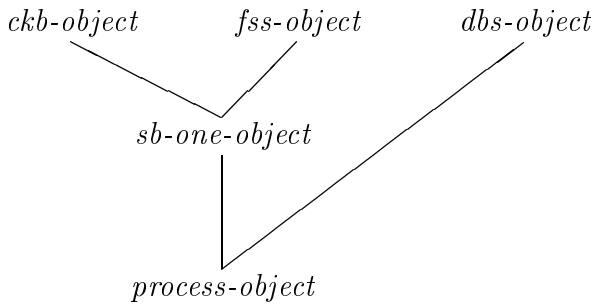


Abbildung 6.3: Die Hierarchie der Prozeßobjekte

dienen als Kommunikationsverbindungen. Die Aufgabe der Objekte besteht nicht mehr darin, eine Transformation in eine weitere Repräsentationsebene durchzuführen, sondern es muß eine Beschreibung erzeugt werden, die es erlaubt, eine syntaktisch korrekte Äußerung an den Dialogpartner auszugeben.

Abbildung 6.2 zeigt die Segmentierung auf der syntaktischen Ebene, wie sie mit der in POPEL-HOW verwendeten Grammatik möglich ist. Die Phrasenstruktur basiert auf den Relationen zwischen dem Kopfelement einer Phrase, in diesem Beispiel der Verbalphrase `VCOMP`, und dessen Dependenten, in diesem Fall den Nominalphrasen `NP`. Die syntaktischen Relationen lehnen sich an die Terminologie von Engel an, wobei in der traditionellen Grammatik `e0` dem Subjekt und `e1` dem Akkusativobjekt entspricht (siehe Abschnitt 6.2 und [Engel, 1982])

6.1.3 Die Definition der Prozeßobjekte

Die Prozeßobjekte sind die Träger der Verarbeitung in den Ebenen von POPEL-HOW, die alle prinzipiell gleich aufgebaut sind. Die Realisierungsgrundlage ist, wie oben gesagt, das objektorientierte Programmierparadigma, in dem es möglich ist, die gewünschte Funktionalität durch das "Mischen" verschiedener Objekte zu erhalten. Abildung 6.3 zeigt die Hierarchie der Objekte in POPEL-HOW: alle Prozeßobjekte erben `process-object`, das die Grundfunktionen und –variablen zur Verfügung stellt, die im parallelen Modell benötigt werden. Darauf aufgebaut sind die Objekte der Ebenen, die mit in SB-ONE definierten Wissensquellen arbeiten, sowie die Objekte der syntaktischen (DBS) Ebene.

Diese Objekte fügen dem Grundobjekt Daten hinzu, die ebenenspezifisch notwendig sind.

Am Beispiel des FSS-Objekts in Abbildung 6.4 sollen die wichtigsten Variablen eines Objekts genannt werden.[2] Die Verbindung mit der Vorgänger- und Nachfolgerebene wird mit den in den Variablen `FATHER-POINTER` und `SON-POINTER` enthaltenen Informationen hergestellt. `CONTEXT` und `FILLED-OBJECTS` enthalten die Informationen, mit welchen Objekte Nachrichten innerhalb einer Ebene ausgetauscht werden können. In `CONTEXT` stehen alle anderen Objekte, die Füller einer von diesem Objekt ausgehenden Verbindung sind, `FILLED-OBJECTS` gibt diejenigen an, deren Füller das Objekt selbst ist. Um feststellen zu können, ob sich im Kontext des Objekts etwas geändert hat, wird zudem in `OLD-CONTEXT` der Vorgängerzustand der Verbindungen gespeichert. Mit `ATTENDANCE-LIST` identifiziert das Objekt, in welcher Ebene es sich befindet. Die Behandlung von Anfragen läuft über die Variablen `OLD-REQUESTS` und `NEW-REQUESTS`. Daneben hat jedes Objekt noch einen Namen, eine Variable `ORDER`, die die Reihenfolge der Aktivierung angibt, und eine Variable `SPECIALS`, in der Informationen gespeichert werden können, die spezielle Verarbeitungskomponenten benötigen, z.B. die Generierung von Referenzausdrücken.

Die Variablen `SBONE-ID` und `GENERAL-CONCEPT-IN-NET` stammen aus der Definition von `SB-ONE-OBJECT`, auf das `FSS-OBJECT` aufbaut, und enthalten den Bezeichner des individuellen und generellen Konzepts der FSS für dieses Prozeßobjekt. Dabei wird eine Individualisierung in der FSS-A-Box erst dann durchgeführt, wenn das dem FSS-Objekt entsprechende syntaktische Objekt vollständig ist und das entsprechende Wort ausgegeben wurde.

6.1.4 Die Verarbeitung in einem Objekt

Die parallele Verarbeitung eines Objektes beruht darauf, daß ihm bei seiner Erzeugung ein sequentielles Programm zugeordnet wird, das parallel zu den Programmen der anderen Objekte ausgeführt wird. Auf einer Mehrprozessoranlage könnte man jedem Objekt einen eigenen Prozessor zuordnen, auf dem dieses Programm abläuft. In der Simulationsumgebung, in der POPEL entwickelt wurde, wird bei der Objektklasse `process-object` die Objektklasse `psim-process` hinzu-"gemischt", die diese Funktionalität auf einer Einprozessoranlage bereitstellt (vgl. Abschnitt 8.1).

Das Programm ist für alle Objekte identisch und definiert den grundlegenden Ablauf, der für die inkrementelle Verarbeitung benötigt wird. Durch die Formulierung der einzelnen Verarbeitungsschritte als generische Funktionen kann die Verarbeitung ebenenspezifisch erweitert werden. So wird z.B. in der DBS-Ebene beim Aufbau der Kommunikationsverbindung, einer Basisfunktion der Verarbeitung im parallelen Modell, zusätzlich die syntaktische Kompatibilität der zu verbindenden Objekte überprüft.

Das Programm (siehe Abbildung 6.5) besteht aus einer Initialisierungsphase, in der u.a. der Name und der Verweis auf das Vorgängerobjekt festgelegt wird (Zeile 1). Die zentrale Verarbeitung findet in einer Endlosschleife statt (2-10). Die Terminierung des Programms kann nur von außen erfolgen, da bedingt durch die inkrementelle Arbeitsweise jederzeit

[2] Das Beispiel stammt aus einer Implementierung der Objekte mit der PCL-Version des *Common Lisp Object Systems* [Keene, 1989]. Die Prozeßobjekte sind hierbei Instanzen einer Klasse, die den jeweiligen Objekttyp definiert.

```
#<FSS-OBJECT 42174614> is an instance of class
                      #<Standard-Class FSS-OBJECT 35434060>:
 The following slots have :INSTANCE allocation:
                         ...
 NAME                       "*PERSON*-<FSS-GC126-1261>"
 SPECIALS                   ((FSS-ORDER 2)(KONSENS +) (CARD INF))
 ORDER                      2
 FATHER-POINTER             (#<CKB-OBJECT 43122214>)
 SON-POINTER                (#S(SON-DESCRIPTION OWNER SELF SONS
                                  (#S(SON&RULE SON "NP-1264"
                                      RULE
                                      #S(RULE LHS NIL RHS
                                         #S(RHS CONCEPT NP
                                            PATH-MAPPINGS NIL)
                                         DISCRIMINATION-NET
                                         NIL)))))
 OLD-CONTEXT                NIL
 CONTEXT                    NIL
 FILLED-OBJECTS             (#S(ELEMENT-OF-CONTEXT
                                ARC GR581             ; agent-Rolle
                                FLAG MARKED
                                FILLER #<FSS-OBJECT 42174614>
                                OWNER #<FSS-OBJECT 34650204>))
 OLD-REQUESTS               NIL
 NEW-REQUESTS               NIL
 ATTENDANCE-LIST            (*FSS-ATTENDANCE-LIST*
                             . #<ATTENDANCE-LIST 34076040>)
 SBONE-ID                   IC10
 GENERAL-CONCEPT-IN-NET     GC126
                         ...
```

Abbildung 6.4: Ein FSS-Objekt

```
(1)   define-initial-state (object)
(2)   while TRUE
(3)         newer-object := get-newer-object (object)
(4)         if    newer-object
(5)         then
(6)               check-for-filler (object, newer-object)
(7)         endif
(8)         handle-changed-environment (object)
(9)         requests (object)
(10)  endwhile
```

Abbildung 6.5: Das Basisprogramm der Prozeßobjekte

neue Objekte in einer Verarbeitungsebene eintreffen können, die Verbindung mit dem Objekt aufnehmen wollen.

Der erste Schritt jedes Verarbeitungszyklus ist die Kontaktaufnahme mit Objekten, die neu in der Ebene eingetroffen sind, und die aufgrund der unterliegenden Wissensquelle direkt mit ihm kommunizieren können (3-6). In dieser Phase wird die Vernetzung der Objekte aufgebaut (6).

Der nächste Schritt untersucht, ob sich der Kontext des Objekts geändert hat und führt, wenn dies möglich ist, die Abbildung in die nächste Ebene durch (8). Dieser Schritt muß in jedem Fall durchgeführt werden, auch wenn sich der *direkte* Kontext eines Objekts nicht geändert hat. Ist keine Kommunikationsverbindung mit einem anderen Objekt vorhanden, muß untersucht werden, ob die Abbildung lokal möglich ist. Hat sich der Kontext nicht geändert, muß untersucht werden, ob Transformationsregeln, die mehrere Objekte als Vorbedingung haben, durch Änderungen in den über den bereits bestehenden Kontext erreichbaren Objekten anwendbar sind. Die wiederholte Abbildung kann zu mehreren Nachfolgerobjekten eines einzigen Objektes auf der nächsten Verarbeitungsebene führen, die dort konkurrieren.

Der letzte Schritt ist die Verarbeitung von Anfragen und teilt sich in die Weiterleitung von Anfragen auf, die von einem Nachfolgerobjekt aus der nächsten Ebene stammen und die weitergeleitet werden, und in das Erzeugen eigener Anfragen (9, siehe auch Abschnitt 6.3). Letzteres ist nur dann zulässig, wenn das Objekt sich nicht abbilden konnte, d.h. wenn es zur Verbalisierung noch keinen eigenen Beitrag geleistet hat. Da möglichst schnell eine Rückwirkung von der Inhaltsrealisierung auf die Inhaltsfestlegung erfolgen soll und ein Objekt nur eine lokale Sicht auf den Gesamtzustand des Systems hat, wird sofort versucht, eine Anfrage an die nächsthöhere Ebene zu stellen, wenn keine Abbildung möglich war. Wird eine Anfrage aus einer tieferen Ebene bearbeitet, wird immer überprüft, ob die angeforderten Informationen aufgrund der inkrementellen Eingabe schon in der Verarbeitungsebene vorhanden sind und die Anfrage daher ignoriert werden kann.

6.1.5 Die regelbasierte Transformation zwischen den Ebenen

Die primäre Aufgabe der Objekte in jeder Verarbeitungsebene ist es, sich in die nächsttiefere Ebene abzubilden. Durch die Anordnung der Ebenen von einer konzeptuellen über eine satzsemantische bis zu einer syntaktischen Beschreibung wird so eine immer sprachnähere Struktur erzeugt, und als Resultat kann eine Äußerung an den Dialogpartner ausgegeben werden. Der Übergang zwischen den Verarbeitungsebenen in POPEL wird mit Hilfe von *Abbildungsregeln* durchgeführt, die einen systematischen Zusammenhang zwischen den Strukturen zweier benachbarter Wissensquellen herstellen.

6.1.5.1 Der Aufbau der Regeln

In POPEL-HOW erfolgt die Abbildung einmal zwischen zwei SB-ONE basierten Ebenen und einmal zwischen einer SB-ONE basierten Ebene und einer in einem anderen Formalismus definierten syntaktischen Ebene. Dennoch können gleich strukturierte Abbildungsregeln für beide Abbildungen verwendet werden, wenn von der real vorliegenden

Wissensquelle abstrahiert wird. Betrachtet man deren Struktur als Graph oder Netz, das aus *Knoten* und *Kanten* besteht, kann man allgemeine Abbildungsvorschriften über diesen Netzen definieren. Im Falle der SB-ONE Wissensquellen werden die generellen oder individualisierten Konzepte und Rollen als Knoten bzw. Kanten interpretiert, im Falle der syntaktischen Wissensquelle die Konstituentenstruktur, wobei die Terminale und Nichtterminale der Grammatik die Knoten darstellen, die syntaktischen Relationen die Kanten.

Die Abbildungsregeln müssen eine verteilte Abbildung erlauben, d.h. ein größerer Ausschnitt einer Wissensquelle muß durch die Abbildung seiner Einzelteile in die nächste Ebene transformiert werden können. Außerdem müssen sie es erlauben, beliebige Netz zu Netz Transformationen formulieren zu können. Letzteres bedeutet z.B. wie in Abbildung 6.1, daß aus einem Knoten einer Wissensquelle größere Netzstrukturen in der anderen Wissensquelle entstehen.

Gemäß der objektorientierten Sicht in POPEL-HOW sind die Regeln für die Knoten einer Wissensquelle definiert, die im Verarbeitungsmodell einem Prozeßobjekt entsprechen. Sie bestehen aus einem *Vorbedingungsteil*, in dem die Struktur des Ausgangsnetzes spezifiziert wird, die als Objektstruktur in der entsprechenden Verarbeitungsebene vorhanden sein muß, damit die Regel angewendet werden kann. Dieser Teil kann auch leer sein. Im *Aktionsteil* werden diejenigen Teile der Zielwissensquelle angegeben, für die auf der nächsten Stufe der Verarbeitung neue Objekte mit ihrem Verbindungskontext erzeugt werden müssen. Außerdem enthält der Aktionsteil die Informationen, wie die Elemente der Ausgangs– und der Zielwissensquelle miteinander in Beziehung stehen. Bei der Definition des Verbindungskontexts wird dann in der Zielstruktur vermerkt, aus welchen Objekten der Ausgangsstruktur die Füller in der Zielstruktur entstehen müssen. Wenn die Objekte bei der Ausführung ihres Programmes versuchen, Kommunikationverbindungen aufzubauen, dient diese Information neben der Struktur der darunterliegenden Wissensquelle dazu, eine korrekte Objektstruktur aufzubauen (siehe [Finkler, 1989, p.60ff] für die Definition der Regelsyntax).

Ein Beispiel für den Aufbau der Regeln ist in Abbildung 6.6 dargestellt. Sie sind für individualisierte Konzepte des generellen Konzepts `COMMUTE-WITH-PUBLIC-TRANSPO` definiert (siehe auch Abbildung 6.1, Seite 99). Die erste Regel ist dann anwendbar, wenn eine Rolle `costs-r` am Ausgangsobjekt vorhanden ist. Mit der Regel wird ein Objekt für das Konzept `*kost*` des semantischen Lexikons erzeugt, mit zwei Kommunikationsverbindungen, die für die Rollen `measure` und `subject` stehen. An der anderen Seite der `measure`-Kommunikationsverbindung kann ein FSS-Objekt stehen, das die Abbildung des Füllers der `costs-r` Rolle in der CKB-Ebene ist. Das `FILLER-PATTERN` beschreibt hier den Pfad, der von dem Objekt `(SELF)` aus zum Füller `OBJECT` vorhanden sein muß. Der Füller der `subject`-Verbindung ist die Abbildung des Objekts selber. Dort kann z.B. eines der Objekte stehen, die mit einer der beiden anderen Regeln erzeugt werden, nämlich ein Objekt für das Prädikat `*fahr*` oder das nominale Konzept `*fahrt*`. Das bedeutet, daß bei einer Ausgangsstruktur, wie sie in Abbildung 6.1 dargestellt ist, aus dem Ausgangsobjekt insgesamt drei Objekte in der Zielebene entstehen. Die Abbildung zeigt die zwei, die miteinander in Verbindung treten können, da sie neben der Bedingung der Regel auch gemäß der unterliegenden FSS in der `subject` Relation zueinander stehen können. In

```
(#S(RULE LHS
         (#S(FATHER-ROLE SELF SELF ROLE costs-r))
         RHS
         #S(RHS CONCEPT *kost*
                PATH-MAPPINGS
                (#S(PATH-MAPPING
                    NEW-ROLE measure
                    FILLER-PATTERNS ((SELF costs-r OBJECT)))
                   #S(PATH-MAPPING
                      NEW-ROLE subject
                      FILLER-PATTERNS ((SELF))))))
   #S(RULE LHS NIL
           RHS #S(RHS CONCEPT *fahr* PATH-MAPPINGS NIL))
   #S(RULE LHS NIL
           RHS #S(RHS CONCEPT *fahrt* PATH-MAPPINGS NIL)))
```

Abbildung 6.6: Eine Abbildungsregel zwischen CKB und FSS (Ausschnitt)

der Abbildung nicht dargestellt ist das Objekt für das Prädikat, das mit keinem anderen Objekt in Verbindung treten kann.

Die Regeln werden bei dem zentralen Knoten in einem Regelpaket gespeichert, das sich aus Regeln dreier verschiedener Typen zusammensetzt. Der Bedingungsteil *einfacher Regeln* besteht nur aus direkt von dem Knoten wegführenden Kanten, während der von *kombinierten Regeln* aus größeren Teilnetzen bestehen kann, d.h. daß zur Überprüfung ihrer Anwendbarkeit mehrere Objekte kooperieren müssen. *Erweiterungsregeln* unterscheiden sich dadurch, daß mit ihnen eine bereits bestehende Zielstruktur erweitert werden kann.

6.1.5.2 Die Abbildung zwischen konzeptueller und satzsemantischer Ebene

Der Strukturaktivator übergibt an die inhaltsrealisierende Kaskade CKB-Konzepte und –Rollen. Die Schnittstelle besteht aus den in Abschnitt 5.3.2 genannten Funktionen `verbalize-ckb-object`, die ein Objekt auf der CKB-Ebene erzeugt, und `verbalize-ckb-link`, die eine Kommunikationsverbindung zwischen zwei Objekten aufbaut.

Parameter für `verbalize-ckb-object` ist ein generelles oder ein individualisiertes Konzept der CKB. Für ein generelles Konzept und für dessen Individualisierungen in der CKB existieren zwei verschiedene Regelpakete, die das Abbildungswissen enthalten.

In der generischen Funktion `handle-changed-environment`, in der die Auswahl der Regeln, die Überprüfung des Bedingungsteils und die Abbildung in die nächste Ebene durchgeführt wird, wird je nach Typ des Objekt auf das entsprechende Regelpaket zugegriffen. Dabei sind auch die Regeln, die für die A-Box Strukturen gelten, über den Elementen der T-Box definiert, da sie nicht nur für eine bestimmte Individualisierung definiert sind, sondern für alle Individualisierungen eines generellen Konzepts. Bei deren

Auswertung wird nicht auf die T-Box Elemente zugegriffen, mit denen sie definiert sind, sondern auf deren Individualisierungen in der A-Box. Die Beispielregeln der Abbildung 6.6 sind für die Individualisierungen des Konzepts `COMMUTE-WITH-PUBLIC-TRANSPO` definiert.

Die Konzepte im Aktionsteil der Regeln entsprechen Konzepten im semantischen Lexikon. Damit legen die Abbildungsregeln zwischen der CKB– und der FSS-Ebene auch die Auswahl der *Inhaltswörter* fest.

Wurde eine Regel ausgewählt, die noch nicht angewendet wurde, wird für das FSS-Konzept im Aktionsteil ein Objekt in der FSS-Ebene erzeugt. Es werden für alle Rollen, die in dem Aktionsteil angegeben sind, Kommunikationsverbindungen initialisiert, an deren anderem Ende sich die Nachfolgerobjekte der in den Pfadspezifikationen angegebenen CKB-Objekte einhängen können. Zudem werden die Verbindungen ziwschen den Objekten aufgebaut, über die die Anforderungen laufen können. Ist eine *Erweiterungsregel* anwendbar, so wird kein neues Objekt erzeugt, sondern lediglich eine neue mögliche Kommunikationsverbindung bei dem Zielobjekt hinzugefügt.

Die Verarbeitung in der FSS-Ebene erfolgt gemäß dem Basisprogramm der Prozeßobjekte. Die Objekte versuchen, miteinander Kommunikationsverbindungen herzustellen und sich in die syntaktische Ebene abzubilden. Die Objektstruktur in der FSS-Ebene repräsentiert mögliche individualisierte Strukturen in der A-Box der FSS. Eine Individualisierung für ein Objekt wird jedoch erst dann erzeugt, wenn es geäußert wurde, d.h. wenn es erfolgreich weiter abgebildet werden konnte. Die Regelanwendung erzeugt u.U. mehrere *konkurrierende* Objekte, die den gleichen Inhalt verschieden ausdrücken. Ein Beispiel dafür sind die FSS-Objekte für die Wörter `*fahrt*` und `*fahr*`, die mit den Regeln aus Abbildung 6.6 erzeugt werden. In dem Beispielskontext aus Abbildung 6.1 ist nur das Objekt `*fahrt*` erfolgreich, da es in die Struktur des Prädikats paßt.

Ebenso kann die Regelmenge für ein CKB-Objekt so aufgebaut sein, daß sie Regeln mit unterschiedlich starken Vorbedingungen enthält. Sind zu Beginn der Generierung noch wenig Kommunikationsverbindungen bei dem Objekt aufgebaut, werden zuerst Regeln mit wenigen Vorbedingungen angewendet, die Objekte für allgemeinere FSS-Konzepte erzeugen. Erzeugt die inhaltsfestlegende Komponente später zusätzliche Objekte im Kontext des CKB-Objekts, können u.U. Regeln mit restriktiveren Vorbedingungen feuern, die zu Spezialisierungen des ursprünglichen FSS-Objekts führen. Diese Spezialisierungen ersetzen dann das allgemeinere Objekt in den passenden Kommunikationsverbindungen der FSS-Ebene. Das ursprüngliche allgemeine Objekt kann also seine Verbalisierung nicht fortsetzen und keine Individualisierung in der FSS A-Box erzeugen.

6.1.5.3 Die Abbildung auf syntaktische Strukturen

Die Abbildung von der FSS-Ebene in die syntaktische Verarbeitungsebene gleicht derjenigen zwischen CKB– und FSS-Ebene. In Abschnitt 6.1.2.2 wurde die heterogene Abbildung zwischen der SB-ONE basierten FSS und der syntaktischen Ebene bereits angesprochen. Die Strukturierung des syntaktischen Wissens, die eine objektorientierte Sichtweise zuläßt, ermöglicht es auch, den Regelmechanismus der ersten Stufe der Abbildung zu übernehmen.

Im Gegensatz zur Aufteilung der Regeln in der CKB-Ebene der Verarbeitung existiert

```
#S(RULE-PACKET
   SIMPLE-RULES
   (#S(RULE LHS NIL
            RHS #S(RHS CONCEPT VCOMP PATH-MAPPINGS NIL)
            DISCRIMINATION-NET NIL))
   AUGMENT-RULES
   (#S(RULE LHS (measure)
            RHS #S(RHS CONCEPT VCOMP
                       PATH-MAPPINGS
                       (#S(PATH-MAPPING
                           NEW-ROLE E1
                           FILLER-PATTERNS
                           ((SELF measure OBJECT)))))
            DISCRIMINATION-NET NIL)
      #S(RULE LHS (subject)
              RHS #S(RHS CONCEPT VCOMP
                         PATH-MAPPINGS
                         (#S(PATH-MAPPING
                             NEW-ROLE E0
                             FILLER-PATTERNS
                             ((SELF subject OBJECT)))))
              DISCRIMINATION-NET NIL)
      #S(RULE LHS (location)
              RHS #S(RHS CONCEPT VCOMP
                         PATH-MAPPINGS
                         (#S(PATH-MAPPING
                             NEW-ROLE E6
                             FILLER-PATTERNS
                             ((SELF location OBJECT)))))
              DISCRIMINATION-NET
              #S(FSS-WORD-CHOICE-RESULT
                 SPECIALS NIL
                 MODIFIERS
                 (#S(ONE-WORD STEM "in" CAT PRAEPOSITION
                              SPECIALS ((DBS-ROLE E6)
                                        (SEM-ROLE location))))))))
```

Abbildung 6.7: Ein Ausschnitt des Regelpakets für das FSS-Konzept `*kost*`

für FSS-Konzepte nur ein Regelpaket. Der Anwendungsbereich ist weder die T– noch die A-Box der FSS, sondern die Struktur der aktiven Objekte und ihrer Kommunikationsverbindungen. Diese beschreiben, wie oben ausgeführt, mögliche Individualisierungen, die in die nächste Ebene abgebildet werden müssen. Im Bedingungsteil der Regeln stehen Konzepte und Rollen der FSS, in den Aktionsteilen syntaktische Kategorien und Relationen zwischen ihnen. In der syntaktischen Ebene werden Objekte für die Kategorien erzeugt, die Relationen dienen als Kommunikationsverbindungen. Die Verarbeitung dieser Objekte beschränkt sich aber nicht auf die Abbildung in eine weitere Ebene, sondern besteht im Aufbau syntaktischer Beschreibungen für die durch die Objekte repräsentierten Wörter und Kategorien.

Der Aufbau der Abbildungsregeln erfolgt automatisch aus den Daten des semantischen Lexikons. Jedes Wortkonzept des semantischen Lexikons steht in einer Subsumptionsbeziehung zu einem Konzept der FSS, aus dem die Wortklasse abgeleitet werden kann, z.B. Verben zum Konzept `PREDICATE` (vgl. Abschnitt 4.2). Damit kann der Typ der Phrase bestimmt werden, dessen Kopfelement das Wort bildet. Die Abbildung einer Rolle ergibt sich aus der Annotation, die an der Rolle die möglichen Relationen angibt, die dieser in der syntaktischen Ebene entsprechen.

Das Regelpaket für ein Konzept des semantischen Lexikons wird erzeugt, indem eine einfache Regel definiert wird, die das Konzept ohne Vorbedingung auf die korrespondierende syntaktische Phrase abildet. Zusätzlich wird für jede Rolle des Konzepts, die eine Annotation besitzt, eine Erweiterungsregel hinzugefügt, die bei dem Zielobjekt eine Kommunikationsverbindung für die der Rolle entsprechende Relation aufbaut.

In der Annotation einer Rolle, die einer Präpositionalphrase entspricht, befindet sich auch die Information über die Präposition, mit der diese Rolle an der Oberfläche realisiert werden kann. Der Erweiterungsregel wird diese zusätzliche Wortwahlinformation hinzugefügt. Bei der Abbildung zwischen der satzsemantischen und der syntaktischen Ebene wird auf diese Weise ein Teil der Funktionswörter bestimmt.

Abbildung 6.7 zeigt einen Ausschnitt für das Regelpaket des Konzeptes `*kost*`. Das Konzept wird auf das Kopfelement einer Verbalphrase (`VCOMP`) abgebildet, die Rolle `measure` auf die syntaktische Relation `e1` (Akkusativobjekt), die Rolle `subject` auf `e0` (Subjekt) und die Rolle `location` auf `e6` (Präpositionalobjekt). Die Erweiterungsregel für die Rolle `location` enthält die Präposition `in`, mit der die Relation verbalisiert wird.

6.2 Die syntaktische Verarbeitung

6.2.1 Anforderungen an die Syntax

Die syntaktische Verarbeitung unterscheidet sich von derjenigen in den SB-ONE basierten Ebenen. Bestand die Aufgabe der Objekte dort in der Abbildung in die nächste Ebene, muß während der syntaktischen Verarbeitung über der Objektstruktur eine syntaktische Beschreibung erzeugt werden, die es erlaubt, die Wörter zu flektieren und zu linearisieren, so daß dem Dialogpartner grammatisch korrekte Äußerungen ausgegeben werden können.

Ein inkrementelles und paralleles Verarbeitungsmodell stellt besondere Anforderungen an das syntaktische Wissen und die Verarbeitungsmethodik, die im System verwendet

werden können [Kempen, 1987a, Neumann and Finkler, 1990]:

- Die Grammatik muß lexikonzentriert sein (siehe auch [Levelt, 1989]).
- Die bevorzugte Expansionsrichtung der syntaktischen Beschreibung sollte in die Tiefe statt in die Breite gehen, und somit das Wachstum um einzelne Blätter unterstützen.
- Die Beschreibung sollte nach oben (ein neues Segment dominiert eine bereits bestehende Struktur) und unten (ein neues Segment erweitert die Struktur nach unten) erweiterbar sein.
- Die Reihenfolge der inkrementellen Eingabe in die syntaktische Verarbeitung muß nicht mit der syntaktisch korrekten Stellung übereinstimmen. Deswegen sollte die Bearbeitung der Dominanz von der Bearbeitung der Präzedenz getrennt werden.
- Damit so bald wie möglich eine inkrementelle Ausgabe erfolgen kann, ist ein lokales Vollständigkeitskriterium bei jeder Teilstruktur notwendig.
- Konkurrierende Verbalisierungsmöglichkeiten müssen Modifikationen und Korrekturen in der syntaktischen Beschreibung erlauben.

In POPEL-HOW wurden diese Anforderungen erfüllt, indem die syntaktische Verarbeitung in zwei Ebenen aufgeteilt wurde. In der Ebene der *dependenzbasierten Struktur* (DBS-Ebene) werden zu dem bei der Abbildung aus der FSS-Ebene erzeugten Strukturgerüst inkrementell die syntaktischen Beschreibungen aufgebaut, die dann in der ILS-Ebene *linearisiert* und *flektiert* werden. Das syntaktische Wissen ist in der unifikationsbasierten Grammatik POPELGRAM enthalten, in der die Beschreibung der Dominanzen von der der linearen Abfolge getrennt ist, und deren deklarative Formulierung eine Erweiterbarkeit sicherstellt.

6.2.2 Die Repräsentation syntaktischen Wissens

6.2.2.1 Die linguistischen Grundlagen

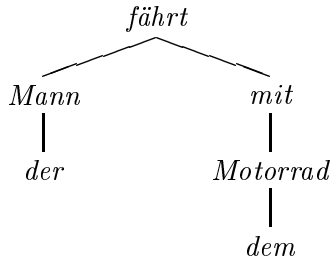
Die Beschreibung des strukturellen Wissens in den syntaktischen Ebenen von POPEL basiert auf der *dependentiellen Beschreibung* der Struktur von Äußerungen. Für diese Entscheidung waren zwei Tatsachen ausschlaggebend:

- die Eigenschaften einer dependenzbasierten Grammatik, vor allem die Wortzentrierung und die Betonung der strukturellen Beziehungen zwischen Wörtern, und die Trennung der Strukturbeschreibung von der linearen Abfolge;
- die Existenz fundierter Beschreibungen des Deutschen auf der Grundlage dieser Theorie (z.B. [Engel, 1988]).

Die fundamentale Relation in der Dependenztheorie ist die Beziehung zwischen einem regierenden Wort, dem *Regens*, und den in einer bestimmten Weise unmittelbar von ihm abhängenden Wörtern, den *Dependenten*. Für den Satz

Der Mann fährt mit dem Motorrad.

sieht die dependentielle Strukturbeschreibung folgendermaßen aus:



Das oberste Element einer Gruppe von Wörtern hat eine besondere Stellung, da es die gesamte Gruppe oder Phrase charakterisiert. In dem Beispiel sind dies das Verb, das die Struktur der Verbalphrase bestimmt, die Nomen für die Nominalphrasen oder die Präposition für die Präpositionalphrase. Jedes Wort, das Dependenten regieren kann, ist demzufolge Kopfelement einer Phrase [Engel, 1988, p.22].[3]

Dabei bestimmt das regierende Element, welche Dependenten bei ihm auftreten dürfen und welche nicht. So kann z.B. ein Artikel nicht Dependens eines Verbs sein. Die Elemente, die bei einem Wort oder bei einer Subklasse von Wörtern auftreten können und bei diesem eine syntaktische Funktion übernehmen, werden unterschieden in *Ergänzungen*, die notwendig (explizit oder implizit) vorhanden sein müssen, und *freie Angaben*, die bei allen Wörtern einer Wortklasse auftreten können. Die Ergänzungen werden zusätzlich unterteilt in die notwendigen, die für die grammatische Wohlgeformtheit benötigt werden, sowie die fakultativen, die weggelassen werden können, jedoch aus dem Kontext erschließbar sein müssen.

Die Objektstruktur, die mit dem verteilten Abbildungsprozeß erzeugt wird, bildet das Gerüst, über dem die Dependenzstruktur aufgebaut werden kann. Die Objekte der DBS-Ebene tragen die Inhaltswörter und fungieren damit als Kopfelemente einer Phrase. Funktionswörter wie Artikel werden zusammen mit den jeweiligen Inhaltswörtern in deren Objekten repräsentiert und bearbeitet. Die Kommunikationsverbindung zwischen zwei Objekten repräsentiert die syntaktische Funktion, die das Füllobjekt bei dem Ausgangsobjekt der Verbindung einnimmt. Über diesem Gerüst muß dann die explizite syntaktische Struktur aufgebaut werden, bei deren Konstruktion der Austausch der morphosyntaktischen Merkmale zwischen den Objekten erfolgt.

Die Dependenzstruktur spiegelt lediglich die gegenseitigen Abhängigkeiten der einzelnen Wörter wider, nicht jedoch die lineare Abfolge. Die Reihenfolge eines Regenten und seiner Dependenten hängt wiederum von der Wortklasse des Regenten ab. Das Verb im Deutschen bildet z.B. mit seinen finiten und infiniten Teilen die *Satzklammer*, und trennt

[3] In neueren anglo-amerikanischen Syntaxtheorien, z.B. HPSG [Pollard and Sag, 1987], wird die Idee, daß Kopfelemente die abhängigen Glieder eines Satz regieren, ebenfalls aufgenommen.

das Vor–, Mittel–, und Nachfeld des Satzes ab [Engel, 1988]:

$$\overbrace{Steuerhandlungen}^{Vorfeld}\underbrace{werden\overbrace{vonPersonen}^{Mittelfeld}ausgeführt}_{Satzklammer}.$$

In diesem Beispiel ist das Nachfeld unbesetzt. Das Vorfeld in Hauptsätzen kann bei Aussagesätzen von genau einem Dependenten des Verbs und seinen Dependenten besetzt sein, wobei es im Deutschen wenige Beschränkungen gibt, welche Dependenten nicht im Vorfeld stehen können. Im Mittelfeld stehen alle diejenigen Elemente, die nicht im Vor– oder Nachfeld stehen. Ihre Reihenfolge ist nicht so sehr von syntaktischen sondern vor allem von pragmatischen Kriterien abhängig. Nur pronominalisierte Elemente müssen direkt hinter dem finiten Verb stehen.

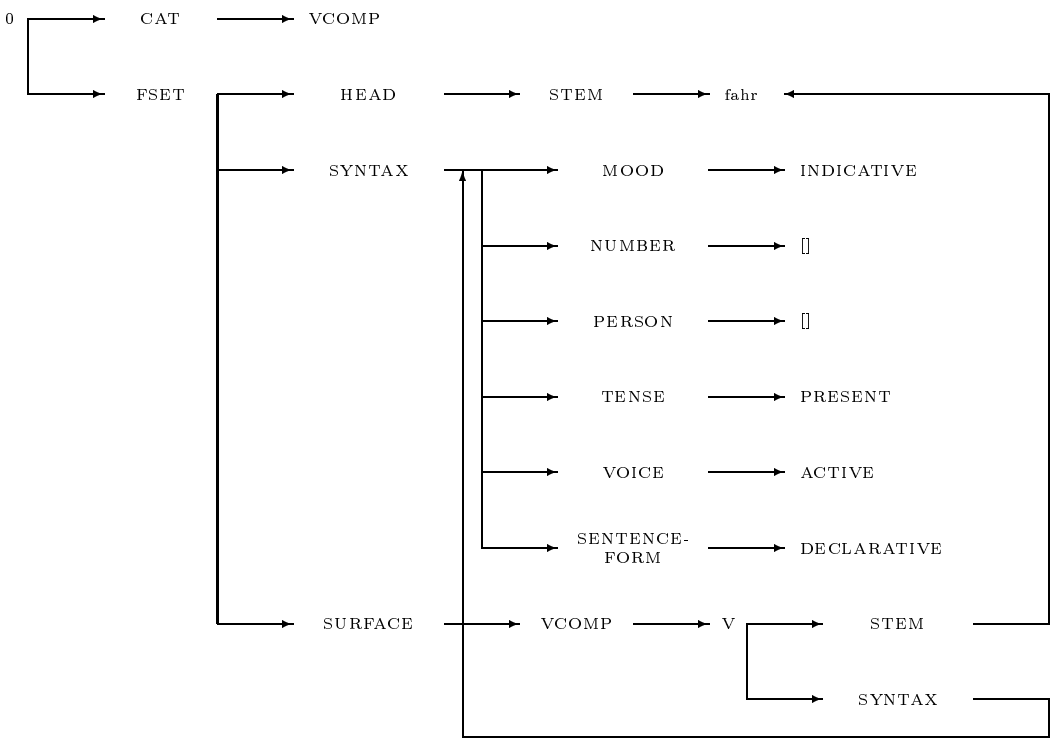
Die wichtigsten Kriterien für die Reihenfolge sind neben den syntaktischen Restriktionen die Absichten des Sprechers und die Dialogstruktur. Im Vorfeld stehen z.B. Dependenten des Verbs, die den Anschluß an den vorhergehenden Satz konstituieren oder besonders herausgehoben werden sollen. Die Reihenfolge im Mittelfeld hängt von der dynamischen Struktur der Elemente ab. Bekannte Elemente stehen i.a. nahe am finiten Verb, wobei sich diese Stellung bei pronominalisierten Elementen, die bereits durch ihre Wortform die Bekanntheit anzeigen, als feste Restriktion in der Grammatik niedergeschlagen hat. Neue oder hervorgehobene Elemente tendieren an das rechte Ende des Mittelfelds.

Die Auswahl der Reihenfolge des konzeptuellen Inhalts durch POPEL-WHAT erfolgt auf der Basis der Wichtigkeit, die den einzelnen Konzepten der CKB zugeordnet wird. Durch die Reihenfolge der Aktivierung und Verarbeitung in POPEL-HOW soll diese Auswahl bis auf die Ebene der Linearisierung erhalten bleiben, so daß im Idealfall das zuerst ausgewählte CKB-Element zuerst verbalisiert wird. Die Linearisierungskomponente dient dann hauptsächlich dazu, *restriktiv* dafür zu sorgen, daß keine ungrammatische Reihenfolge erzeugt wird, nicht aber dazu, die Reihenfolge der Elemente neu zu bestimmen.

6.2.2.2 Die Repräsentation des grammatischen Wissens mit PATR

Für die Repräsentation grammatischen Wissens gibt es ebenso wie für die des konzeptuellen Wissens Formalismen und Verarbeitungsmethoden, die es erlauben, Grammatiken deklarativ zu formulieren, und die hinreichend ausdrucksstark und formal fundiert sind. *Unifikationsbasierte Grammatiken*, die in den letzten Jahren entwickelt wurden, bieten aus theoretischer und praktischer Sicht eine gute Grundlage. Im letzten Jahrzehnt wurden eine Reihe solcher Grammatikformalismen vorgeschlagen, z.B. die *categorial unification grammar* [Uszkoreit, 1986]. Auch in Generierungssystemen wurden unifikationsbasierte Grammatiken verwendet, z.B. in den Systemen KAMP [Appelt, 1985] und EPICURE [Dale, 1988].

Ein Repräsentationssystem für unifikationsbasierte Grammatiken ist PATR-II [Shieber *et al.*, 198
Für PATR-II existieren eine Reihe von Entwicklungsumgebungen, die dem Entwickler auch einen Parser zur Verfügung stellen, anhand dessen Grammatiken durch die Analyse von Sätzen verifiziert werden können, z.B. Z-PATR [Shieber *et al.*, 1983] und D-PATR [Karttunen, 1986], aus dem das in XTRA verwendete System SB-PATR entwickelt wurde.




```
[0: [CAT: VCOMP
     FSET: [HEAD: [STEM: fahr]
            SYNTAX: [MOOD: INDICATIVE
                     NUMBER: []
                     PERSON: []
                     SENTENCE-FORM: DECLARATIVE
                     TENSE: PRESENT
                     VOICE: ACTIVE]
            SURFACE: [VCOMP: [V: [STEM: <0 FSET HEAD STEM>
                                  SYNTAX: <0 FSET SYNTAX>]]]]]]
```

Abbildung 6.8: Die graphische und die Listendarstellung eines DAGs

Die Repräsentation des syntaktischen Wissens erfolgt in PATR-II mit *gerichteten azyklischen Graphen* (directed acyclic graphs, DAG) , die Basisoperation ist die *Unifikation* zweier DAGs.

Mit DAGs lassen sich Merkmale, wie z.B. `NUMERUS`, und deren Werte, wie z.B. `3`, darstellen, indem eine gerichtete Kante zwischen dem Merkmal und dem Wert gezogen wird. Ein Wert kann leer sein oder selbst wiederum ein DAG. Abbildung 6.8 zeigt einen DAG, oben als Graph, unten in der Listennotation. Die Listennotation ist folgendermaßen zu lesen: jeder DAG ist mit eckigen Klammern geklammert, wobei `[]` den leeren Dag bezeichnet. Nach jedem Merkmal steht ein Doppelpunkt. In spitzen Klammern sind Pfade geschrieben. `STEM: <0 FSET HEAD STEM>` bedeutet z.B., daß der Wert des Merkmals `STEM` gleich dem Wert ist, den man erreicht, wenn man ausgehend vom Merkmal `0` die Merkmale `FSET`, `HEAD` und `STEM` durchläuft.

DAGs dienen in PATR-II zur Darstellung der morphosyntaktischen Strukturen, und zur Formulierung der Regeln, die die Bedingungen angeben, wie diese kombiniert werden können und welche Merkmale zwischen den Gliedern eines Satzes gleich sein müssen. Die Regeln bestehen aus einem Teil, der die Konstituentenstruktur der Phrase festlegt und aus einer Menge von Spezifikationen, die die Merkmale einzelner Konstituenten festlegen, oder die angeben, welche Merkmale unterschiedlicher Konstituenten gleich sein müssen. Die Regel für einen einfachen Satz mit Subjekt und Objekt

$$s \longrightarrow subj \;\; vp \;\; obj$$

$$\begin{aligned} < subj \;\; person > &= < vp \;\; person > \\ < subj \;\; numerus > &= < vp \;\; numerus > \\ < subj \;\; kasus > &= nom \\ < obj \;\; kasus > &= akk \end{aligned}$$

legt die Kasus der DAGs von Subjekt und Objekt fest. Bei der Expansion der Regel kann damit überprüft werden, ob eine Konstituente an die Stelle von Subjekt oder Objekt treten kann. Die ersten zwei Gleichungen sorgen dafür, daß die Merkmale *person* und *numerus* bei Verb und Subjekt gleich sind und in den DAG der Verbalphrase übernommen werden. Dabei werden die Merkmale nicht kopiert, sondern die Pfade resultieren im identischen Merkmal. Die Gleichungen sorgen dafür, daß die Merkmale durch die syntaktische Struktur "fließen".

In der Entwicklungsumgebung von SB-PATR erfolgt die Notation der Regeln in einer Listenstruktur. In ihr erhalten die Konstituenten des kontextfreien Teils Indizes von Null aufwärts, über die im Spezifikationsteil auf die DAG-Struktur der einzelnen Konstituenten zugegriffen werden kann. Der Spezifikationsteil selbst besteht aus zweielementigen Listen, wobei ein Listenelement ein Pfad durch einen DAG oder ein Merkmal ist. Sind die zwei Elemente der Spezifikationsliste Pfade, so wird damit ausgedrückt, daß diese gleich sein müssen, damit die Regel angewandt werden kann. Ist das zweite Element ein Merkmal, wird damit dem Pfad des ersten Elements ein Wert zugewiesen. Die obige Regel kann in SB-PATR folgendermaßen notiert werden:

```
(s subj vp obj
   ((1 person) (2 person))
   ((1 numerus) (2 numerus))
   ((1 kasus) nom)
   ((3 kasus) akk))
```

Die Überprüfung der Korrektheit und die Erweiterung eines DAGs mit den Informationen, die in einem anderen stehen, leistet die Unifikation. Unifiziert man den DAG aus dem Beispiel der Abbildung 6.8 z.B. mit dem DAG

```
[O: [FSET: [SYNTAX: [VOICE: PASSIVE]]]]
```

so schlägt dies fehl, da der Wert des Merkmals `VOICE` beider DAGs nicht übereinstimmt. Unifiziert man mit

```
[O: [FSET: [SYNTAX: [NUMBER: SG]]]]
```

so wird der leere DAG durch den Wert `SG` ersetzt.

6.2.2.3 Die syntaktische Wissensquelle POPELGRAM

Das grammatische Wissen in POPEL-HOW ist im PATR-II Formalismus repräsentiert. Der ursprüngliche Formalismus erlaubt es jedoch nicht, die Dominanzrelation von der linearen Abfolge zu trennen. Deswegen besteht die Grammatik aus zwei Regelmengen, von denen eine die Dominanzen zwischen den Phrasen darstellt (ID-Regeln), und die andere die Restriktionen für die lineare Abfolge enthält (LP-Regeln[4]).

Die ID-Regeln

Der ursprüngliche Aufbau der Regeln in PATR-II wird für die Notation der ID-Regeln übernommen, jedoch wird der kontextfreie Teil der Regeln neu interpretiert. Zudem müssen im Spezifikationsteil einige Merkmale vorhanden sein, die der syntaktische Generierungsalgorithmus benötigt.

Anstelle der Konstituentenstruktur enthält der kontextfreie Teil die *Dependenzstruktur* der Phrase. Das erste Element der Liste beschreibt die Phrase, die die Projektion des Kopfelements ist. Das Kopfelement selbst und die unmittelbaren Dependenten sind dann hinter der Phrasenklasse notiert, wobei deren Reihenfolge ohne Bedeutung ist. Über das spezielle Merkmal `head` wird im Spezifikationsteil das Kopfelement markiert, das die anderen Geschwister in der Phrase regiert. Ein Beispiel ist die Regel für die Projektion des Verbs, die Verbalphrase:

```
(vcomp v e0 e1
       ((0 head) (1 head)))
```

[4]LP steht für *linear precedence*

Die Bezeichner `e0` und `e1` wurden aus [Engel, 1982] übernommen und stehen für die Dependenten des Verbs, die als Ergänzung im Nominativ (`e0`) und im Akkusativ (`e1`) stehen.

Durch das `head`-Merkmal ist auch dasjenige Element der Regel ausgewiesen, das *lokal vollständig* sein muß, um sich in der verteilten Verbalisierung unabhängig von den Dependenten abbilden zu können. Wenn in der DAG-Struktur, die der Wert des Merkmals ist, keine leeren Teil-DAGs mehr enthalten sind, sind zu dem Wort, dessen Projektion die Phrase ist, alle relevanten Informationen vorhanden, die für eine Abbildung in die ILS-Ebene notwendig sind. Unter dem Merkmal `surface` werden dabei alle Lexeme gesammelt, die nicht mit Prozessen unabhängiger Dependenten verbalisiert werden, sondern zusammen mit dem Kopfelement. In POPEL sind dies größtenteils Funktionswörter, z.B. Artikel bei Nominalphrasen. Die morphosyntaktischen Merkmale der Phrasen wie Numerus, Person und Genus sind unter dem Merkmal `syntax` zusammengefaßt. Mit dieser Bündelung wird die Unifikation zweier Phrasen vereinfacht.

Der Merkmalstransport, der bei der Unifikation zweier DAGs erfolgt, ist *richtungsinvariant*. Nach der Unifikation ist aus einem DAG nicht mehr erschließbar, woher eines seiner Merkmale stammt. Die Objektstrukturen in den Verarbeitungsebenen von POPEL-HOW sind jedoch nicht fix. Mit dem Auftreten konkurrierender Objekte können Verbindungen gelöst werden. In den SB-ONE basierten Verarbeitungsebenen ist dies einfach möglich, da die Objektstrukur mit der Datenstruktur übereinstimmt. In der DBS-Ebene ist dies nicht mehr der Fall, da hier die Objektstrukur nur einen Teilbereich der syntaktischen Struktur umfaßt. In den DAGs der Objekte ist zusätzliche Information enthalten, die beim Lösen von Verbindungen zurückgesetzt werden muß. Für die DAGs bedeutet dies, daß Teile gelöscht und durch Strukturen des konkurrierenden Objekts ersetzt werden müssen. Damit festgestellt werden kann, in welche Richtung Merkmale fließen, wird unter dem Merkmal `synchronize` festgehalten, in welcher Richtung der Transport der Merkmale erfolgt.

Abbildung 6.9 zeigt eine Regel für Verbalphrasen in ihrer externen Notation und den daraus erzeugten DAG. Bei der Übersetzung werden für die Indizes der Elemente Merkmale erzeugt. Zusätzlich werden die Gleichungen des Spezifikationsteils unter dem Merkmal `fset` zusammengefaßt. Kongruenzbeziehungen bestehen in dieser Regel für den Aktivsatz zwischen dem Verb und der Nominativergänzung `e0`, die für den Merkmalstransport von Numerus und Genus sorgen, sowie zusätzlich mit der Akkusativergänzung `e1` die Bedingung, daß der Satz im Aktiv steht. Diese Merkmale können geändert werden und stehen deshalb unter dem `synchronize`-Merkmal.

Im Deutschen existieren ein– bis vierwertige Verben (ohne freie Angaben), für die sich, abhängig von den syntaktischen Kasus, Satzmuster definieren lassen ([Engel, 1988, p.200ff] definiert ca. 50). Diese könnten in die Grammatik aufgenommen werden, würden aber zum Aufbau einer Vielzahl redundanter syntaktischer Beschreibungen führen. Da bei der inkrementellen Generierung in einem dependentiellen Ansatz immer nur auf *lokale Vollständigkeit* untersucht wird, wurden die ID-Regeln so aufgebaut, daß die allgemeinsten möglichen Restriktionen formuliert werden, denen die Objekte genügen müssen.

Die Regel der Abbildung 6.9 zeigt einen allgemeinen Verbrahmen, mit den Ergänzungen, die bei einem Satz mit Vollverb auftreten können und die derzeit in POPELGRAM

a) Die externe Notation

```
(VCOMP V E0 E1 E3 E4 E6 E6-SOURCE E6-DEST E7
       ((0 SYNTAX VOICE) ACTIVE)
       ((0 SYNTAX) (1 SYNTAX))
       ((1 SYNTAX VOICE) (2 SYNTAX VOICE))
       ((1 SYNTAX VOICE) (3 SYNTAX VOICE))
       ((1 SYNTAX NUMBER) (2 SYNTAX NUMBER))
       ((1 SYNTAX PERSON) (2 SYNTAX PERSON))
       ((0 SYNCHRONIZE DOWN VOICE) (0 SYNTAX VOICE))
       ((0 SYNCHRONIZE DOWN NUMBER) (0 SYNTAX NUMBER))
       ((0 SYNCHRONIZE DOWN PERSON) (0 SYNTAX PERSON))
       ((0 HEAD) (1 HEAD))
       ((0 SURFACE VCOMP V) (1 SURFACE)))
```

b) Der DAG in der internen Darstellung

```
[0: [CAT: VCOMP
     FSET: [HEAD: []
            SYNTAX: [NUMBER: []
                     PERSON: []
                     VOICE: ACTIVE]
            SYNCHRONIZE: [DOWN: [VOICE: <0 FSET SYNTAX VOICE>
                                 NUMBER <0 FSET SYNTAX NUMBER>
                                 PERSON <0 FSET SYNTAX PERSON>]]
            SURFACE: [VCOMP: [V: []]]]]
 1: [CAT: V
     FSET: [HEAD: <0 FSET HEAD>
            SYNTAX: <0 FSET SYNTAX>
            SURFACE: <0 FSET SURFACE VCOMP V>]]
 2: [CAT: E0
     FSET: [SYNTAX: [NUMBER: <0 FSET SYNTAX NUMBER>
                     PERSON: <0 FSET SYNTAX PERSON>
                     VOICE: <0 FSET SYNTAX VOICE>]]]
 3: [CAT: E1
     FSET: [SYNTAX: [VOICE: <0 FSET SYNTAX VOICE>]]]
 4: [CAT: E3]
 5: [CAT: E4]
 6: [CAT: E6]
 7: [CAT: E6-SOURCE]
 8: [CAT: E6-DEST]
 9: [CAT: E7]
 ARITY: 9]
```

Abbildung 6.9: Eine Regel für Verbalphrasen

definiert sind. Es ist also der *allgemeinste* Argumentrahmen, der bearbeitet werden kann. Welche Dependenten gefüllt werden, hängt von dem jeweiligen Satz und dessen Wörtern ab. Das Satzmuster für ein spezielles Verb entspricht den Erweiterungsregeln, die für ein Wort des semantischen Lexikons bei der Abbildung zwischen FSS– und DBS-Ebene berechnet werden. Die Transformationsregel zwischen FSS– und DBS-Ebene für `*kost*` (siehe Abbildung 6.7) definiert z.B. einen dreiwertigen syntaktischen Argumentrahmen. In der obigen ID-Regel werden mit dieser Regel die Leerstellen `e0`, `e1` und `e6` gefüllt, während die übrigen offen bleiben. Da immer nur die lokale Vollständigkeit des Kopfelements überprüft wird, kann die Abbildung dieses Elements erfolgen, auch wenn global gesehen kein DAG ohne Leerstellen erzeugt wird.

Die LP-Regeln

Aufgabe der *Linearisierungsregeln* ist es, die Wörter und Phrasen in eine grammatisch korrekte Reihenfolge zu bringen. Sie müssen es erlauben, die in Abschnitt 6.2.2.1 genannten Reihenfolgebedingungen für das Verb und seine Dependenten kompakt darzustellen. Außerdem soll die Aktivierungsreihenfolge in der Linearisierungsphase nur dann verändert werden, wenn syntaktische Restriktionen dies erforderlich machen.

In ID/LP-Grammatiken wie GPSG [Gazdar *et al.*, 1985] werden Dominanz und lineare Abfolge durch getrennte Regelmengen dargestellt. Üblicherweise impliziert die Schreibweise

$$s \longrightarrow subj \;\; vp \;\; obj$$

daß die Konstituenten auf der rechten Seite der Regel genau in dieser Reihenfolge auftreten. In GPSG wird diese Ordnung aufgehoben und durch eine extra Regelmenge ausgedrückt. Die Regel

$$s \longrightarrow vp, obj, subj$$

drückt nur noch die Dominanz aus, während die LP-Regeln

$$\begin{aligned} subj &< vp \\ vp &< obj \end{aligned}$$

die Reihenfolge ausdrücken, wobei $<$ der binäre Reihenfolgeoperator ist. Jede kontextfreie Grammatik läßt sich in eine solche ID/LP-Grammatik transformieren, so daß diese Grammatiken keine Erweiterung gegenüber den kontextfreien Grammatiken sind. Die Notation erlaubt jedoch eine kompaktere Darstellung vor allem der freien Wortstellung.

Die LP-Regeln von POPELGRAM sind eine Erweiterung dieser binären Regeln. Die Notation entspricht der der ID-Regeln, der kontextfreie Teil wird jedoch als *vorschreibende Abfolgerelation* interpretiert. Die Notation

```
VCOMP FRONTFIELD V-FIN MAINFIELD V-INFIN RESTFIELD
```

besagt, daß in einer Verbalphrase `VCOMP` zuerst das Vorfeldelement kommen muß, dann der finite Verbteil, das Mittelfeld, der infinite Verbteil und am Ende das Nachfeld. Die in POPELGRAM verwendete Notation ist wesentlich kompakter als die binäre Schreibweise, bei der die Regelmenge für `VCOMP` 10 Regeln umfassen würde.

a) Eine Linearisierungsregel für Aussagesätze

```
(VCOMP FRONTFIELD V-FIN MAINFIELD V-INFIN RESTFIELD
       ((0  SYNTAX SENTENCE-FORM) DECLARATIVE)
       ((1  DEPENDENT) +)
       ((3  DEPENDENT) +)
       ((5  DEPENDENT) +))
```

b) Eine Linearisierungsregel für Nominalphrasen

```
(NP-ENG DET ADJP N
        ((0 ELISION) -)
        ((0 ANAPHER) -)
        ((2 DEPENDENT) +))
```

Abbildung 6.10: Zwei LP-Regeln

Neben den Wort– und Phrasenklassen können im kontextfreien Teil der Regel auf der rechten Seite auch die Metaklassen `ANY` und `MULTIPLE-ANY` auftreten. `ANY` besagt, daß an dieser Position jeder beliebige Dependent stehen kann, `MULTIPLE-ANY`, daß beliebig viele Dependenten stehen können. Der Spezifikationsteil der Regeln enthält zum einen Bedingungen, die die Anwendbarkeit der Regel steuern, und die die Merkmale der zu linearisierenden DAGs untersuchen. Zum anderen enthält er Merkmale, die charakterisieren, ob ein Element der LP-Regel sich autonom linearisiert.

Abbildung 6.10 zeigt zwei LP-Regeln für Aussagesätze und für einfache Nominalphrasen. Bei der Regel für `VCOMP` unifiziert die erste Gleichung des Spezifikationsteils nur mit DAGs, deren Merkmal `SENTENCE-FORM` den Wert `DECLARATIVE` besitzt. Die drei restlichen Gleichungen geben an, daß die DAGs, die das Vor-, Mittel- und Nachfeld besetzen, von anderen Objekten der DBS-Ebene gefüllt werden, die autonom linearisiert werden.

6.2.3 Die Verarbeitung in den syntaktischen Ebenen

6.2.3.1 Die DBS-Ebene

Die Verarbeitung der Objekte in der DBS-Ebene unterscheidet sich von der in den SB-ONE basierten Ebenen vor allem dadurch, daß die Objektstruktur nicht mit der Struktur der Wissensquelle übereinstimmt und daß die Aufgabe eines Objektes nicht mehr die regelbasierte Transformation in eine andere Ebene ist. Zusätzlich zur Objektstruktur enthalten die einzelnen Objekte die DAGs, die die syntaktischen Daten zu den Wörtern und Phrasen enthalten. Die Aufgabe eines Objekts ist es, mindestens einen DAG aufzubauen, der lokal vollständig ist, und diesen an die ILS-Ebene weiterzugeben.

Die Basisoperation bei der Konstruktion von DAGs ist ein Suchverfahren, das zwischen zwei DAGs, die in einer Dominanzrelation stehen, Grammatikregeln sucht, die es erlauben, die DAGs zu unifizieren, und das diese Unifikation durchführt. Diese Operation wird in der generischen Initialisierungsfunktion `define-initial-state` (siehe Abbildung 6.5) im Basisprogramm der DBS-Objekte benutzt, um aus dem Lexem und den ID-Regeln die initialen DAGs zu erzeugen. Es handelt sich i.a. um mehrere DAGs, da mehrere ID-Regeln anwendbar sind.

Die Überprüfung, ob zwei miteinander verbundene Objekte eine korrekte syntaktische Struktur aufbauen können, sowie der Transport morphosyntaktischer Merkmale erfolgt in der generischen Funktion `handle-changed-environment`. Zu den DAGs des regierenden Objekts werden mit Hilfe des Suchverfahrens ID-Regeln gesucht, die eine Unifikation mit den DAGs des abhängigen Objekts erlauben. Bei der Unifikation transportieren die Gleichungen der Spezifikationsteile die morphosyntaktischen Merkmale zwischen den DAG-Strukturen.

Falls ein Element des Kontexts des Objekts durch ein anderes, konkurrierendes ersetzt wurde, erfolgt hier auch die Modifikation der DAGs des Objekts. Durch die Angabe der Flußrichtung der Merkmale in dem Merkmal `SYNCHRONIZE` können die Informationen in den DAGs lokal zurückgesetzt werden.

Abbildung 6.11 zeigt einen Ausschnitt des DBS-Objekts für das Verb `fahr`, das bei der Generierung des Satzes

Ein Mann fährt mit einem Motorrad von Saarbrücken nach Völklingen.

aufgebaut wird. Der DAG für das Verb entstand aus der ID-Regel aus Abbildung 6.9. Bei der Unifikation mit dem Objekt, das die Funktion der Nominativergänzung `e0` im Kontext des `VCOMP`-Objekts übernimmt, erfolgte der Transport der `NUMBER`– und `PERSON`-Merkmale in die Merkmalsmenge der Phrase und damit des Verbs selbst. Gleichzeitig erhielt der DAG der Nominalphrase das `CASE`-Merkmal für den Kasus.

In der Instanzvariablen `WORD-CHOICE` sind die Merkmale zu sehen, die beim Aufbau der initialen DAGs des VCOMP-Objekts benötigt werden. Neben dem Wort selbst und dessen Wortklasse sind dies Informationen über Tempus, Modus, Satzform und solche zur Aktiv/Passivbildung. Zur Bestimmung der ersten beiden Merkmale wurden in POPEL keine weiteren Untersuchungen durchgeführt (siehe die Einleitung zu diesem Kapitel). Wenn im System Wissen zur Bestimmung beider Merkmale integriert werden soll, ist durch die deklarative Formulierung der Grammatik eine einfache Erweiterung möglich. Die Satzform wird von POPEL-WHAT bestimmt und bei der Erzeugung des CKB-Objekts, aus dem das Verb-Objekt mittelbar entstanden ist, diesem als zusätzliche Information mitgegeben. Die Entscheidung, ob ein Aktiv- oder Passivsatz gebildet wird, hängt von zwei Bedingungen ab. Die eine ist, ob das Verb überhaupt passivfähig ist.[5] Diese Information ist im morphosyntaktischen Lexikon des XTRA-Systems enthalten. Die zweite Bedingung ist die Reihenfolge, in der die Ergänzungen des Verbs aktiviert werden, d.h. die Verarbeitungsdynamik des Systems paralleler Prozesse. Erreicht das Objekt, das die Stelle der Akkusativergänzung in der Verbalphrase einnimmt, als erstes die DBS-Ebene, so erfolgt

[5] In POPEL wird nur die Bildung des *werden*-Passivs betrachtet [Engel, 1988, p.454].

```
#<DBS-OBJECT 42743520> is an instance of class
                            #<Standard-Class DBS-OBJECT 34233220>:
                           ...
 CONTEXT                (#S(ELEMENT-OF-CONTEXT ARC E0 FLAG ...) ...)
 WORD-CHOICE            #S(WORD-CHOICE MAIN
                           #S(ONE-WORD STEM "fahr" CAT VERB SPECIALS
                              ((SYNTAX ((VOICE ACTIVE)
                               (SENTENCE-FORM DECLARATIVE) ...)))))
                           ...
 DAGS                   (
"[0: [CAT: VCOMP
      FSET: [HEAD: [STEM: fahr]
             SYNTAX: [MOOD: INDICATIVE
                      NUMBER: SG
                      PERSON: 3
                      SENTENCE-FORM: DECLARATIVE
                      TENSE: PRESENT
                      VOICE: ACTIVE]
             SYNCHRONIZE: [DOWN: [VOICE: <0 FSET SYNTAX VOICE>
                                  NUMBER <0 FSET SYNTAX NUMBER>
                                  PERSON <0 FSET SYNTAX PERSON>]]
             SURFACE: [VCOMP: [V: [STEM: <0 FSET HEAD STEM>
                                   SYNTAX: <0 FSET SYNTAX>]]]]
      PROCESS: [ALLOWED-TO-SEND-P: -
                COMPLETE-P: +
                INDEX: 1
                SUBDAG-CHANGED-P: -]]
  1: [CAT: V
      FSET: [HEAD: <0 FSET HEAD>
             SYNTAX: <0 FSET SYNTAX>
             SURFACE: <0 FSET SURFACE VCOMP V>]]
  2: [CAT: E0
      FSET: [ANAPHER: -
             ELISION: -
             HEAD: [STEM: mann]
             SYNTAX: [ARTICLE: INDEF
                      CASE: NOM
                      GENDER: MAS
                      NUMBER: <0 FSET SYNTAX NUMBER>
                      PERSON: <0 FSET SYNTAX PERSON>
                      VOICE: <0 FSET SYNTAX VOICE>]
                            ... ]]
]")
 DAG-CATEGORY           VCOMP
```

Abbildung 6.11: Ein Ausschnitt eines DBS-Objekts für das Verb *fahr*

eine Passivbildung, damit dieses auch an erster Stelle im Satz stehen kann und durch die Passivbildung hervorgehoben wird.

Die Übergabe der syntaktischen Daten eines DBS-Objekts an die ILS-Ebene kann dann erfolgen, wenn für das Kopfelement alle notwendigen Informationen vorhanden sind. Das ist dann der Fall, wenn für das Kopfelement im DAG alle Merkmale einen Wert besitzen, d.h. wenn alle Merkmale des Teil-DAGs unter dem Merkmal `HEAD` nicht leer sind. Diese prozeßspezifischen Informationen werden im DAG unter dem Merkmal `PROCESS` vermerkt, auch um Mehrfachabbildungen zu verhindern. Das Kopfelement des DBS-Objekts aus Abbildung 6.11 konnte z.B. bereits abgebildet werden und darf, da sich nichts im `HEAD` DAG geändert hat, nicht mehr an die Linearisierungskomponente weitergegeben werden.

6.2.3.2 Die ILS-Ebene

Die ILS-Ebene hat die Aufgabe, die Lexeme eintreffender DAGs zu flektieren, eine syntaktisch korrekte Abfolge aufzubauen und diese an den Dialogpartner auszugeben. In POPEL-HOW wurde diese Ebene als eigenständige Verarbeitungsebene realisiert, die aus einem einzigen Prozeß besteht, an den alle DBS-Objekte die lokal vollständigen DAGs senden, zusammen mit Informationen über das zugehörige Objekt, regierende DAGs bzw. Objekte und die aus der Aktivierungsreihenfolge stammende Position.

Die Flektion wird von dem Morphologiepaket MORPHIX [Finkler and Neumann, 1988] durchgeführt, das die meisten Flexionsphänomene des Deutschen bearbeiten kann. Dazu werden die morphosyntaktischen Informationen aus dem DAG des Kopfelements und die unter dem Merkmal `SURFACE` stehenden Stämme an MORPHIX übergeben. Das Resultat ist eine präterminale Kette, die aus Paaren von Wortklassen und Lexemen besteht. Für eine Nominal– bzw. Verbalphrase sehen diese z.B. folgendermaßen aus:

```
((N "mann") (DET "der"))
((V-INFIN "ausgefuehrt") (V-FIN "wird"))
```

Aus der Information über die regierenden DAGs wird die dependentielle Struktur für alle DBS-Objekte (bzw. deren lokale DAGs und Wörter) rekonstruiert, die sich bereits in die ILS-Ebene abbilden konnten. Da nicht unbedingt zusammenhängende Strukturen in der ILS-Ebene vorhanden sein müssen, kann es mehrere solcher *Positionsbäume* geben.

Ein Positionsbaum wird mit Hilfe der LP-Regeln von oben nach unten und von links nach rechts abgearbeitet. Jeder Knoten des Baums entspricht einer Phrase, für die eine Linearisierungsregel ausgewählt werden muß, die die Grundlage für die Positionierung bildet. Die Anwendbarkeit wird überprüft, indem der Spezifikationsteil der LP-Regel mit dem lokalen DAG unifiziert wird. Der rechte Teil der ausgewählten LP-Regel schreibt vor, in welcher Reihenfolge die Wörter des Kopfelements und die der Dependenten geäußert werden müssen. Ist durch die Kategorien `ANY` oder `MULTIPLE-ANY` in der Regel kein bestimmter Dependent spezifiziert, der an dieser Stelle der Regel positioniert werden kann, werden ein bzw. mehrere Elemente gemäß ihrer Aktivierungsreihenfolge an dieser Stelle positioniert.

Derzeit ist die Ausgabe an den Benutzer so organisiert, daß die Lexeme, die mit den LP-Regeln linearisiert wurden, entweder sofort auf dem Bildschirm ausgegeben werden, oder erst nach der Terminierung von POPEL-HOW durch den Strukturaktivator. Bei

der sofortigen Ausgabe können, besonders wenn es mehrere Positionsbäume in der LP-Ebene gibt, teilweise unverständliche Äußerungen produziert werden, die nach und nach durch das Auffüllen von Lücken und das dadurch bedingte Zusammenwachsen der Positionsbäume verständlich werden. Ist die Ausgabe spontan, werden z.B. folgende Teilsätze produziert:

faehrt
ein mann faehrt
ein mann faehrt mit einem motorrad
ein mann faehrt mit einem motorrad von saarbruecken
ein mann faehrt mit einem motorrad von saarbruecken nach voelklingen

Dieses Problem kann durch eine zusätzliche Kontrolle gelöst werden, die nur zusammenhängende Positionsbäume ausgibt. Zusätzlich muß darauf geachtet werden, daß vor der Ausgabe eines bestimmten Knotens alle DBS-Objekte, die in der Aktivierungsreihenfolge vor diesem stehen, in die ILS-Ebene abgebildet und ausgegeben wurden. Die Ausgabe des Verbs ohne das vorangehende Subjekt in dem obigen Beispiel würde dadurch solange unterdrückt, bis auch das Subjekt verbalisiert ist.

Treten Reformulierungen oder Präzisierungen wie

Günter schreibt ... Günter und Wolfgang schreiben ...
Fredo fährt ... fliegt ...

auf, wenn inkrementell zusätzliches oder präzisierendes Wissen in die Realisierungskaskade eingegeben wird, wird bei der spontanen Ausgabe der bereits linearisierten Phrasen ebenfalls eine Kontrollfunktion benötigt, die die Ausgabe korrigiert.[6]

Wird eine Phrase ausgegeben, so heißt das, daß der Dialogpartner sich auf diese im weiteren Dialog beziehen kann und ein Eintrag in das Dialoggedächtnis LDM durchgeführt werden muß. Das LDM verbindet die konzeptuelle und funktionalsemantische Repräsentation einer verbalisierten Äußerung mit deren Position im Dialog. Während jedes CKB-Objekt ein Konzept in der A– oder T-Box der CKB repräsentiert, entspricht ein FSS-Objekt nur einem möglichen individualisierten Konzept in der FSS A-Box. Soll der Eintrag in das LDM für eine Phrase erfolgen, muß zunächst die mögliche FSS-Struktur individualisiert werden, damit im weiteren Verlauf des Dialogs Bezugnahmen auf diese Phrase möglich sind. Die Individualisierung kann erst dann vorgenommen werden, wenn für die DAGs, die aus der Abbildung des entsprechenden FSS-Objekts entstanden sind, in der ILS-Ebene die Verarbeitung durchgeführt wurde. Das bedeutet, daß auch die Einträge in das LDM inkrementell erfolgen, und dem Planer noch während der Generierung einer Äußerung die LDM-Einträge der bereits verbalisierten Teile derselben Äußerung für die weitere Verarbeitung zur Verfügung stehen, z.B. für die Bestimmung von intrasententiellen Anaphern und Ellipsen.

[6] Wurden die Inhalte bereits geäußert, müßten Korrekturphrasen wie "äh ..." erzeugt werden, falls die inkrementelle Ausgabe unmittelbar dem Benutzer präsentiert wird (vgl. zu diesem Thema [Levelt, 1989, p.478ff] und die Diskussion der hybriden Form der Kommunikation in [Schmauks and Reithinger, 1988])

6.3 Die Erzeugung und Behandlung von Anfragen

6.3.1 Bedingungen für das Stellen von Anfragen

Ein wichtiges Entwurfsziel von POPEL ist, daß der Inhaltsfestlegung von der Komponente zur Inhaltsrealisierung Unterstützung gegeben werden muß, wenn sprachliche Restriktionen zusätzliche Informationen aus der konzeptuellen Wissensquelle notwendig machen (siehe Abschnitt 3.4.2.3). Diese Rückwirkungen sollen so schnell wie möglich erfolgen, damit POPEL-WHAT möglichst frühzeitig entscheiden kann, ob zusätzliche Inhalte aktiviert werden sollen und damit die inkrementelle Verarbeitung nicht durch fehlende Daten verzögert wird (vgl. Abschnitt 5.5).

Jedes Prozeßobjekt in POPEL-HOW hat eine festgelegte Aufgabe: in den SB-ONE basierten Ebenen ist es die Abbildung auf die nächsttiefere Ebene, in der neue Objekte und Verbindungsstrukturen zwischen diesen Objekten erzeugt werden. In der DBS-Ebene werden lokal vollständige syntaktische Beschreibungen erzeugt, die an den Linearisierungsprozeß weitergegeben werden.

Die Abbildung in der CKB– und der FSS-Ebene erfolgt mittels Regeln, die feuern können, wenn die im Vorbedingungsteil geforderte Objekt– und Verbindungsstruktur vorhanden ist. Wenn keine dieser Regeln feuern kann, da sie Vorbedingungen enthalten, die nicht erfüllt sind, kann ein Objekt dieser Ebenen eine Anfrage stellen. Die nicht erfüllten Vorbedingungen liefern dazu die notwendigen Daten.

In der DBS-Ebene ist die Grundlage der Abbildung das lokale Vollständigkeitskriterium, das prüft, ob die syntaktische Beschreibung des Kopfelements vollständig ist. Der Ausgangspunkt für die Anfrage sind hier die undefinierten Merkmale und die Information darüber, woher diese stammen. Dazu werden die DAGs des Objektes daraufhin untersucht, welches Element des Dependenzrahmens diese Merkmale liefert und deshalb angefordert werden muß.

Bevor ein Objekt Anfragen stellen darf, muß zuerst sichergestellt werden, daß es bereits mit allen in der Ebene befindlichen Objekten, mit denen es in Verbindung treten könnte, diese auch wirklich aufgenommen hat. Sonst wäre es möglich, daß überflüssige Anfragen gestellt werden. Das Basisprogramm der Objekte stellt sicher, daß diese Bedingung erfüllt ist. Zudem darf ein Objekt aus den SB-ONE basierten Ebenen nur dann eine Anfrage stellen, wenn bisher keine Abbildung in die nächste Ebene durchgeführt worden ist. Da das Basisprogramm in einer Endlosschleife immer wieder durchlaufen wird, würden sonst für alle Vorbedingungen der Regeln eines Objekts Anfragen gestellt. Eine möglichst genaue Angabe an POPEL-WHAT, welche Daten fehlen, ist dann nicht mehr gegeben.

6.3.2 Die Bearbeitung von Anfragen

Die Behandlung von Anfragen erfolgt im Basisprogramm der Prozeßobjekte mit der generischen Funktion `requests` (siehe Abbildung 6.12). Die Funktion überprüft zunächst, ob eine Anfrage gestellt werden kann (Zeile 1) und erzeugt diese, falls möglich (3-17). Ferner werden in dieser Funktion Anfragen, die von Nachfolgerobjekten kommen, an das Vorgängerobjekt weitergeleitet (18-28). Diese Anfragen, die ein Objekt zur Weiterleitung

```
(1)   if not son-pointer (object)
(2)   then
(3)     if    dbs-object (object)
(4)     then
(5)           loop for dag in dags (object)
(6)           do
(7)                missing-context-element := paths-to-null-dags (dag)
(8)                create-request-at-father (object, missing-context-element)
(9)           endloop
(10)    else
(11)          loop for rule in get-rules (object)
(12)          do
(13)               missing-context-element :=
                     find-unsuccessful-precondition (object, rule)
(14)               create-request-at-father (object, missing-context-element)
(15)          endloop
(16)    endif
(17)  endif
(18)  if new-requests (object)
(19)  then
(20)    loop for request in new-requests (object)
(21)    do
(22)          if    not-yet-satisfied (request)
(23)          then
(24)               missing-context-element :=
                     find-unsuccessful-precondition (object, rule (request))
(25)               create-request-at-father (object, missing-context-element)
(26)          endif
(27)    endloop
(28)  endif
```

Abbildung 6.12: Der Algorithmus zur Behandlung von Anfragen

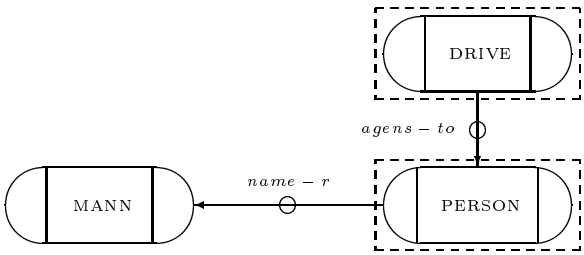


Abbildung 6.13: Der Ausschnitt aus der A-Box der CKB für das Beispiel

erhält, werden nicht sofort beim Eintreffen bearbeitet, sondern zunächst in der Variablen `NEW-REQUESTS` gespeichert.

Objekte der SB-ONE basierten Ebenen, die regelbasiert arbeiten, haben für das Stellen der Anfragen ein gemeinsames Regelauswahl– und Verarbeitungschema. Aus den einfachen und kombinierten Regeln werden diejenigen Regeln ausgewählt, für die noch keine Anfrage erzeugt wurde, die die geringsten Vorbedingungen stellen, und deren Vorbedingungen vom unterliegenden SB-ONE Netz erfüllt werden können (13). Aus diesen Regeln werden die Konzepte und Rollen des Bedingungsteils extrahiert, die die Grundlage der Anfrage bei Vorgängerobjekt bilden. CKB-Objekte reichen die extrahierten Konzepte und Rollen der CKB ohne weitere interne Verarbeitung direkt an POPEL-WHAT weiter, wo dann entscheiden werden muß, ob die angeforderten Elemente verbalisiert werden sollen (14).

Abbildung 6.13 zeigt einen Ausschnitt der A-Box der konzeptuellen Wissensquelle, der als Satz

Der Mann fährt.

verbalisiert werden kann. Der Planer hat, durch die gestrichelten Rahmen angedeutet, Objekte für `DRIVE` und `PERSON` erzeugt. Das individualisierte Konzept `PERSON` kann jedoch nur zusammen mit dem Rollenfüller von `name-r` abgebildet werden, in diesem Beispiel zusammen mit dem individualisierten Konzept `MANN`. Die Anfrage, die daraufhin erzeugt wird, ist in Abbildung 6.14 (a) dargestellt, zusammen mit der Information, die die Komponente zur Behandlung von Anfragen erhält. Da die Regeln immer über Elemente der T-Box formuliert sind, enthalten die Anforderungen generelle Konzepte und Rollen. Das anfordernde Objekt teilt jedoch mit, ob Elemente der A– oder T-Box benötigt werden, so daß auch die entsprechenden individualisierten Elemente aktiviert werden können.

Ein FSS-Objekt stellt die Anfragen bei seinem Vorgängerobjekt[7] in der CKB-Ebene (14). Dazu greift es auf das Regelpaket des CKB-Objektes zu und extrahiert alle Regeln, deren zentrales Konzept im Aktionsteil dem eigenen FSS-Konzept entspricht. Befinden sich Erweiterungsregeln im Regelpaket, die im Aktionsteil die benötigten FSS-Objekte und Rollen erzeugen, wird an das CKB-Objekt die Anfrage gestellt, diese Regeln zu feuern. Kann keine passende Erweiterungsregel gefunden werden, so werden die einfachen und kombinierten Regeln auf die benötigten Elemente in den Aktionsteilen hin untersucht und es wird eventuell eine Anfrage initiiert. Während bei Erweiterungsregeln das

[7] Oder den Vorgängerobjekten, falls mehrere Objekte gemeinsam dieses Objekt erzeugt haben.

a) Die Anfrage des CKB-Objekts für PERSON

```
PERSON-<CKB-IC25> does an own-request for its rule
#S(RULE LHS
        (#S(ROLE&VR ROLE name-r VR MANN))
        RHS
        #S(RHS CONCEPT *mann* PATH-MAPPINGS NIL) ... )
                  ...
REQUEST HANDLER:
   Role name-r for Obj "PERSON-<CKB-IC25>" and VR MANN wanted
```

b) Die Anfrage eines syntaktischen Objekts

```
fahr-<DBS-VCOMP-1259> starts own-request
fahr-<DBS-VCOMP-1259> did a father-request to get the role E0
*FAHR*-<FSS-GC94-1258> is requested to fire the rule
#S(RULE LHS (agent)
        RHS
        #S(RHS CONCEPT VCOMP PATH-MAPPINGS
               (#S(PATH-MAPPING NEW-ROLE E0 FILLER-PATTERNS
                      ((SELF agent OBJECT))))) ...)
```

c) Die Weiterleitung der Anfrage durch das FSS-Objekt

```
*FAHR*-<FSS-GC94-1258> starts a called-request with
 new-requests:
 (#S(REQUEST TYPE MISSING-ROLE CONTENTS
             #S(RULE LHS (agent)
                     RHS
                     #S(RHS CONCEPT VCOMP PATH-MAPPINGS
                        (#S(PATH-MAPPING NEW-ROLE E0
                            FILLER-PATTERNS
                            ((SELF agent OBJECT))))) ...)
             REQUESTOR
             #<DBS-OBJECT 44205774> CONTEXT NIL))

*FAHR*-<FSS-GC94-1258> did a pattern-request:
PERSON-<CKB-IC25> is requested to fire the rule
#S(RULE LHS
      (#S(ROLE&VR ROLE name-r VR MANN))
      RHS
      #S(RHS CONCEPT *mann*
            PATH-MAPPINGS NIL) ...)
```

Abbildung 6.14: Beispiele für Anfragen

anfragende FSS-Objekt nur erweitert wird, erzeugen die anderen Regeln konkurrierende FSS-Objekte zum anfragenden Objekt, falls beim CKB-Objekt die Vorbedingungen erfüllt werden können.

Die Anfragen bei DBS-Objekten basieren auf der Untersuchung der DAGs des Objekts und der undefinierten Merkmale in deren Kopfelementen (3-9). Eine Anforderung zusätzlicher Information hängt davon ab, welcher der Dependenten die fehlenden Merkmale liefern kann (7). Aufgrund der Gleichungen im Spezifikationsteil kann dieser Dependent gefunden und bei dem FSS-Objekt angefordert werden, das das DBS-Objekt erzeugt hat. Dazu wird, analog zur Anfrage eines FSS-Objekts bei einem CKB-Objekt, ebenfalls dessen Regelpaket auf eine Regel hin untersucht, in deren Aktionsteil sich die gewünschten Strukturen befinden (8).

Nimmt man wieder die A-Box Struktur aus Abbildung 6.13 als Beispiel, und aktiviert man zunächst nur das individualisierte Konzept `DRIVE`, so wird ein FSS-Objekt erzeugt, das wiederum auf ein DBS-Objekt abgebildet wird. Erfolgt keine weitere Aktivierung von Objekten in POPEL-HOW, so kann sich das DBS-Objekt nicht weiter abbilden und stellt an das FSS-Objekt eine Anfrage. Damit das Verb sich abbilden kann, benötigt es Person- und Numerusinformation aus einer Nominativergänzung (`e0`), wenn eine Grammatikregel wie in Abbildung 6.9 zugrundegelegt wird. Das DBS-Objekt kann bei seinem Vorgängerobjekt eine Anfrage wie in Abbildung 6.14 (b) stellen. Das FSS-Objekt erhält die Aufforderung, die Regel zu feuern, die in ihrem Bedingungsteil das Vorhandensein der Rolle `agent` erforderlich macht und die in ihrem Aktionsteil die Kommunikationsverbindung für `e0` anlegt.

Neben dem *Erzeugen* von Anforderungen müssen die Objekte auch Anforderungen von Objekten aus tieferen Ebenen *weiterleiten*, so wie das FSS-Objekt im vorigen Beispiel. Findet ein Objekt eine Anfrage in seiner Instanzvariablen `NEW-REQUESTS`, wird zuerst überprüft, ob diese Anfrage noch bearbeitet werden muß, oder ob durch neu von POPEL-WHAT aktivierte Information und deren Bearbeitung diese Anfrage inzwischen überflüssig geworden ist (22). In der Anforderung ist eine Regel enthalten, deren Anwendung die auf der Nachfolgerebene benötigten Strukturen erzeugt (24). Die Anforderung kann dann ignoriert werden, wenn diese Regel in der Zwischenzeit aktiviert wurde oder durch eine andere Regel die Strukturen erzeugt wurden. Muß die Anfrage bearbeitet werden, so wird versucht, die Regel der Anforderung anzuwenden. Sind in der Regel Bedingungen enthalten, die nicht erfüllt werden können, so stellt das Objekt, das die Anforderung bearbeitet, selbst eine Anfrage, die die in der Regel benötigten Elemente bei der nächsthöheren Ebene bzw. bei POPEL-WHAT anfordert (25).

Abbildung 6.14 (c) zeigt, wie die Anfrage des DBS-Objekts aus (b) vom FSS-Objekt weitergeleitet wird. Die Anfragen, die in dieser Abbildung dargestellt sind, stammen aus der Verbalisierung des individualisierten Konzepts `DRIVE` mit dem Ziel

```
(SWMB (KNOW H DRIVE))
```

Die Verzahnung von Planung und Realisierung führt dazu, daß das Verb bereits in der DBS-Ebene vorhanden ist, während die Planung des Füllers der Rolle `agens-to` die Aktivierung des Konzepts `PERSON` zur Folge hat. Während das FSS-Objekt die Anfrage aus

der DBS-Ebene weiterleitet, stellt das CKB-Objekt für `PERSON` die in (a) gezeigte Anfrage, die zur Aktivierung des Konzepts `MANN` führt. Die Anfrage des FSS-Objekts wird vom CKB-Objekt ignoriert, da zwischenzeitlich die Regel bereits angewendet werden konnte (siehe auch Abschnitt 8.2.3).

6.4 Die Terminierung von POPEL-HOW

Das Basisprogramm der Prozeßobjekte läuft in einer Endlosschleife und sorgt dafür, daß ein Objekt auch dann noch arbeitet, wenn es sich bereits abgebildet hat. Es muß jedoch immer in der Lage sein, auf Situationsänderungen in seinem lokalen Kontext zu reagieren, wenn durch die inkrementelle Arbeitsweise von POPEL neue Objekte entstehen. Die einzelnen Objekte haben keine Daten darüber, wann ein Generatorlauf zu Ende ist, und können daher nicht selbst über den Zeitpunkt entscheiden, wann die Verarbeitung beendet werden kann.

Die Schleife darf nur dann abgebrochen werden, wenn diese Entscheidung von einer externen Instanz mitgeteilt wird. Der Strukturaktivator von POPEL-WHAT entscheidet, wann genügend konzeptuelle Einheiten ausgewählt und aktiviert wurden, die in einer Äußerung dem Dialogpartner mitgeteilt werden sollen. Mit der Funktion `terminator-go` initiiert er die Terminierung der Prozeßobjekte in POPEL-HOW. Das Terminierungssignal wird durch die Kommunikationsverbindungen zwischen den Objekten einer Ebene und über die Verbindungen in die Nachfolgerebene allen Objekten mitgeteilt. Die Verarbeitung in den Objekten bricht nicht sofort ab, wenn das Signal eintrifft. Solange es noch Objekte gibt, die ihre Kommunikationsverbindungen nicht aufgebaut haben und sich noch nicht abgebildet haben, wird das Signal in dieser Ebene nicht weitergeleitet. Zudem werden von den Objekten in allen Ebenen noch eventuell vorhandene Anfragen bearbeitet. Erst wenn diese Bedingungen erfüllt sind, terminieren die Objekte.

Objekte der DBS-Ebene, die sich nicht weiter abbilden konnten, werden in der Terminierungsphase speziell untersucht. Gibt es zu einem DBS-Objekt kein konkurrierendes Objekt, das sich in die ILS-Ebene abbilden konnte, muß die dem Objekt fehlende Information aus dem Dialogkontext abgeleitet werden, damit Äußerungen wie

> BEN: *Ich fahre täglich von Saarbrücken nach Völklingen zur Arbeit.*
> SYS: *Mit dem Bus?*

generiert werden können.

In Abschnitt 3.4.2.3 wurde anhand dieses Beispiels bereits dargestellt, welche Verarbeitungsschritte notwendig sind, um die Präpositionalphrase erzeugen zu können. Die fehlenden syntaktischen Merkmale des Kopfelements werden bestimmt und es wird, je nach Typ des Kopfelements, eine Anfrage nach dem fehlenden Prädikat und der Argumentstelle gestellt, die das DBS-Objekt einnehmen kann, oder nach dem Argument, das als Nominativergänzung des Verbs fungieren kann. Die Komponente zur Behandlung von Anfragen bestimmt aus dem LDM diese FSS-Elemente und gibt sie an das anfragende Objekt zurück. Dieses berechnet daraus und den Einträgen des semantischen Lexikons die fehlenden syntaktischen Merkmale. Für die DBS-Objekte, die die Anfragen gestellt

haben, werden die passenden ID-Regeln angewendet. Die Kopfelemente, die danach voll spezifiziert sind, werden in die ILS-Ebene abgebildet, dort flektiert und ausgegeben.

Diese Anfragen in der Terminierungsphase werden nicht wie die Anfragen nach zusätzlicher Information im Basisprogramm der Objekte asynchron über die Zwischenebenen transformiert und an POPEL-WHAT übergeben. Es erfolgt ein direkter, synchroner Zugriff auf die Vorgängerobjekte und die Komponenten von POPEL-WHAT, die die benötigte Information liefern können, da angenommen werden kann, daß die dazwischenliegenden Objekte ihre Aufgabe bereits erfüllt haben und ihr Basisprogramm bereits terminiert ist.

Kapitel 7

Die Generierung multimodaler Referenzen

Die bisherige Darstellung von POPEL hat das Problem ausgeklammert, wie Referenzausdrücke bestimmt werden. In Abschnitt 3.4.2.2 wurde diese Bestimmung als einer der Punkte genannt, an dem eine Interaktion zwischen der Inhaltsfestlegung und Inhaltsrealisierung stattfindet. In diesem Kapitel wird gezeigt, welches Wissen und welche Verarbeitungsmethoden in POPEL verwendet werden, um Konzepte so zu spezifizieren, daß der Hörer sie identifizieren kann. Neben den sprachlichen Mitteln verfügt POPEL auch über extralinguistische Mittel, nämlich Zeigegesten, deren Gebrauchsvoraussetzungen und Generierung ebenfalls vorgestellt werden.

7.1 Die grundlegenden Begriffe

In Dialogen beziehen sich die jeweiligen Sprecher mit ihren Äußerungen auf Objekte, die bereits erwähnt oder nicht erwähnt, bekannt oder unbekannt, sichtbar oder nicht sichtbar sein können. Natürliche Sprachen verfügen über eine Vielzahl von Mitteln, um mit Referenzausdrücken Bezüge zu Objekten in einem realen oder imaginären Kontext herzustellen. Dabei hängt es von der Dialogsituation ab, welche sprachlichen und außersprachlichen Mittel jeweils verwendet werden. Vor allem zur Einführung sichtbarer Diskursobjekte werden sprachliche Äußerungen oft von Zeigegesten begleitet. Jeder der drei folgenden Sätze kann in einer bestimmten Dialogsituation angemessen sein, wobei '↗' für eine die Sprache begleitende Zeigegeste steht:

Ein Mann fährt mit einem Motorrad nach Völklingen
Dorthin fährt er mit dem Motorrad.
Damit ↗ fährt er dorthin.

Ein Generierungssystem muß also entscheiden können, welche sprachliche Form es für ein Element seines konzeptuellen Wissens wählen muß, so daß der Dialogpartner mit hoher Wahrscheinlichkeit dieses Objekt identifizieren kann. In [Dale, 1988, p.52ff] wird

eine ausführliche Übersicht über Ansätze gegeben, die bisher in Generierungssystemen zum Einsatz gekommen sind. Von besonderem Interesse sind dabei die Entscheidungen, in welcher Form auf ein bereits in den Diskurs eingeführtes Objekt referiert werden soll. Bereits in [Wong, 1975] wird ein Algorithmus beschrieben, der entscheidet, wann man aufgrund syntaktischer und semantischer Kriterien ein Pronomen generieren kann. Das System HAM-ANS verwendet eine sog. 'Antizipations-Rückkopplungsschleife', in der ein Referenzausdruck, der generiert werden soll, zuerst durch das Analysemodul verarbeitet wird, um aus dem Analyseergebnis zu sehen, ob angenommen werden kann, daß der Benutzer das intendierte Objekt identifizieren kann [Jameson and Wahlster, 1982]. Wenn dies nicht der Fall ist, wird der Ausdruck solange modifiziert, bis das Analyseergebnis eindeutig ist. Dieses Vorgehen entspricht in etwa der Rückwirkung in Levelts Modell der Spracherzeugung [Levelt, 1989, p.467ff]. Das System KAMP [Appelt, 1985, p.114ff] plant *Concept Activations*, die sowohl sprachliche Referenzausdrücke als auch physikalische Aktionen wie z.B. Gesten beinhalten können.

7.1.1 Terminologische Grundlagen

Unter *Referenz* versteht man die Beziehung zwischen einem Objekt oder mehreren Objekten in der Welt und sprachlichen Ausdrücken oder Elementen eines sonstigen Zeichensystems, die dieses Objekt 'bezeichnen' sollen. In der Philosophie und Linguistik existieren viele Arbeiten, die sich z.B. damit beschäftigen, ob Referenz und Bedeutung gleichzusetzen sind, womit und worauf man referieren kann.[1] Im Kontext dieser Arbeit wird von einem vereinfachten Referenzbegriff ausgegangen, der sich an den Rahmenbedingungen eines KI-Systems orientiert.

Auf ein *Referenzobjekt* kann durch sprachliche *Referenzausdrücke* referiert werden, die unter bestimmten Voraussetzungen von Gesten begleitet werden können. An den folgenden drei Sätzen sollen die einzelnen Begriffe erläutert werden:

Tom ist auch hier.
Schau, der ↗ ist es.
Er hat eine Brille auf der Nase.

Im ersten Satz wird die Person mit einem Eigennamen eingeführt. Mit dem Demonstrativpronomen und der begleitenden Zeigegeste wird er im visuellen Kontext lokalisiert. Der dritte Satz referiert mit dem Pronomen auf den bereits erwähnten Antezedens und gibt eine kennzeichnende Eigenschaft an.

Im Rahmen von XTRA können Referenzen auf das konzeptuelle Wissen des Systems verarbeitet werden, auf das jede Eingabe abgebildet werden muß und das den Ausgangspunkt der Generierung bildet. Nur auf Objekte, die in der T– oder A-Box der CKB repräsentiert sind, kann erfolgreich referiert werden. Der sich im Verlauf des Dialogs entwickelnde Dialogkontext und dessen Struktur gehört im Prinzip ebenfalls zu den Objekten, auf die Bezüge möglich sein sollten, wie z.B.

Das habe ich Dir alles schon gesagt!

[1] Vgl. z.B. [Dale, 1988, p.14ff] für eine Diskussion.

wo *das* auf einen Dialogausschnitt referiert. Da die Repräsentation des Dialogverlaufs jedoch außerhalb der konzeptuellen Wissensquelle erfolgt, sind solche Referenzen nicht möglich.

7.1.2 Referenz mit sprachlichen Mitteln

Sprachliche Referenzausdrücke, die hier betrachtet werden, sind Nominalphrasen und Präpositionalphrasen, sowie die Prowörter, die an Stelle dieser Phrasen stehen können. Nach der Aufgabe, die diese Ausdrücke übernehmen, kann man drei Arten von Referenz unterscheiden, wobei hier lediglich die Formen der Nominalphrasen aufgelistet werden, da Präpositionalphrasen als Phrasen mit eingebetteter Nominalphrase interpretiert und behandelt werden:

- Indefinite Referenzen werden verwendet, um neue Elemente der Welt in den Diskurs einzuführen. Sie werden im Deutschen mit einer Nominalphrase mit unbestimmten Artikel oder einem Indefinitpronomen realisiert, z.B.

  Ein Mann fährt mit einem Motorrad.

- Definite Referenzen beziehen sich auf explizit oder implizit Bekanntes und werden mit einer Nominalphrase mit bestimmtem Artikel, mit einem Prowort oder als Eigenname realisiert, z.B.

  Dieser Mann fährt nach Völklingen.
  Er kauft dort ein Buch.

- Generische Referenzen beziehen sich auf Klassen oder einen Begriff und sind nicht an eine bestimmte Form gebunden. Sie können sowohl definit als auch indefinit realisiert werden, z.B.

  Eine Steuerhandlung wird von einer Person ausgeführt
  Steuerhandlungen werden von einer Person ausgeführt
  Die Steuerhandlung wird von einer Person ausgeführt

Diese Aufzählung berücksichtigt nur den *referentiellen* Gebrauch von Nominalphrasen, nicht den *prädikativen*, wie er in dem Satz

Paulchen ist der Mann am Drucker.

mit *der Mann am Drucker* auftritt.

Wird definit auf ein dem Dialogpartner bekanntes Objekt referiert, muß das Hauptaugenmerk darauf gerichtet werden, daß die sprachliche Realisierung die Dekodierung und die Identifikation beim Partner möglichst leicht macht. Die Abstufung der Mittel, die natürliche Sprachen bieten, von Pronomen über die Verwendung demonstrativer Determinatoren bis hin zu vollständig beschreibenden Nominalphrasen, ermöglicht es dem Sprecher, dem Dialogpartner genaue Hinweise zu geben, wo im Kontext er nach dem Referenzobjekt suchen soll.

Die Beschränkungen für die Struktur eines Referenzausdrucks sind teils syntaktisch-semantischer Art, teils stammen sie aus der Dialogstruktur und dem angenommenen Benutzerwissen. Die Verwendung eines Pronomens hängt z.B. nicht nur von der Eindeutigkeit von Person und Numerus im lokalen Kontext ab, sondern auch von der Position im Satz, an der es stehen soll. Die Pronomen stehen im Deutschen i.a. direkt hinter dem finiten Verb und können nicht nach einer vollständigen Nominalphrase im Mittelfeld stehen, auch wenn es einen eindeutigen Antezedenten gibt:

(1) *Der Mann will frischen Labskaus.*
(2a) *Er kauft ihn in Hamburg.*
(2b) * *Er kauft in Hamburg ihn.*

Wird in einem inkrementellen Generator zuerst die syntaktische Struktur für *Hamburg* festgelegt, kann das Konzept für *Labskaus* nicht mehr als Pronomen verbalisiert werden. Im Gegensatz zu einem inkrementellen System wie POPEL muß in einem System, das nicht inkrementell arbeitet, auf Beschränkungen dieser Art nicht geachtet werden.

7.1.3 Multimodale Referentenidentifikation

Das System XTRA ermöglicht die Interaktion mit einem Dialogpartner nicht nur mittels Sprache, sondern es können auch Zeigegesten verwendet werden, um auf Elemente eines gemeinsamen visuellen Kontexts zu referieren. In Abschnitt 3.2.4 wurde als eine Anforderung an den Entwurf von POPEL genannt, daß das System Zeigegesten generieren soll. Ferner wurden einige Vorteile des Einsatzes von Gesten aufgeführt. In diesem Abschnitt werden die Grundlagen und Probleme dargestellt, die sich aus dieser Anforderung ergeben (vgl. auch [Schmauks and Reithinger, 1988]). Eine vollständige Darstellung multimodaler Referentenidentifikation findet sich in [Schmauks, 1991].

7.1.3.1 Grundlagen für den Gebrauch von Zeigegesten

Multimodale Interaktionen kombinieren Elemente aus verschiedenen Zeichensystemen. Hier werden die Zeichensysteme *Sprache* und *Zeigegesten* betrachtet, die zusammen verwendet werden, um auf Elemente eines den Dialogpartnern gemeinsamen visuellen Kontexts zu referieren. Die Elemente der Zeichensysteme haben hierbei komplementäre Funktionen. In Ausdrücken wie

diese Fahrkosten ↗

kategorisiert der sprachliche Ausdruck das Referenzobjekt, die Geste *lokalisiert* es im visuellen Kontext. Sind in der lokalisierten Region mehrere Objekte der gleichen Klasse vorhanden, so muß der sprachliche Teil durch Hinzufügen zusätzlicher Information eine Disambiguierung gewährleisten.

Neben dem Standardfall, in dem Zeigegeste und die begleitende Phrase das gleiche Objekt identifizieren, gibt es Fälle, in denen das Objekt, auf das gezeigt wird und das intendierte Referenzobjekt nicht identisch sind. Auch hier muß die Phrase die Identifizierbarkeit gewährleisten. Zwei dieser Fälle sind das *verschobene Zeigen*, bei dem auf ein

angrenzendes Gebiet zum ursprünglich intendierten gezeigt wird, um dieses nicht zu verdecken, und die *pars-pro-toto Deixis*, bei der auf einen Teil des Gesamtobjekts gezeigt wird, aber das gesamte gemeint ist.

Zeigeaktionen können mit unterschiedlichen *Bewegungsabläufen* und verschiedenen *Zeigemitteln* realisiert werden. Gesten können auf einen Punkt im visuellen Kontext gerichtet sein, der damit selektiert werden soll, oder mit einer komplexen Bewegung größere Ausschnitte überdecken, oder nacheinander mehrere Punkte lokalisieren. Zeigemittel können z.B. Finger oder Stifte sein, wobei damit die Reichweite oder die Präzision der Zeigegeste verändert werden kann (siehe [Schmauks, 1991, p.75ff] für eine weitergehende Diskussion).

Zur Realisierung der Zeigegesten in der Mensch-Maschine-Interaktion können zwei unterschiedliche Strategien herangezogen werden, von denen die eine den Gebrauch natürlicher Gesten *simuliert*, während die zweite, *performanzorientierte*, die Realisierung des Ziels der multimodalen Referentenidentifikation durch beliebige Mittel erreichen will.

Bei der *simulationsorientierten* Strategie ähneln die verwendeten Zeigegesten den natürlichen Zeigegesten in vielen Aspekten. Die Eingabe einer Geste kann z.B. durch berührungssensitive Bildschirme erfolgen, oder, als partielle Simulation, durch die Positionierung eines speziellen Ikons mit einer Maus oder einem Trackball. Die Ausgabe kann in diesem Ansatz ebenfalls durch die Positionierung eines speziellen Ikons in der Graphik erfolgen und damit die Geste eines Menschen simulieren.

Die *performanzorientierte* Strategie verwendet alle technischen Möglichkeiten, mit denen Teile der Graphik optisch herausgehoben werden können und die Aufmerksamkeit des Dialogpartners hervorgerufen wird, z.B. durch Invertieren, Umrahmen oder Blinken. Diese Mittel können nicht nur auf der Ausgabeseite verwendet werden, sondern es kann damit auch auf der Eingabeseite eine direkte visuelle Rückmeldung an den Dialogpartner erfolgen. Damit stellt sich der Analyse das Problem mehrdeutiger Zeigegesten nicht mehr, wie sie bei der Analyse simulierter natürlicher Zeigegesten auftreten können.

7.1.3.2 Die Verarbeitung von Zeigegesten in POPEL

In XTRA, und damit in POPEL, wurde trotz der scheinbaren Überlegenheit des performanzorientierten Ansatzes der simulationsorientierte Ansatz realisiert. Dieser erfordert nicht, daß im System eine vollständige interne Repräsentation der visuellen Struktur und des konzeptuellen Inhalts der dargestellten Graphik vorhanden ist, wie dies benötigt wird, um jedes visuell unterscheidbare Element der Graphik ansprechen zu können. Wenn es sich bei der Graphik gar um Bilder handelt, die mit Scannern eingelesen werden, müssen diese erst extrem aufwendig aufbereitet werden, damit sie für performanzorientierte Zeigegesten handhabbar werden. Beispielsweise muß die exakte Form einzelner Objekte identifiziert werden, damit diese dann, wenn darauf gezeigt wird, umrahmt oder andersfarbig dargestellt werden können. Zur Simulation natürlichen Zeigens hingegen ist es nur notwendig, eine näherungsweise Beschreibung der Graphik zu erstellen, so daß bei dem Anschluß an eine neue Domäne mit einer neuen Graphik der Anpassungsaufwand gering gehalten wird. Zudem können in diesem Ansatz verschobenes Zeigen und pars-pro-toto Deixis verwendet werden, so daß besonders bei einem Dialog mit Nichtexperten, die den Aufbau und

die Zusammenhänge der Graphik nicht kennen, die natürlichen Zeigestrategien verwendet werden können.

Während der visuelle Kontext in natürlichen Dialogen der dreidimensionale Raum ist, beschränkt er sich in der Dialogsituation, in der POPEL operiert, auf eine zweidimensionale, auf einem Bildschirm dargestellte Graphik, die aus dem Aufgabengebiet des Expertensystems stammt. Derzeit ist dies ein Formular aus dem Lohnsteuerjahresausgleich, auf dessen Felder gezeigt werden kann. Die Repräsentation der internen Struktur erfolgt durch die *Formularhierarchie* [Allgayer, 1986], die die geometrischen und logischen Strukturen der Teile des Formulars enthält. So gibt sie an, welche Felder des Formulars nebeneinanderliegen und welche zusammen zu einer bestimmten Kategorie gehören, z.B. zu 'Werbungskosten'. Mit einem Kopplungssystem wird die Verbindung zwischen der Formularhierarchie und Konzepten der CKB hergestellt, wenn zu einem Ausschnitt der Formularhierarchie ein entsprechendes Konzept vorhanden ist.

Die Eingabe von Gesten und deren Analyse wird von dem System TACTILUS-II [Allgayer, 1986, Wille, 1989] durchgeführt. Die Eingabe erfolgt mittels der Maus, mit der der Benutzer ein Ikon für das Zeigemittel auf der Graphik bewegen kann. Wird die Geste beendet, berechnet TACTILUS-II, auf welche Regionen des Formulars mit welcher Plausibilität gezeigt worden ist. Die Plausibilität wird aus der Größe der gezeigten Region und dem gewählten Zeigemittel (s.u.) berechnet. Mittels eines Propagierungsalgorithmus durch die Formularhierarchie wird nicht nur die direkt gezeigte Region untersucht, sondern auch einbettende, um pars-pro-toto Gesten verarbeiten zu können.

Empirische Untersuchungen [Wille, 1989] haben ergeben, daß als Zeigemittel Ikone, die dem Zeigen mit dem Bleistift und mit dem Finger entsprechen, von den Versuchspersonen am häufigsten benutzt wurden, diese jedoch frei bewegbar sein müssen, um dem natürlichen Zeigen möglichst nahe zu kommen. Zudem wurde bei diesen Untersuchungen der Gebrauch von *fokussierenden Gesten* festgestellt, mit denen der gerade bearbeitete Bereich eingegrenzt wird, sowie ein häufiges Auftreten von pars-pro-toto Deixis und verschobenem Zeigen.

7.1.3.3 Probleme bei der Generierung von Zeigehandlungen

Die bei der Analyse problematische Mehrdeutigkeit von Zeigehandlungen tritt bei deren Generierung aus der Sicht des Generators nicht auf. Das System hat Zugriff auf das gesamte konzeptuelle und geometrische Wissen, das das Formular beschreibt. Deshalb kann es immer eine Zeigegeste genau auf die Region produzieren, auf die das System referieren will, zusammen mit einem voll spezifizierten Referenzausdruck.

Diese totale Präzision kann jedoch hinsichtlich zweier Aspekte dem Benutzer nicht ausreichend angepaßt sein. Wenn das System entscheidet, wie es welche Medien einsetzt, um eine kommunikativ adäquate Reaktion zu generieren, muß es die Reaktionen des Dialogpartners berücksichtigen. Die multimodale Referentenidentifikation bietet z.B. die Möglichkeit, mit einem Laien durch eine sprachlich unterspezifizierte Phrase zu interagieren, und damit Fachtermini zu umgehen, wenn er diese nicht kennt. Für den Experten könnte die Zeigegeste aber zu wenig Information bieten, so daß hier ihr Einsatz nicht angebracht ist. Ein weiteres Problem ist, daß die Geste vom Dialogpartner falsch interpretiert

werden kann, und eine nichtexakte, z.B. verschobene Geste, korrekt interpretiert würde. Das System muß bei der Generierung diese benutzerspezifische Interpretation berücksichtigen.

Bei Zeigegesten, die in der Kommunikation zwischen Menschen verwendet werden, wird i.a. visuell kontrolliert, ob der Geste Aufmerksamkeit geschenkt wird. Ist dies nicht der Fall, kann sie explizit mit Äußerungen wie

Schau her! Da ↗!

angefordert werden. Menschen können gleichzeitig Informationen über mehrere Kanäle empfangen, z.B. optische, akustische und thermische Reize, und darauf mit Sprache und Gesten reagieren. Im Gegensatz dazu können Computersysteme bislang nur rudimentär z.B. visuelle Signale verarbeiten. Dies macht es unmöglich, die Reaktion des Dialogpartners zu beobachten.[2] Besteht bei der Generierung multimodaler Äußerungen keine visuelle Kontrolle, ob die Zeigegeste vom Dialogpartner beachtet wurde, muß das System die Möglichkeit berücksichtigen, daß die Zeigegeste nicht beachtet wurde, insbesondere dann, wenn die Ausgabe der begleitenden Phrase nicht akustisch, sondern textuell auf dem Bildschirm erfolgt. Die Informationen, die in natürlicher Kommunikation über zwei Kanäle mitgeteilt werden, werden hier in zwei Modi in einem Kanal übermittelt. Die Gefahr des Nichterkennens der Geste erhöht sich noch, wenn diese nicht punktuell ist, sondern sich aus komplexen Bewegungen zusammensetzt.

In der Simulation natürlichen Zeigens kann man dieses Problem teilweise lösen, indem man bei einer punktuellen Geste in einem Dialogschritt das Zeigemittel an der Stelle fixiert, auf die gezeigt werden soll. Komplexe Gesten können durch das Hinterlassen einer Spur markiert werden, was z.B. dem Unterstreichen oder Einkreisen eines Objekts mit einem Bleistift entspricht. Ist ein Übergang auf einen anderen Kanal möglich, kann die visuelle Aufmerksamkeit auch z.B. durch akustische Signale angefordert werden, was aber zu einer sehr unnatürlichen Dialogsituation führen würde.

Bei dem Einsatz von Zeigegesten in POPEL wird davon ausgegangen, daß der Dialogpartner auf den Bildschirm blickt, d.h. daß er die Zeigegeste sieht. Zur Unterstützung der Wahrnehmung werden Mittel angeboten, die beiden genannten Fixierungsmethoden zu implementieren.

7.2 Die Generierung von Referenzen in POPEL

7.2.1 Die Stellung im Modell

Die Generierung von Referenzausdrücken gehört zu den Aufgaben, zu deren Lösung die inhaltsfestlegende und die inhaltsrealisierende Komponente interagieren (siehe Abschnitt 3.4.2.3). Aus der Struktur des konzeptuellen Wissens in der CKB kann nicht abgeleitet werden, welches Element als Referenzausdruck realisiert wird, oder mit welcher syntaktischen Oberflächenstruktur das entsprechende Element realisiert wird. Die Entscheidung

[2] Ganz abgesehen davon ist es anzuzweifeln, ob eine solche Kontrolle wünschenswert ist.

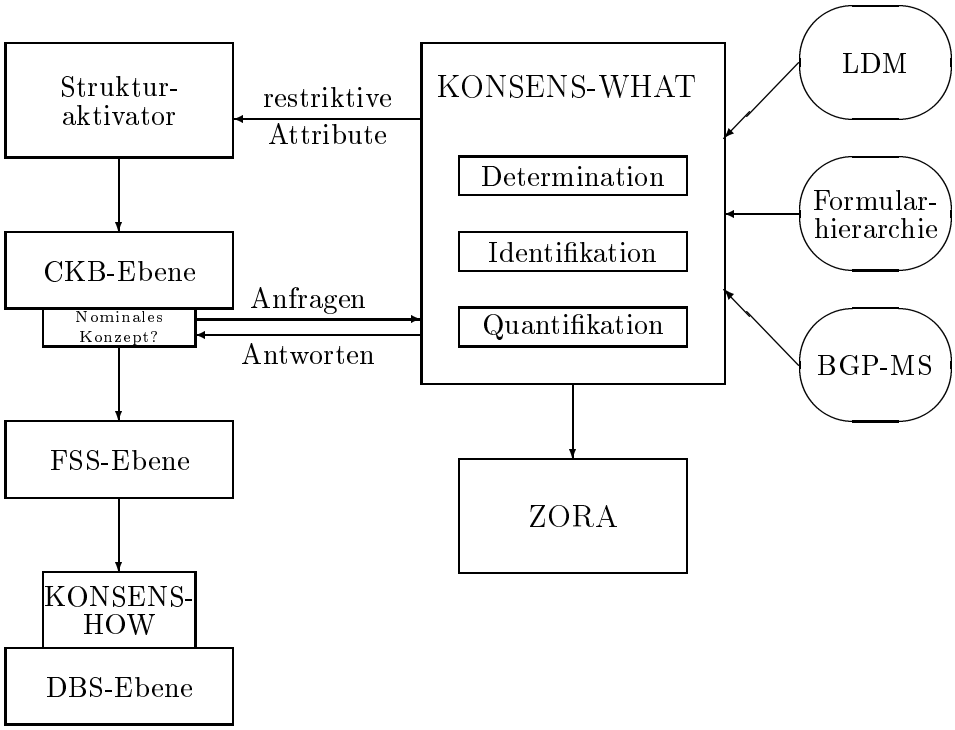


Abbildung 7.1: Die Einbindung der Generierung von Referenzausdrücken in POPEL

zur Pronominalisierung hängt z.B. auch von der Stellung im Satz ab (vgl. Abschnitt 7.1.2), auf die POPEL-WHAT keinerlei Zugriff hat.

Ob ein CKB-Element als Referenzausdruck realisiert wird, wird erst bei der Abbildung in die FSS entschieden. Die Prozesse in POPEL-HOW, die über dieses Wissen verfügen und die die Realisierung der Phrasen durchführen, haben keinen Zugang zu den kontextuellen Wissensquellen im System, so daß auch sie nicht alleine über die Struktur und die Merkmale entscheiden können.

Diese Restriktionen resultieren in einer Zweiteilung der Verarbeitung (siehe Abbildung 7.1): in der Komponente zur Behandlung von Referenzen werden die aus der CKB und den kontextuellen Wissensquellen bestimmbaren Parameter festgelegt und bei der syntaktischen Realisierung die syntaktisch-semantischen [Kresse, 1991][3]. Stellt ein CKB-Objekt bei der Regelanwendung fest, daß ein Objekt für ein FSS-Konzept erzeugt werden soll, das Subkonzept des Konzepts `THING` ist, wird *synchron* eine Anfrage an KONSENS-WHAT gestellt. Als Daten werden das CKB-Objekt und das FSS-Konzept übergeben.

Die Entscheidungen, die KONSENS-WHAT trifft, lassen sich in drei Hauptaufgaben unterteilen:

- die Art der Referenz, z.B. ob definit oder indefinit (Determination);

[3]Die beiden Teile wurden in Anlehnung an die Trennung in einen inhaltsfestlegenden und inhaltsrealisierenden Teil KONSENS-WHAT und KONSENS-HOW genannt.

- die Auswahl zusätzlicher Attribute zur Abgrenzung gegen andere Objekte (Identifikation);
- die Auswahl von Quantoren (Quantifikation).

Der Rückgabewert ist die Art der Referenz, sowie Quantifizierungsinformation. Sollen zusätzliche restriktive Attribute verbalisiert werden, werden diese Elemente dem Strukturaktivator zugeleitet, der sie in die Realisierungskaskade übergibt. Besteht die Möglichkeit, eine begleitende Zeigegeste zu generieren, wird außerdem die Zeigegestengenerierungskomponente aufgerufen.

Die Daten der Antwort, die das CKB-Objekt von KONSENS-WHAT erhält, werden in dem nominalen FSS-Objekt gespeichert und an das daraus entstehende DBS-Objekt übergeben. Die zweite Stufe bei der Generierung eines Referenzausdrucks erfolgt in der Initialisierungsphase des DBS-Objekts, wenn die syntaktischen Merkmale, wie Wortklasse, Typ des Artikels oder Numerus, benötigt werden, um den initialen DAG des Objekts aufzubauen.

Im Zusammenhang mit der Verarbeitung von Referenzausdrücken sind noch viele Fragen offen. Sowohl die Darstellung der Diskursstruktur, als auch die Entscheidung, wann welcher Ausdruck verbalisiert werden soll, können nicht exakt und "richtig" definiert werden. Die Implementierung mit dem System KONSENS wurde deshalb so durchgeführt, daß das Entscheidungswissen deklarativ vorliegt und nicht durch die Verarbeitungsalgorithmen festgelegt wird.

7.2.2 Die Auswahl des Referenzausdrucks

Die Aufgaben, die KONSENS-WHAT übernimmt, sind die Determination, Identifikation, und Quantifikation für das Objekt, so daß die beabsichtigte Beziehung zwischen dem Referenzausdruck und dem Referenzobjekt möglichst eindeutig und schnell erkannt wird. Die Bestimmung der Quantoren zu einem Referenzausdruck wurde im Rahmen dieser Arbeit nicht näher untersucht.[4] Aus der Struktur der CKB wird lediglich bestimmt, ob das CKB-Element, für das eine Anfrage gestellt wird, aus einem Element oder mehreren Elementen besteht, und diese Information an das anfragende Objekt zurückgegeben.

Die zentrale Aufgabe von KONSENS-WHAT ist es, diejenigen Informationen auszuwählen, die dazu führen, daß dem Dialogpartner durch die Form und den Inhalt eines Referenzausdrucks signalisiert wird, ob ein neues Objekt eingeführt wird, ob es bekannt ist, und wo im Kontext er nach dem Referenzobjekt suchen muß. In Abschnitt 7.1.2 wurden drei verschiedenen Arten von Referenz unterschieden. Die wichtigste Unterscheidung ist die in indefinit und definit. Definite Referenzen können abermals in verschiedene Kategorien unterteilt werden, die davon abhängen, wo im situativen Kontext der Antezedens zu suchen ist.

Erreicht eine Anfrage KONSENS-WHAT, muß zuerst die *Deskriptionskategorie* für das CKB-Objekt bestimmt werden. Es muß dazu festgestellt werden, ob der Dialogpartner das Konzept kennt, und wenn ja, ob es aus seinem Wissen ableitbar ist, oder ob es bereits

[4] Siehe [Allgayer and Reddig, 1990] für die Verarbeitung von Quantoren in XTRA.

explizit im Dialog erwähnt wurde und wo. In Anlehnung an [Reichman, 1985] werden derzeit fünf Kategorien unterschieden [Kresse, 1991]:

- indefinit: das Objekt ist unbekannt.
- vollständig definit: das Objekt ist zwar bekannt, ist jedoch entweder in einem Zusammenhang erwähnt worden, der nicht zu dem aktuellen Thema gehört, oder es ist nicht explizit erwähnt worden, kann aber aus dem impliziten Benutzerwissen als bekannt abgeleitet werden.
- eng definit: das Objekt wurde im engeren Dialogkontext erwähnt, eine Verbalisierung z.B. als Pronomen könnte aber zu Verwechslungen führen.
- pronominell: das Objekt hat einen hohen Fokuswert und kann eindeutig identifiziert werden, wenn es z.B. als Pronomen verbalisiert wird.
- demonstrativ: im unmittelbaren sprachlichen Dialogkontext befindet sich ein Antezedens für das Objekt.

Die Benennung der Kategorien deutet an, wie die syntaktische Form für das CKB-Objekt aussehen könnte, legt diese aber nicht fest. Sie ordnet lediglich die kontextuelle Information, die zu diesem Objekt vorhanden ist, in eine Kategorie ein. Ein Objekt, das als 'pronominell' festgelegt wurde, kann z.B. auch als Nominalphrase realisiert werden.

Mit dieser Kategorisierung wird die *sprachliche Information* festgelegt, die in POPEL-HOW weiterverarbeitet wird. Die Entscheidung, ob eine Zeigegeste generiert werden kann, und welche Auswirkungen dies auf die Kategorie der begleitenden sprachlichen Äußerung hat, ist in der Wissensquelle von KONSENS-WHAT kodiert (siehe unten und Abschnitt 7.2.4.2).

Wurde eine Kategorie ausgewählt, wird diese als Antwort auf die Anfrage zurückgegeben und bei der syntaktischen Realisierung weiter verarbeitet.

Die Auswahl benutzt als Wissensquellen die Dialogstrukturierung mit dem LDM, das Benutzermodell BGP-MS, die Verbindung zur Formularhierarchie, sowie das System SPREDIAC, mit dem implizites Wissen aus der CKB bestimmt werden kann. Das Entscheidungswissen ist in dem HC2LC-Prädikat `PRE-DETERMINATION` kodiert. Abbildung 7.2 zeigt einige Klauseln, die derzeit in POPEL verwendet werden. Sie basieren auf einer in [Kresse, 1991] beschriebenen Taxonomie, die auf der Grundlage der Wissensbasis von XTRA ausgearbeitet wurde, und Bedingungen für die Wahl der Determinatoren angibt. Die Determinationskategorie wird mit der Anfrage

`(ask ()(PRE-DETERMINATION` < *object* > `?KATEGORIE))`

an die Variable `?KATEGORIE` gebunden.

Besonders das LDM, das den Diskursverlauf mit den Kontexträumen in Abschnitte aufteilt und Fokuswerte für die im Dialog erwähnten Phrasen enthält, bietet die Grundlagen für die Entscheidung über die Determinationskategorie. Prädikate wie `FOCUS>` zum Vergleich des Fokuswertes, `ACTIVE-CONTEXT`, das überprüft, ob das Objekt ein referentielles Objekt im aktuellen Kontextraum besitzt, und `REFO` zur Feststellung, ob es bereits ein

```
(DEFPRED PRE-DETERMINATION
                    ...
         ((PRE-DETERMINATION ?OBJECT PRONOMINAL)
          (REFO ?OBJECT)
          (FOCUS> ?OBJECT 2)
          (IDENTIFICATION= ?OBJECT 1))
         ((PRE-DETERMINATION ?OBJECT SHORT-DEFINITE)
          (REFO ?OBJECT)
          (ACTIVE-CONTEXT ?OBJECT)
          (FOCUS> ?OBJECT 0)
          (IDENTIFICATION< ?OBJECT 3))
         ((PRE-DETERMINATION ?OBJECT SHORT-DEFINITE)
          (REFO ?OBJECT)
          (CONTROLLING-CONTEXT ?OBJECT)
          (FOCUS> ?OBJECT 4)
          (IDENTIFICATION< ?OBJECT 3))
         ((PRE-DETERMINATION ?OBJECT INDEFINITE)
          (NO-REFO ?OBJECT)
          (NO-KNOWLEDGE ?OBJECT))
                    ...
)
```

Abbildung 7.2: Einige Beispielklauseln aus dem Kategorisierungswissen

referentielles Objekt für dieses Element der konzeptuellen Wissensquelle gibt, erlauben den Zugriff auf das LDM. Ebenso sind Prädikate definiert, die den Zugriff auf das Benutzermodell und SPREDIAC ermöglichen. In der ersten Klausel der Abbildung wird die Kategorie `PRONOMINAL` ausgewählt, wenn für das Objekt ein referentielles Objekt im LDM existiert, dessen Fokuswert größer als 2 ist und das eindeutig identifiziert werden kann (s.u.). Existiert für ein Objekt kein Eintrag im LDM und ist es auch nicht als implizites Wissen des Dialogpartners ableitbar, wird eine indefinite Referenz erzeugt.

Die Prädikate `IDENTIFICATION<` und `IDENTIFICATION=` überprüfen, ob ein Objekt eindeutig identifiziert werden kann. Dabei werden drei Werte berechnet, die angeben, ob eine eindeutige Identifikation ohne zusätzliche Attribute (Rückgabewert `1`) oder mit zusätzlichen Attributen (`2`) möglich ist, oder ob eine eindeutige Identifikation nicht möglich ist (`3`). Als Nebeneffekt der Auswertung des Prädikates werden die zusätzlichen Attribute, d.h. Elemente der CKB, bestimmt, die an den Strukturaktivator zur Verbalisierung übergeben werden, wenn die Klausel erfolgreich ausgewertet werden konnte. Der Algorithmus zur Berechnung der Attribute basiert auf [Dale, 1988, p.212ff] und grenzt das Objekt, das bearbeitet wird, in Bezug auf die lexikalischen Merkmale Numerus und Genus, auf gleiche Konzepte im aktuellen Kontextraum des LDM, Elemente mit gleichem Fokus und gleichem Tiefenkasus ab.

Abbildung 7.3 zeigt ein Beispiel für den Ablauf von KONSENS-WHAT. Der hier vorangegangene Satz war

Steuerhandlungen werden von Personen ausgeführt.

und es soll für das CKB-Konzept `TAX-ACTION`, das als *Steuerhandlung* verbalisiert werden kann, die richtige Determinationskategorie für das FSS-Objekt `*STEUERHANDLUNG*--<FSS-GC124-3697>` gefunden werden. Es existiert ein referentielles Objekt, das einen Fokuswert von 5 hat. Bei der Bestimmung der Identifikation wird festgestellt, daß es zu dem Konzept `TAX-ACTION` im Kontext kein Objekt gibt, das gleich oder ähnlich ist. Wegen des hohen Fokuswerts kann also eine pronominale Realisierung vorgeschlagen werden. Der Numerus `plural` wird derzeit gewählt, wenn Elemente der T-Box verbalisiert werden und keine weiteren Informationen über das Element bekannt sind. Numerus `plural` ist z.B. ausgeschlossen, wenn das Element Rollenfüller einer Rolle ist, die als Kardinalitätsrestriktion hat, daß der Füller nur einmal individualisiert werden kann.

7.2.3 Die Erweiterung der Verarbeitung in den DBS-Objekten

Die Daten, die in KONSENS-WHAT bestimmt werden, erhält das anfragende CKB– bzw. FSS-Objekt als Antwort. Sie werden bei der Abbildung in die DBS-Ebene mit übergeben und in der durch KONSENS-HOW erweiterten Initialisierungsphase des Basisprogramms ausgewertet. In dieser Phase wird festgelegt, mit welcher Wortklasse und mit welchen Merkmalen der initiale DAG des Objekts aufgebaut wird. Die Wortklasse kann sich ändern, wenn festgestellt wird, daß für ein Nomen die entsprechende Proform erzeugt werden kann, oder wenn anstelle des Nomens ein Interrogativpronomen erzeugt werden muß. Die Information, daß nach einem Element der CKB gefragt werden soll,

```
==> *STEUERHANDLUNG*-<FSS-GC124-3697>:

...applying contextual knowledge:

refo in previous sentence...
focus of referent: 5 (> 2)...
age of referent: 2 (< 10)...

Identification:
known-objects (ckb-id fss-id) with same or similar...
......genus: (((GC24 . AGENT) (IC37 . person)))
...quantity: (((NO . NO)) ((NO . NO)))
......focus: (((GC49 . TAX-ACTION) (IC38 . ausfuehr))
               ((GC24 . AGENT) (IC37 . person))
               ((GC49 . TAX-ACTION) (IC36 . steuerhandlung)))
........age: (((GC24 . AGENT) (IC37 . person)))
..deep-case: (((GC24 . AGENT) (IC37 . person)))

=> ambiguous-object(s) (ckb-id fss-id):
   NIL

identification possible...

result of contextual knowledge: PRONOMINAL

quantification: plural ...
```

Abbildung 7.3: Ein Beispiel für die Bearbeitung in KONSENS-WHAT

```
(DEFPRED PRONOMINAL
         ((PRONOMINAL ?OBJECT ((PRONOMINALADVERB +)))
          (PRAEPOSITION ?OBJECT)
          (INANIMATE ?OBJECT))
         ((PRONOMINAL ?OBJECT ((REFLEXIVEPRONOUN +)))
          (AGENS ?OBJECT))
                  ...
         ((PRONOMINAL ?OBJECT ((PERSONALPRONOUN +)))
          (SENTENCE-POSITION-OK ?OBJECT))
         ((PRONOMINAL ?OBJECT ((DET-STEM DIES)))
          (OTHERWISE)))
```

Abbildung 7.4: Ein Ausschnitt der Wissensquelle von KONSENS-HOW

ergibt sich aus der Planung in POPEL-WHAT, wird vom Strukturaktivator den entsprechenden CKB-Objekten übergeben und bei den Transformationen durch die Ebenen mit abgebildet.

Das Wissen, welche Merkmale beim Aufbau des initialen DAGs verwendet werden, liegt in Form von HC2LC-Prädikaten vor. Die Determinationskategorie des DBS-Objekts wählt ein Prädikat aus, dessen Klauseln ausgewertet werden und dessen Rückgabewert eine Liste von Merkmal-Wert Paaren ist. Abbildung 7.4 zeigt das Prädikat für die Determinationskategorie `PRONOMINAL`. In ihm wird zunächst überprüft, ob ein Pronominaladverb wie *damit* als Anapher einer Präpositionalphrase gebildet werden kann, was dann möglich ist, wenn das Objekt einen nicht belebten Gegenstand bezeichnet. Die nächste Klausel testet, ob das Pronomen als Reflexivpronomen realisiert werden muß, wenn das gleiche Referenzobjekt auch als Subjekt im Satz aufgetreten ist, oder ob ein Personalpronomen möglich ist. Dabei werden mit dem Prädikat `SENTENCE-POSITION-OK` die Objekte, mit denen das Objekt verbunden ist, angefragt, ob sich in der DBS-Ebene bereits ein anderes nominales Objekt befindet, das nicht pronominalisiert ist, und das nach dem Verb in dieser Ebene eingetroffen ist. Die letzte Klausel, die immer erfüllbar ist, führt anstelle einer nicht möglichen Pronominalisierung zur Verwendung des demonstrativen Artikels *dies*. Damit wird die Intention der Determinationskategorie, nämlich einen Bezug auf ein im engen Kontext stehendes Referenzobjekt herzustellen, mit anderen Mitteln erfüllt.

Abbildung 7.5 zeigt die Weiterverarbeitung der Daten, die in Abbildung 7.3 bestimmt wurden. Das aus der Abbildung des FSS-Objekts `*STEUERHANDLUNG*-<FSS-GC124-3697>` entstehende DBS-Objekt ist Nominativergänzung (`E0`) bei einem Verb. Die Auswertung des Prädikats `PRONOMINAL` ergibt, daß ein Pronomen erzeugt werden kann. Zusammen mit der Information über den Numerus wird eine Liste mit Merkmal-Wert Paaren erstellt. Beim Aufbau des initialen DAGs führen diese Daten dazu, daß das DBS-Objekt syntaktisch als eine Pronominalphrase und nicht als eine Nominalphrase realisiert wird. Die weitere Verarbeitung führt, zusammen mit dem Verb, zur Ausgabe von

Sie verursachen ...

```
==> *STEUERHANDLUNG*-<FSS-GC124-3697> (E0):

...applying semantical & syntactical knowledge:

no praeposition...
no profession...
not emphasized...
not embedded...
no agens...
sentence-position o.k...

result: ((PERSONALPRONOUN +) (NUMBER PL))

complete determination: ((ANAPHER +) (ELISION +) (NUMBER PL))
```

Abbildung 7.5: Die Auswahl der initialen Merkmale in KONSENS-HOW

7.2.4 Die Generierung von Zeigegesten

7.2.4.1 Die Entscheidung zur Generierung einer Zeigegeste

Über ein Kopplungssystem kann zu jedem Konzept der CKB festgestellt werden, ob es einen Ausschnitt der Graphik gibt, der diesem Konzept zugeordnet ist und auf den gezeigt werden kann. Das Prädikat `POINTABLE` kann in den Klauseln, die die Determinationskategorie bestimmen, benutzt werden, um zu testen, ob auf ein Objekt der Graphik gezeigt werden kann. Somit kann in KONSENS-WHAT festgestellt werden, ob ein Referenzausdruck mit einer begleitenden Zeigegeste generiert werden soll oder nicht.

Für ein Konzept, das im unmittelbaren Dialogkontext erwähnt wurde, das daher einen hohen Fokuswert im LDM hat und mit einem Pronomen sehr gut verständlich verbalisiert werden kann, ist die Generierung einer Zeigegeste wenig sinnvoll. Ebenso wie im natürlichen Gebrauch sind Zeigegesten vor allem dann angebracht, wenn mit ihnen neue Objekte eingeführt werden oder wenn komplexe Beschreibungen vermieden werden können.

Probleme treten auf, wenn in einem Satz mehrere Referenzausdrücke mit begleitenden Zeigegesten generiert werden sollen. Die in Abschnitt 7.1.3.3 beschriebene mögliche visuelle Unaufmerksamkeit, etwa wenn der Dialogpartner gleichzeitig der textuellen und gestischen Ausgabe folgen muß, kann dazu führen, daß Gesten nicht wahrgenommen werden. Die Ausgabe des sprachlichen Anteils des Dialogbeitrags mit einem Sprachsynthesizer könnte dieses Problem lösen. Im Hinblick auf den Einsatz eines solchen Geräts wurde die Beschränkung auf eine Zeigegeste pro Dialogbeitrag, die in [Schmauks and Reithinger, 1988] genannt wurde, nicht beibehalten.

7.2.4.2 Der Zeigegestengenerator ZORA

Die Auswahl der Geste und ihre Visualisierung auf dem Bildschirm ist Aufgabe des Zeigegestengenerators ZORA [Jung *et al.*, 1989]. Das System folgt dem Simulationsansatz und versucht, natürlichem Zeigeverhalten zu folgen, indem ein Ikon auf der Graphik bewegt wird. Das Entwurfsziel war auch hier, eine flexible, modular aufgebaute Komponente zu entwickeln, die nicht domänengebunden arbeitet.

Voraussetzung für die Generierung einer Zeigegeste ist, daß eine Datenstruktur existiert, die zumindest näherungsweise die Struktur der Graphik und die geometrischen Beschreibungen der Teile repräsentiert, auf die gezeigt werden soll. In der Steuerdomäne ist dies die *Formularhierarchie*, die die Positionen und Größen der Formularfelder, die sog. *Regionen*, enthält, sowie deren geometrischen bzw. logischen Zusammenhang. Die Formularfelder, die zu dem Oberbegriff 'Werbungskosten' gehören, bilden beispielsweise keine zusammenhängende Region. Die Formularhierarchie faßt diese Regionen in einer *virtuellen Region* zusammen, auf die ebenfalls referiert werden kann.

Die Wissensquelle von ZORA ist in vier Teile unterteilt und beschreibt den Zusammenhang zwischen der Region, auf die gezeigt werden soll, und einer Geste. Sie setzt geometrische Eigenschaften der Umgebung, geometrische Spezifika der Regionen und mögliche Zeigegesten miteinander in Beziehung. Anhand der Beispiele aus Abbildung 7.6 sollen die Teile der Wissensquelle beschrieben werden:

- Durch diskursbereichsspezifisches Wissen für die Bestimmung der *Grundmuster* werden Parameter wie Proportionen und Gestalt von Objekten definiert (a). In der Steuerdomäne sind dies z.B. Quadrate, deren Seitenverhältnis gleich 1 sein muß, oder Zeilen, die in dieser Domäne mindestens fünfmal so breit wie hoch sein müssen.

- Anhand der absoluten Größen wird der *Objekttyp* bestimmt (b). Eine `AREA` ist z.B. dann groß, wenn ihre Fläche größer als 3600 Quadratpixel ist.

- Anhand des Objekttyps wird eine *Zeigegeste* und ein *Zeigemittel* festgelegt (c). In ZORA werden derzeit Ikone verwendet, die eine stifthaltende Hand (`*pencil*`) oder eine mit einem Finger zeigende Hand (`*finger*`) verwenden. Dies entspricht den Untersuchungen über die Verwendung von Zeigegesten in [Wille, 1989].[5]

- Die *Gesten* setzen sich aus atomaren Aktionen zusammen (d). ZORA kann folgende Aktionen ausführen:

  - eine geradlinige Bewegung des Zeigemittels (`MOVE`);
  - eine Kreisbewegung (`CIRCLE`);
  - eine Bewegung entlang eines Kreisausschnitts (`SEMI-CIRCLE`);
  - eine elliptische Bewegung (`ELLIPSIS`);
  - die Definition des Startpunkts (`START`);

[5] Die Parameter `ULC-...` etc. geben die Eckkoordinaten der Felder an, wobei der erste Buchstabe oben (`U` für 'upper') oder unten `L` (`L` für 'lower'), der zweite für links oder rechts steht. `C` steht für 'corner'.

a) Bestimmung der geometrischen Grundmuster des Objekts

```
((=  (/ LEFT-RIGHT UPPER-LOWER) 1) BOX)
((>= (/ LEFT-RIGHT UPPER-LOWER) 5) LINE)
((OTHER) OTHER)
```

b) Bestimmung des Objekttyps

```
(AREA  (>  (* LEFT-RIGHT UPPER-LOWER) 3600) BIG-AREA)
(OTHER (<= (* LEFT-RIGHT UPPER-LOWER) 3600) FIELD)
(OTHER (>  (* LEFT-RIGHT UPPER-LOWER) 3600) BIG-FIELD)
```

c) Auswahl der Geste und des Zeigemittels

```
((FIELD ULC-X ULC-Y URC-X URC-Y LRC-X LRC-Y)
        (CIRCLE (+ ULC-X (/ (- URC-X ULC-X ) 30)) ULC-Y
                LRC-X (+ URC-Y (/ (- LRC-Y URC-Y ) 30)))
        *pencil* )
((FIELD LLC-X LLC-Y LRC-X LRC-Y)
        (MULTIPLE-LINE-ALONG
           (+ LLC-X 4) (+ LLC-Y 0) (- LRC-X 2) (+ LRC-Y 0))
        *pencil* )
```

d) Definition der Gesten

```
((POINT x1 y1)              ((FAST) (MOVE x1 y1)))
((LINE-ALONG x1 y1 x2 y2)   ((START x1 y1) (SLOW) (MOVE x2 y2)))
((MULTIPLE-LINE-ALONG x1 y1 x2 y2)
                            ((START x1 y1) (SLOW)
                             (MOVE x2 y2) (MOVE x1 y1)
                             (MOVE x2 y2)))
```

Abbildung 7.6: Ausschnitte aus der Wissensquelle von ZORA

- die Festlegung der Geschwindigkeit (`SPEED`, `SLOW` und `FAST`)
- die Markierung der Geste mit einer Spur (`TRACE` und `UNTRACE`).

Aus diesen Primitiva können beliebige, der Domäne und der Graphik angepaßte Bewegungen definiert werden.

Die Generierung der Zeigegeste beginnt mit einer *Vorlaufphase*, in der die Region, die ZORA als Eingabe erhält, diskursbereichsspezifisch geometrisch analysiert wird. Es werden z.B. Eckkoordinaten und Längen von Kanten berechnet. In der Steuerdomäne wird zudem noch die Feldart bestimmt, z.B. ob etwas in das Feld eingetragen werden kann.

Die dort ermittelten Parameter werden in der *Klassifizierungsphase* bei der Bestimmung des Objekttyps verwendet. Dazu wird das Wissen über die Grundmuster und die Objekttypen verwendet. Liegt der Typ der Region fest, auf die gezeigt werden soll, werden alle möglichen anwendbaren Zeigegesten ermittelt, d.h. die Bewegungen und Ikone.

Die *Auswahlphase* wählt aus diesen möglichen Gesten diejenige, von der angenommen werden kann, daß sie der Dialogpartner richtig interpretiert. Die Auswahl erfolgt in einer lokalen Antizipations-Rückkopplungsschleife [Jameson and Wahlster, 1982], in der die Analysekomponente TACTILUS-II dazu verwendet wird, die Benutzerreaktion zu simulieren. Die Geste soll so ausgewählt werden, daß die Analyse der Geste mit TACTILUS-II als Ergebnis diejenige Region ergibt, auf die ZORA zeigen soll. Die eventuelle Korrektur durch den simulierten Verstehensprozeß des Dialogpartners erlaubt die Bestimmung einer Geste, von der angenommen werden kann, daß der damit beabsichtigte Referenzakt erfolgreich ist. Bei der Korrektur werden nicht nur die gesamten Gesten, sondern auch die Zeigemittel einer Geste variiert, so daß auch Gesten, die nicht explizit in der Wissensquelle von ZORA definiert wurden, visualisiert werden können.

Zu der ausgewählte Geste werden in der abschließenden *Visualisierungsphase* die atomaren Aktionen bestimmt. Ein Interpreter wertet diese aus und bewegt das Zeigemittel entsprechend auf der Graphik.

xtra7.ps15cm[Eine von ZORA visualisierte Zeigegeste][grammelzora420]

Abbildung **??** zeigt eine von ZORA generierte Zeigegeste. Die Eingabe bestand darin, daß auf das Wertfeld, das die Zahl '420' enthält, gezeigt werden sollte. Das Feld hat in der Formularhierarchie den Bezeichner `REGION437`. Die Region wird als `FIELD` klassifiziert (siehe Abbildung 7.7). Die erste Geste ist ein Kreis um die Region. Bei der Analyse in der Rückkopplungsschleife wird sie jedoch von TACTILUS-II als Unterstreichen (`UNDERLINE`) klassifiziert.[6] Als Region wird `REGION435` erkannt und nicht die intendierte `REGION437`. Die nächste Geste, ein Unterstreichen des Feldes, erkennt TACTILUS-II so, wie es von ZORA intendiert ist. Das Ergebnis der Analyse ist eine Liste von Regionen, die nach der Güte sortiert ist, in der sie erkannt wurden. Da `REGION437` an erster Stelle steht, wird angenommen, daß der Dialogpartner die Geste richtig interpretiert und sie wird

[6] TACTILUS-II erkennt eine unterstreichende Geste, obwohl die kreisförmige Bewegung das gesamte Feld umschließt. Da ZORA annimmt, daß TACTILUS-II korrekt arbeitet und ein größeres Wissen der vom Benutzer präferierten Gesten hat, wird das Ergebnis akzeptiert.

```
*** I will show you a FIELD!

*** 1. examined gesture:
(*PENCIL* (CIRCLE 880 229 967 1151/5))

*** Gesture-type:
UNDERLINE

*** ==> *Deictic-info*:
((880 227) 88 5 UNDERLINE)

*** Result of deictic analysis by TACTILUS-II:
(REGION435)

*** 2. examined gesture:
(*PENCIL* (MULTIPLE-LINE-ALONG 881 265 965 265))

*** Gesture-type:
UNDERLINE

*** ==> *Deictic-info*:
((923 262) 43 4 UNDERLINE)

*** Result of deictic analysis by TACTILUS-II:
(REGION437 REGION435 REGION366 REGION447 REGION438)

*** REGION RECOGNIZED!!
```

Abbildung 7.7: Die Korrektur der Geste in der Antizipations-Rückkopplungsschleife

visualisiert. Der Bewegungsablauf der Geste kann in der Abbildung **??** nicht dargestellt werden. Es ist lediglich das Ikon abgebildet, das am Endpunkt der Bewegung verharrt.

Kapitel 8

Die Realisierung von POPEL

In diesem Kapitel wird zunächst die Simulationsumgebung für die Parallelverarbeitung vorgestellt. Daran anschließend wird an der Generierung von zwei Sätzen die inkrementelle Verarbeitung und die Verzahnung der Festlegungs– und der Realisierungskaskade gezeigt. Abschließend werden einige Informationen zur gegenwärtigen Implementation von POPEL angegeben.

8.1 Die Simulation paralleler Prozesse

In Abschnitt 6.1 wurde bei der Darstellung des Basismodells von POPEL-HOW die *objektorientierte* Definition der Prozeßobjekte vorgestellt und die Hierarchie der Objekte. Der erste Prototyp von POPEL-HOW auf einer Lispmaschine der Firma Symbolics nutzte zur Implementation der Parallelverarbeitung aus, daß im System Objekte oder *Flavors* definiert sind, die es ermöglichen, Prozesse zu erzeugen, die zu allen im System definierten Prozessen quasiparallel abgearbeitet werden [Symbolics, 1990]. Durch Beimischen des Systemflavors `si:process` in das Flavor `process-object` von POPEL-HOW erbten alle Prozeßobjekte die Variablen und Methoden des Systemflavors.

Das Basisprogramm der Prozeßobjekte wurde dem Objekt als `initial-run-function` übergeben. Diese Funktion wird abgearbeitet, wenn der Scheduler des Systems, der eine Round-Robin Strategie verwendet, einem Prozeß Rechenzeit zuteilt. Die Verarbeitung des Basisprogramms kann durch den Scheduler jederzeit an jeder beliebigen Stelle im Programm unterbrochen werden, was die Definition untrennbarer Einheiten notwendig machte, z.B. den Verbindungsaufbau zwischen zwei Objekten.

Für die Common-Lisp Version von POPEL mußte ein eigenes, einfaches System zur Definition und Verarbeitung paralleler Prozesse implementiert werden, da die Standarddefinition von Common-Lisp [Steele, 1984] keine Systemfunktionen zur Verfügung stellt, mit denen Parallelverarbeitung simuliert werden kann. In Anlehnung an das Flavor `si:process` wurde eine Objektklasse `psim-process`[1] definiert, deren Aufbau und Leistung denjenigen des Flavors entspricht. Auch die notwendigen Methoden und Funktionen wurden nachgebildet, wie z.B. die Funktion `psim-make-process`, die einen Prozeß kreiert und beim

[1] 'psim' ist eine Abkürzung für "**P**rozess**SIM**ulation".

Scheduler anmeldet, oder die Funktion `psim-process-wait`, die einen Prozeß aus der Verarbeitung solange herausnimmt, bis eine Wartebedingung nicht mehr erfüllt ist. Ein Scheduler arbeitet die `initial-run-function` der Instanzen Round-Robin ab und terminiert selbst, wenn kein lauffähiges Objekt mehr in der Schlange ist, oder wenn ein Objekt den Scheduler anhält.

Die Unterbrechung und spätere Fortsetzung laufender Funktionen sind im Common-Lisp Standard nicht definiert. Deshalb basiert die Simulation in POPEL darauf, daß die `initial-run-function` terminiert, z.B. nach einem Durchlauf einer Schleife, und die Kontrolle wieder an den Scheduler abgibt. Zwischenergebnisse, die von der Funktion benötigt werden, müssen im Objekt gespeichert werden bis zum nächsten Aufruf der Funktion.

Wenn die `initial-run-function` wie die Basisprogramme der Prozeßobjekte eine Endlosschleife enthielte, würde nur ein Objekt bearbeitet werden. Deswegen wurde für die Simulation das Basisprogramm in zwei Teile aufgeteilt, eine Initialisierungsfunktion und die Bearbeitungsfunktion, die aus dem Inhalt der Endlosschleife besteht und die die Initialisierungsfunktion nach deren Abarbeitung im Prozeßobjekt ersetzt. Ähnlich wurde die Arbeitsschleife des Planers, des Strukturaktivators und der anderen quasiparallel laufenden Prozesse aufgebrochen. Da keine unvorhergesehenen Unterbrechungen möglich sind, konnte auf die Definition untrennbarer Einheiten verzichtet werden.

8.2 Die qualitative Analyse eines Generierungsbeispiels

8.2.1 Die Generierung einer Sequenz

Das Beispiel besteht aus der Verbalisierung zweier individualisierter Konzepte und deren Rollenfüller (siehe Abbildung 8.1)[2]. Vor dem Generierungslauf wurde das LDM initialisiert, d.h. es ist kein weiterer Dialogkontext vorhanden. Der Aufruf von POPEL erfolgt mit der Funktion `popel` und dem Ziel, das zu verbalisieren ist:

```
POPEL-WHAT>(popel '(swmb (know h (ic40 ic37))))
faehrt
ein mann faehrt
ein mann faehrt mit einem motorrad
ein mann faehrt mit einem motorrad von saarbruecken
ein mann faehrt mit einem motorrad von saarbruecken nach voelklingen

SAY: ein mann faehrt mit einem motorrad von saarbruecken nach voelklingen

dort
dort kauft er
dort kauft er einer frau
```

[2]Da im weiteren Verlauf vom Programm protokollierte Daten verwendet werden, werden die systeminternen Benennungen der Konzepte und Rollen verwendet.

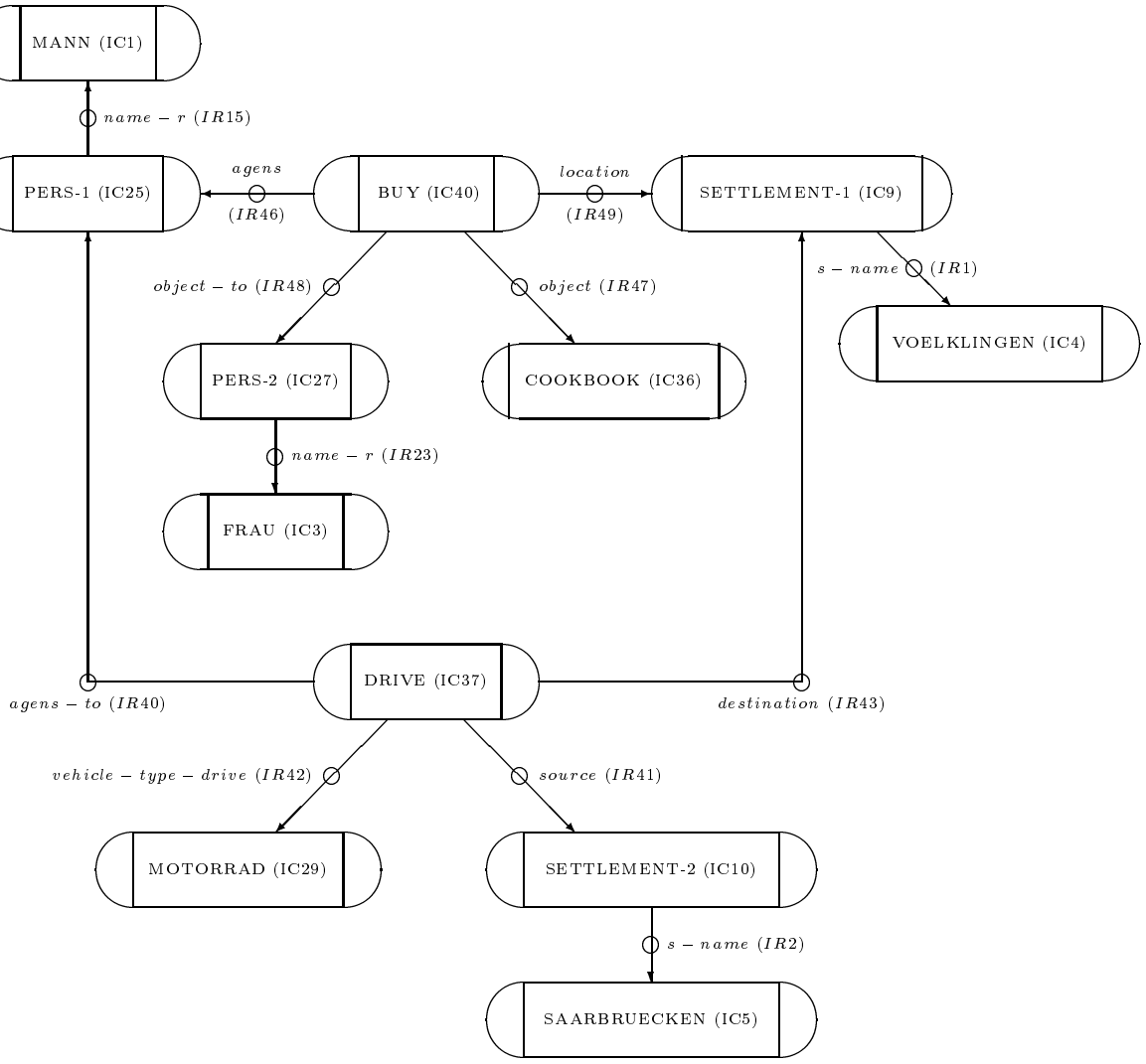


Abbildung 8.1: Die individualisierten Konzepte für das Beispiel

```
dort kauft er einer frau ein kochbuch

SAY: dort kauft er einer frau ein kochbuch
(24890000 622)
```

In diesem Beispiel werden die Zwischenergebnisse inkrementell ausgegeben. Die mit `SAY:` beginnenden Zeilen enthalten die Sätze, die endgültig bestimmt wurden. Das Beispiel wurde auf einer Solbourne S4000 mit Sun Common-Lisp berechnet (siehe Abschnitt 8.3). Die beiden Zahlen, die als Ergebnis von POPEL zurückgeliefert werden, sind die Laufzeit in Mikrosekunden und die Anzahl der Aufrufe der Prozeßfunktionen im Scheduler. Für die Planung und Verbalisierung dieser Sätze werden also 24,89 Sekunden benötigt.

8.2.2 Die Planung des ersten Satzes

Der erste Schritt ist es, die Konzepte des Ziels in die Reihenfolge zu bringen, in der sie verbalisiert werden sollen. Das Prädikat `SORT-ICONCEPTS` im Bedingungsteil des Planoperators

```
(define-text-plan-operator
   :NAME verbalize-a-sequence
   :EFFECT (swmb (know h ?icon))
   :CONSTRAINTS
     (P-AND (IS-LIST ?icon)
            (ICONCEPTS ?icon)
            (SORT-ICONCEPTS ?icon ?sorted-icon))
   :NUCLEUS (SEQUENCE ?sorted-icon)
   :SATELLITES NIL)
```

legt diese fest. Dieses Prädikat untersucht Konzepte und deren Rollen daraufhin, ob sie in eine zeitliche bzw. direktionale Reihenfolge gebracht werden können. Da in dem Prädikat die Repräsentationsprimitiva der CKB für die Begriffe wie 'Ausgangsort' oder 'Zielort' bekannt sein müssen, ist das Prädikat zwar abhängig von der speziellen Repräsentation der Domäne, die Feststellung der Reihenfolge ist aber allgemeingültig.

Die beiden Konzepte werden so linearisiert, daß zuerst dasjenige bearbeitet wird, in dem die Bewegung zum Zielort beschrieben wird, und danach dasjenige, das den Zielort enthält. Im Plan wird damit der Knoten mit dem Ziel

```
(SEQUENCE (IC37 IC40))
```

eingetragen. Die Expansion mit dem `SEQUENCE`-Planoperator

```
(define-text-plan-operator
   :NAME sequence
   :EFFECT (SEQUENCE ?sorted-icon)
   :CONSTRAINTS (ICONCEPTS ?sorted-icon)
   :NUCLEUS ((FORALL ?sorted-icon
                     (SWMB (KNOW H ?sorted-icon))))
   :SATELLITES NIL)
```

erzeugt für jedes der beiden Konzepte einen Planknoten, die nacheinander bearbeitet werden. Für das erste Ziel

```
(SWMB (KNOW H IC37))
```

ist der Operator

```
(define-text-plan-operator
   :NAME say-an-iconcept
   :EFFECT (SWMB (KNOW H ?icon))
   :CONSTRAINTS
      (P-AND (P-IS ?roles-to-verbalize
                   (GET-RELEVANT-ROLES '?icon))
             (P-NOT-EQ NIL ?roles-to-verbalize))
   :NUCLEUS (activate (INFORM ?icon ?roles-to-verbalize))
   :SATELLITES (((FORALL ?roles-to-verbalize
                         (SWMB (KNOW H ?roles-to-verbalize)))
                 (*optional*))))
```

anwendbar, der das zentrale Konzept und die Rollen an den Strukturaktivator übergibt. Die Auswahl und die Festlegung der Reihenfolge der Rollen erfolgt im Prädikat GET-RELEVANT-ROLES. Dieses verwendet die gleiche Funktion wie das Prädikat SORT-ICONCEPTS, um eine direktionale bzw. zeitliche Abfolge festzustellen. Die Ziele, die im Satelliten expandiert werden, sind optional: wenn zu einem dieser Ziele kein Unterplan erzeugt werden kann, wird das Ziel ignoriert.

Die Planung für das erste Ziel der Sequenz führt zu folgenden Unterzielen, die in dieser Reihenfolge berechnet werden:

```
(ACTIVATE (INFORM IC37 (IR40 IR42 IR41 IR43)))
(SWMB (KNOW H IR40))
(ELABORATION IC25)
(ACTIVATE (INFORM IC25))
(SWMB (KNOW H IR42))
(ELABORATION IC29)
(ACTIVATE (INFORM IC29))
(SWMB (KNOW H IR41))
(ELABORATION IC10)
(ACTIVATE (INFORM IC10))
(SWMB (KNOW H IR43))
(ELABORATION IC9)
(ACTIVATE (INFORM IC9))
(SWMB (KNOW H IC40))
```

In den ELABORATION-Operatoren wird untersucht, ob neben den Konzepten selbst noch Rollenfüller verbalisiert werden sollen. Da in dem Beispiel nur Rollen individualisiert sind, die etwas über die Bezeichnung der Konzepte aussagen und dafür keine Operatoren definiert sind, werden nur die Konzepte selbst an den Strukturaktivator gesendet.

8.2.3 Der kaskadierte Verlauf der Generierung

Die Abbildung 8.2 zeigt qualitativ den Verlauf der Generierung des ersten Satzes. Die x-Achse der Graphik zeigt den *Schedulertakt*. In einem Schedulertakt werden alle Prozesse bzw. deren `initial-run-function` einmal aufgerufen.

An der y-Achse sind die Prozesse aufgeführt. Oben stehen die Komponenten von POPEL-WHAT, die derzeit als eigenständige Prozesse realisiert sind (wobei *RKS* den Planer bezeichnet, *SA* den Strukturaktivator und *RH* die Behandlung von Anfragen)[3] und der Terminator. Darunter befinden sich die Prozesse der CKB–, FSS–, und DBS-Ebene mit ihren entsprechenden Bezeichnern, ganz unten das LP-Objekt, das die Linearisierung ausführt.[4]

Die durchgezogenen Linien in der x-Richtung zeigen die Takte an, in denen die Prozesse versuchen, sich abzubilden, oder Anfragen zu bearbeiten. Die gepunkteten Linien zeigen diejenigen, in denen der Prozeß keine Verarbeitung durchführen kann und sich nur mit eventuell neu in der Ebene eintreffenden Objekten bekannt macht. Die Pfeile zwischen den Ebenen stehen für die Abbildung bzw. die Anfragen an die nächsthöhere Ebene. Wenn bei Abbildungen mehrere Objekte gemeinsam ein Nachfolgerobjekt erzeugen, ist an dem Abbildungspfeil bei dem zweiten beteiligten Objekt ein schwarzer Punkt.

Die Unterbrechung der Verarbeitung im Planer in den Takten vier bis sechs hat seine Ursache in der Synchronisation mit dem Strukturaktivator. Damit der Planer nicht dem Strukturaktivator 'davonläuft', d.h. schneller plant, als dieser die Daten weiterverarbeiten kann, überprüft der Planer die Eingabewarteschlange des Strukturaktivators daraufhin, ob sich zu viele Elemente darin befinden. Derzeit ist dieser Wert auf zwei Elemente festgelegt. Da der Planer neben den Elementen der CKB, die zu verbalisieren sind, auch die RST-Relationen an den Strukturaktivator sendet, befinden sich am Anfang der Verarbeitung zwei Elemente in der Eingabewarteschlange.

Vor der Expansion des zweiten Konzepts der Sequenz hält der Planer in Takt 19 an und wartet, bis alle primitiven Ziele, die an den Strukturaktivator gesendet wurden, auch von diesem verarbeitet wurden. `SEQUENCE` ist eine multinukleare RST-Relation, d.h. jedes Konzept trägt eine eigenständige Bedeutung. Bevor die Planung des nächsten Nukleusziels weitergeführt wird, wird deshalb auf die Realisierung des vorangegangenen gewartet.

Anfragen in der CKB-Ebene werden in diesem Beispiel dann gestellt, wenn ein CKB-Konzept eine Namensrolle hat. Die Abbildungsregeln sind so definiert, daß ein solches Konzept, z.B. `SETTLEMENT-2`, nur dann abgebildet werden kann, wenn auch sein Name, z.B. `SAARBRUECKEN`, in der Objektstruktur vorhanden ist. Die Namen werden vom Planer nicht berücksichtigt und müssen deswegen durch Anfragen aktiviert werden.

Daß die Bearbeitung der Anfragen und das Erzeugen der angefragten Objekte im gleichen Zeittakt erfolgen, beruht auf der Simulation der Parallelverarbeitung. Der Prozeß zur Behandlung von Anfragen wird vom Scheduler *nach* den CKB-Objekten bearbeitet, die die Anfragen stellen. Neue Prozesse werden immer nach dem sie erzeugenden Prozeß

[3] Die Abkürzungen sind angelehnt an die englischen Bezeichnungen der Komponenten von POPEL-WHAT (vgl. [Reithinger, 1991])

[4] Die Verbalisierung des Verbs, das als erstes inkrementell ausgegeben wird, ist nicht separat eingezeichnet, da es im gleichen Takt wie die Ausgabe von "Ein Mann fährt" erfolgt.

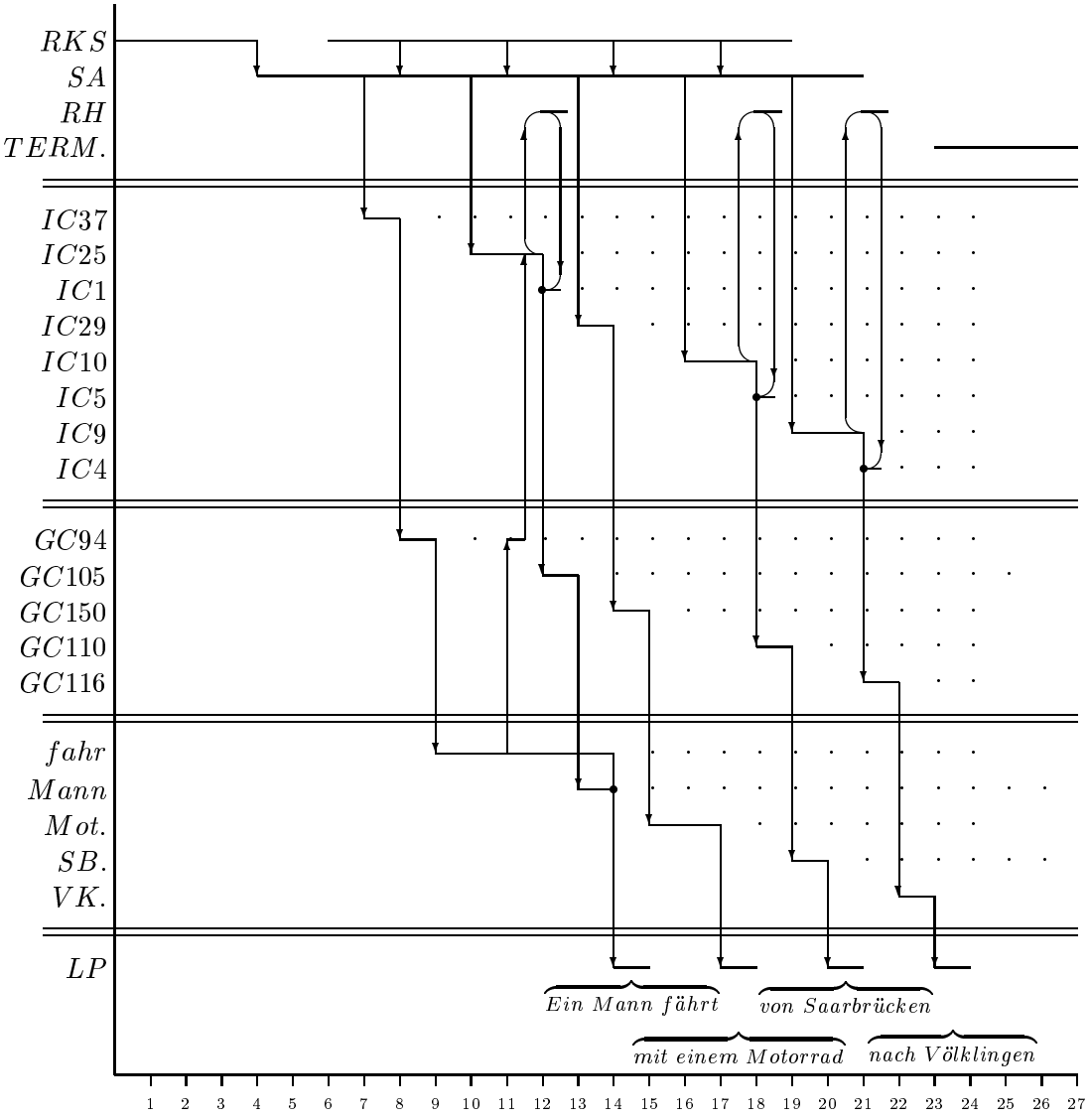


Abbildung 8.2: Der Verlauf der Verbalisierung des Ziels `(swmb (know h (ic40 ic37)))` (Anfang)

in die Prozeßliste eingetragen und noch im gleichen Takt bearbeitet. Zuerst wird also die Anfrage behandelt und das Objekt erzeugt. Dieses kommt direkt nach der Erzeugung im gleichen Schedulertakt zur Verarbeitung und kann sich eventuell abbilden. Während der Verarbeitung des zweiten Satzes wird die Behandlung von Anfragen *vor* den Prozessen der CKB-Ebene durchgeführt. Deswegen erfolgt die Behandlung von Anfragen dort erst einen Takt später (siehe Abbildung 8.3).

Das Verb, das zuerst in der DBS-Ebene ist, stellt eine Anfrage, die über die Ebenen propagiert wird. Die Anfrage ist dieselbe, die auch `IC25 (PERSON-1)` stellt.

Nachdem der Strukturaktivator alle CKB-Objekte aktiviert hat und festgestellt wurde, daß ein Element auf ein Prädikat abgebildet wird, d.h. daß ein Satz entstehen kann, deaktiviert er sich selbst und den Planer (was in diesem Fall bereits geschehen ist) und startet den Terminator (Takt 21). Dieser wartet, bis alle Objekte ihre Anfangskommunikation durchgeführt haben. Dann terminiert er die Prozesse von POPEL-HOW und startet wieder den Strukturaktivator (Takt 27).

Dieser gibt den endgültigen Satz aus und reaktiviert den Planer (vgl. Abbildung 8.3, Takt 28). Die Generierung des zweiten Satz verläuft ähnlich wie die des ersten. Lediglich die Reihenfolge der Rollen und der Rollenfüller ist hier anders. Das Prädikat zu deren Bestimmung muß nun den Dialogkontext berücksichtigen und stellt die Rolle des Ortes an die erste Stelle, da dieser Ort unmittelbar im vorangegangenen Satz erwähnt wurde.

Das Objekt für das Verb stellt in diesem Satz zwei Anfragen (Takt 37 und 42), im Gegensatz zum ersten. Die Ursache liegt darin, daß Anfragen immer dann neu gestellt werden können, wenn sich der Kontext eines Elements geändert hat, was nach dem Eintreffen des Objektes für 'Völklingen' in der DBS-Ebene und dem Aufbau einer Kommunikationsverbindung mit dem Verb-Objekt zutrifft (Takt 41).

8.2.4 Anmerkungen zur Laufzeit

Die Simulation läßt eine exakte Aussage über den Zuwachs an Geschwindigkeit nicht zu, der aus der Parallelverarbeitung stammt. Beispielsweise beeinflußt die Reihenfolge der Prozesse, ob ein Verarbeitungsschritt im gleichen Schedulerschritt ausgeführt wird oder erst im nächsten. Ein Beispiel hierfür ist die Behandlung der Anfragen.

Die gesamte Verarbeitung ist nach 55 Takten beendet, wobei die letzte inkrementelle Ausgabe nach 51 Takten erfolgt. Die Summe der Takte der einzelnen Prozesse, die in der Abbildung mit einer Linie gezeichnet sind, d.h. in denen eine Verarbeitung durchgeführt wird, beträgt 204. Bei einem sequentiellen Vorgehen ohne Anfragen würde der Planer unverändert 31 Takte benötigen. Wenn man die Aufgaben, die der Strukturaktivator übernimmt, dem Planer zuschlägt und die Verarbeitung jedes Objekts in POPEL-HOW ohne das LP-Objekt mit zwei Verarbeitungstakten (einer für die Initialisierung und einer für die Abbildung) durchgeführt werden kann, sind dies nochmals 72 Takte. Es werden also insgesamt mindestens 103 Takte benötigt.

Die Parallelverarbeitung erbringt also gegenüber der günstigsten angenommenen sequentiellen Verarbeitung einen Vorteil von etwa der Hälfte der Laufzeit. Zudem ermöglicht erst sie es, daß Rückwirkungen in dem kaskadierten System unmittelbar verarbeitet werden können.

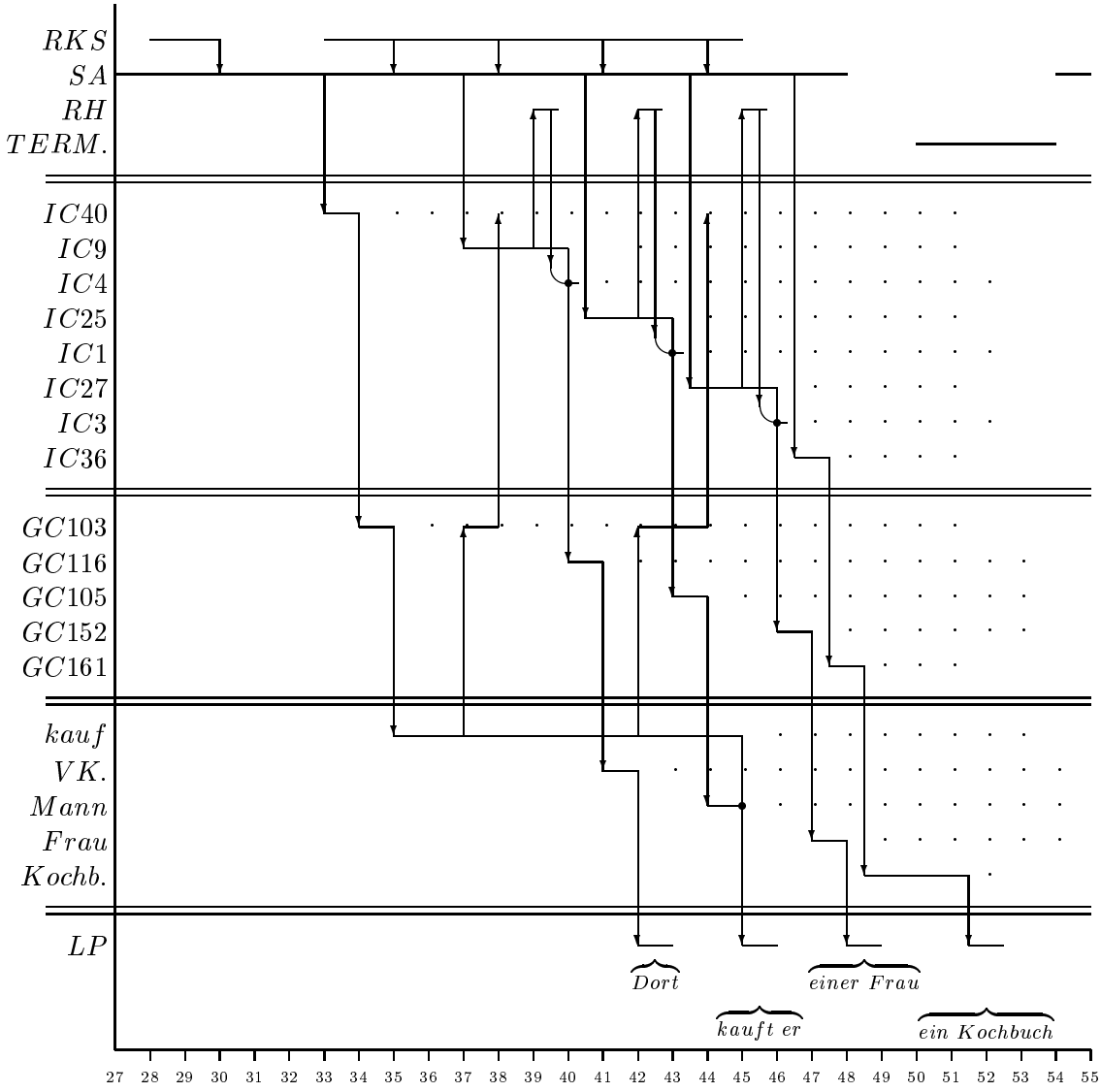


Abbildung 8.3: Der Verlauf der Verbalisierung des Ziels `(swmb (know h (ic40 ic37)))` (Fortsetzung)

8.2.5 Der Einfluß des Dialogkontexts

In einer aufbereiteten Version hat das Dialogfolgegedächtnis von XTRA nach der Generierung des ersten Abschnittes folgende Einträge:

```
Nr. 0: Fokus: 5 CKB: IC37 (GC55)  FSS: IC9  (GC94  . *FAHR*)
Nr. 1: Fokus: 5 CKB: IC25 (GC29)  FSS: IC12 (GC105 . *MANN*)
Nr. 2: Fokus: 5 CKB: IC29 (GC39)  FSS: IC13 (GC150 . *MOTORRAD*)
Nr. 3: Fokus: 3 CKB: IC10 (GC25)  FSS: IC14 (GC110 . *SAARBRUECKEN*)
Nr. 4: Fokus: 3 CKB: IC9  (GC25)  FSS: IC15 (GC116 . *VOELKLINGEN*)
Nr. 5: Fokus: 6 CKB: IC40 (GC60)  FSS: IC16 (GC103 . *KAUF*)
Nr. 6: Fokus: 3 CKB: IC9  (GC25)  FSS: IC19 (GC116 . *VOELKLINGEN*)
Nr. 7: Fokus: 6 CKB: IC25 (GC29)  FSS: IC20 (GC105 . *MANN*)
Nr. 8: Fokus: 6 CKB: IC27 (GC29)  FSS: IC21 (GC152 . *FRAU*)
Nr. 9: Fokus: 6 CKB: IC36 (GC74)  FSS: IC22 (GC161 . *KOCHBUCH*)
```

Wenn man unmittelbar nach der Generierung der zwei Sätze des Beispiels nochmals die Generierung des gleichen Ziels startet, werden die Sätze in einer anderen Reihenfolge und mit anderen Referenzausdrücken verbalisiert:

```
POPEL-WHAT>(popel '(swmb (know h (ic40 ic37))))
das
das kauft er
das kauft er ihr
das kauft er ihr in voelklingen

SAY: das kauft er ihr in voelklingen

dorthin
dorthin faehrt er
dorthin faehrt er mit dem motorrad
dorthin faehrt er mit dem motorrad von saarbruecken

SAY: dorthin faehrt er mit dem motorrad von saarbruecken
(35090000 630)
```

Die Ursache der veränderten Reihenfolge liegt in dem Dialogkontext, der mit den vorangegangenen Sätzen erzeugt wurde. Das Prädikat `SORT-ICONCEPTS` in dem Planoperator `verbalize-a-sequence` berücksichtigt vorrangig den Dialogkontext, bevor es versucht, die Elemente in eine unfokussierte Reihenfolge zu bringen. Da sich der aktuelle Kontextraum des LDMs nicht geändert hat und damit die referentiellen Objekte der vorangegangenen Sätze zugreifbar sind, wird die Reihenfolge gemäß der kontextuellen Bedingungen festgelegt, was sich auch in der Reihenfolge der Elemente des Satzes widerspiegelt. Das Konzept `IC36` besitzt im LDM den höchsten Fokuswert und wurde am Ende des letzten Satzes geäußert, so daß ein unmittelbarer Anschluß erzeugt werden kann.

Die Form der Referenzausdrücke wird ebenfalls durch den Dialogkontext bestimmt. Das Demonstrativpronomen 'das', das den ersten Satz einleitet, wurde ausgewählt, da damit die Verbindung zur unmittelbar vorangehenden Phrase am besten ausgedrückt

werden kann. Die Phrase 'in Völklingen' wurde nicht durch 'dort' pronominalisiert, da im engeren Dialogkontext mit 'Saarbrücken' ein Wort benutzt wurde, auf das sich dieses Prowort auch hätte beziehen können und das den gleichen Fokuswert besitzt.

Die komplexeren Berechnungen bei der Generierung der Referenzausdrücke ergeben eine höhere Laufzeit (35,09 vs. 24,89 Sekunden) als im vorangegangenen Beispiel, bei dem kein Eintrag im LDM vorhanden war.

8.3 Technische Angaben

Das System umfaßt ca. 1,1 Megabyte Lisp-Code in ca. 120 Dateien. Neben dem System selbst werden noch die Wissensrepräsentationssysteme und Wissensquellen des Gesamtsystems benötigt. Die Größe des Lisp-Systems auf einer Solbourne S4000, in dem neben dem Sun Common-Lisp System, das ca. 6,6 MB groß ist, das Common-Lisp Objekt-System *CLOS*[5], SB-ONE und $\mathcal{L}_{\mathcal{SB}-\mathcal{ONE}^+}$ inklusive des HC2LC-Systems, MORPHIX, das LDM-System und die Daten der Wissensquellen geladen sind, beträgt ca. 13 Megabyte, ohne ZORA. Die Zeigegestengenerierung kann nur auf einer Symbolics Lispmaschine verwendet werden und benötigt zusätzlich das System TACTILUS-II. Einige Module des XTRA-Systems, wie z.B. das Benutzermodell BGP-MS oder die in [Allgayer and Reddig, 1990] dargestellte Repräsentation und Verarbeitung von Quantoren wurden parallel zu POPEL entwickelt und sind in der hier präsentierten Version nicht integriert. Die aus diesen Systemen benötigte Funktionalität wurde durch eine Testumgebung simuliert.

Die Entwicklung von POPEL erstreckte sich über mehrere Jahre. Der erste Prototyp von POPEL-HOW und das Zeigegestengeneratorprogramm ZORA wurden auf Lispmaschinen des Typs Symbolics 36xx entwickelt. Zur Implementation der Parallelverarbeitung in den Objekten wurden die von der Betriebssystemsoftware bereitgestellten Mittel verwendet. Die Portierung von POPEL-HOW auf eine VAX 8700 und das Common-Lisp System AKCL, der Entwicklungsumgebung von POPEL-WHAT, machte es notwendig, diese Mittel auch dort bereitzustellen (siehe Abschnitt 8.1). Da sich AKCL sehr eng an den Common-Lisp Standard [Steele, 1984] anlehnt, war eine problemlose Portierung des Systems in die derzeit aktuelle Version auf einem Arbeitsplatzrechner möglich.

Die Beispiele, die in Abschnitt 8.2 vorgestellt werden, sind sowohl auf der VAX 8700 als auch auf der Solbourne S4000 berechnet worden und verwenden daher keine Zeigegesten. Die reine Laufzeit für die Generierung eines Satzes auf diesen Rechnern schwankt zwischen 10 und 150 Sekunden auf der VAX und 2 und 20 Sekunden auf dem Arbeitsplatzrechner, je nach Komplexität des Satzes und den Strukturen des Dialogkontexts.

Laufzeituntersuchungen ergaben, daß besonders die Basisalgorithmen der unifikationsbasierten syntaktischen Verarbeitung in der jetzigen Implementation, die von D-PATR [Karttunen, 1986] bzw. SB-PATR übernommen wurden, überproportional viel Rechenzeit benötigen. Von den externen Modulen, die POPEL verwendet, sind besonders SB-ONE und das HC2LC-System rechenzeitaufwendig. Das HC2LC-System verursacht bei komple-

[5]Ursprünglich wurden die Objekte in *Flavors* implementiert, der objektorientierten Erweiterung von Lisp der Firma Symbolics. Zur Anpassung an CLOS wird ein Makro-Paket verwendet, das die Verwendung der Flavors-Syntax auch in CLOS erlaubt.

xeren Anfragen deutlich erhöhte Laufzeiten. Dies wirkt sich besonders bei der Generierung von Referenzausdrücken aus, wenn im LDM mehrere Einträge vorhanden sind (siehe Abschnitt 8.2).

Die Laufzeitverbesserung auf einem Parallelrechner wurde in Abschnitt 8.2 anhand eines Beispiels diskutiert.

Kapitel 9

Zusammenfassung und Ausblick

9.1 Zusammenfassung

In dieser Arbeit wurde ein leistungsfähiges und flexibles Generierungssystem für multimodale Dialogbeiträge entwickelt. Für dieses System wurde eine neuartige Architektur entworfen, die eine inkrementelle und parallele Verarbeitung mit Rückwirkungen zuläßt.

Die Ausgangsfragestellung war, wie ein Generierungssystem aufgebaut sein muß und welche Verarbeitungsstrategie es verfolgen muß, damit es flexibel in verschiedenen Situationen einsetzbar ist. Zudem soll es in der Lage sein, multimodale Dialogbeiträge zu erzeugen. Auf dem Hintergrund einer gegebenen Anwendungssituation in einem natürlichsprachlichen Dialogsystem wurde diese Frage untersucht.

Die Evaluation mehrerer bereits implementierter Systeme und die Analyse der Anforderungen aufgrund der Integration in ein Dialogsystem führten zur Entwicklung einer neuen, performanzorientierten Architektur, in der die sprachliche Verarbeitung in einer doppelten Verarbeitungskaskade abläuft, wobei parallel zur zweiten Kaskadenstufe die Generierung von Zeigegesten erfolgt. Die Architektur, die auf Grund dieser Anforderungen konzipiert wurde, weist Parallelen zu psycholinguistischen Modellen der menschlichen Sprachproduktion auf. Maßgebliche Faktoren für die Performanz eines menschlichen Sprechers sind die inkrementelle und parallele Produktion von Redebeiträgen.

Die äußere Kaskade des Systems besteht aus den beiden Komponenten zur Inhaltsfestlegung und Inhaltsrealisierung. Intern besteht in den Komponenten nochmals ein kaskadierter Verarbeitungsablauf. In der Inhaltsfestlegung arbeiten der Planer und die Komponente zur Feinstrukturierung und Aktivierung des Inhalts kaskadiert. Die Kaskadenstufen in der Realisierungskomponente entsprechen den Wissensrepräsentationsebenen im System. Zu diesem Modell wurden Algorithmen entwickelt, die in einer Implementation erfolgreich eingesetzt wurden.

Die Architektur des Systems POPEL erlaubt es, die Verarbeitung vollständig inkrementell durchzuführen. Inkrementell heißt, daß Teilergebnisse einer Komponente an die nachfolgenden Komponenten schon weitergegeben werden, bevor die Verarbeitung in der Ausgangskomponente vollständig abgeschlossen wurde.

Dadurch können zwischen den Komponenten Rückwirkungen erfolgen, so daß eine Komponente Daten, die für die Bearbeitung notwendig sind, aber aufgrund einer un-

terspezifizierten Eingabe fehlen, bei der Komponente anfordern kann, die diese Informationen liefert. Damit konnte die Trennung der beiden Hauptkomponenten zur Festlegung und Realisierung des Inhalts überwunden werden, ohne daß das aufgabenspezifische Wissen und die Methoden der Einzelkomponenten miteinander vermischt werden mußten. Die Rückwirkungen werden über klar definierte Kommunikationsschnittstellen durchgeführt. In POPEL sind drei Rückwirkungsmöglichkeiten realisiert worden: die Anforderung zusätzlichen konzeptuellen Wissens, die Erschließung kontextueller Daten bei der Generierung von Ellipsen und die Generierung von Referenzausdrücken.

Damit die inkrementelle Verarbeitung mit Rückwirkungen in allen Ebenen gleichzeitig durchgeführt werden kann, arbeiten in POPEL die Komponenten parallel zueinander. In der Kaskade der Inhaltsfestlegung arbeiten der Planer und der Strukturaktivator parallel, innerhalb der Realisierungskaskade von POPEL-HOW wird jedem Element der darunterliegenden Wissensquelle, das verarbeitet werden soll, ein eigener Prozessor zugewiesen. So wird die verteilte, kaskadiert ablaufende Transformation von konzeptuellem Wissen in sprachliche Repräsentationen ermöglicht.

Ergibt die Abbildung eines Elements der konzeptuellen Wissensquelle, daß dieses als referentieller Ausdruck realisiert werden soll, wird eine Anfrage an die Komponente zur Behandlung von Referenzen gestellt. Dort wird, gestützt auf den Inhalt der Kontextwissensquellen, die vorläufige Einordnung in eine Determinationskategorie vorgenommen und das Ergebnis an das anfragende Element zurückgegeben. Vor dem Aufbau der syntaktischen Beschreibung für das Element der syntaktischen Ebene, auf das das konzeptuelle Element abgebildet wird, wird diese Kategorie ausgewertet und die endgültige Form des Referenzausdrucks festgelegt.

Besteht eine Verbindung zwischen dem konzeptuellen Element und einem korrespondierenden graphischen Objekt im gemeinsamen visuellen Kontext der Dialogpartner, der aus einer auf einem Bildschirm dargestellten Graphik besteht, kann auch eine sprachbegleitende Geste erzeugt werden. Diese Erweiterung der sprachlichen Reaktion wurde in POPEL untersucht und erstmals implementiert.

Die Realisierung der multimodalen Ausgabe wird parallel zur Realisierungskaskade von einem flexiblen, wissensbasierten System durchgeführt, das natürliches Zeigen simuliert. Der Ansatz und die verwendeten Mittel, um Gesten zu generieren, können sich dabei auf die Ergebnisse einer empirischen Untersuchung stützen. In einem vierstufigen Prozeß wird diejenige Geste ausgewählt, von der das System annimmt, daß der Benutzer sie korrekt interpretieren kann. Dazu wird die Geste vor der Visualisierung in einer Antizipations-Rückkopplungsschleife in das Zeigegestenanalysesystem eingegeben. Die Geste wird solange modifiziert, bis der intendierte Bereich der Graphik erkannt wird.

Um die geforderte Flexibilität zu ermöglichen, müssen die Wissensquellen des Systems deklarativ dargestellt und kompositional aufgebaut sein, damit sie wechselnden Domänen angepaßt werden können und die sukzessive Konstruktion der Repräsentationsstrukturen für die Äußerungen möglich ist. POPEL wurde innerhalb eines Dialogsystems entwickelt und kann auf die Wissensbasis und Wissensrepräsentationsformalismen des Gesamtsystems zugreifen. Diejenigen Wissensquellen, die speziell nur in POPEL verwendet werden, basieren auf Vorschlägen, die auch in anderen Generierungssystemen erfolgreich eingesetzt werden, wie z.B. die *Rhetorical Structure Theory* bei der Auswahl und Strukturierung des

Inhalts oder die unifikationsbasierte syntaktische Verarbeitung.

Die qualitative Analyse der Generierung von Beispielsätzen zeigt, wie die inkrementelle Verarbeitung und die Verzahnung von Planung und Realisierung verläuft. Die Produktion des Anfangs einer Äußerung beginnt dabei bereits, während die Festlegung des Inhalts noch im Gange ist. Da gleichzeitig mit der inkrementellen Ausgabe auch die Kontextwissensquellen des Systems auf den jeweils neuesten Stand gebracht werden, und diese eine Grundlage der Entscheidungen des Planers sind, beeinflussen diese Änderungen den Planungsprozeß. Die Verarbeitung der gleichen Eingabe mit veränderten Kontextwissensquellen zeigt, daß das System eine Äußerung verbalisieren kann, die der geänderten Situation angepaßt ist.

9.2 Ausblick

Die prototypische Implementierung von POPEL hat gezeigt, daß die Ziele erreicht werden konnten, die mit dem Entwurf der Systemarchitektur angestrebt wurden. An dieser Stelle soll diskutiert werden, welche Probleme weiter untersucht werden sollten, und wie zukünftige Weiterentwicklungen aussehen können.

Interaktion von Anfragen mit der Planung

Bislang interagieren die Behandlung von Anfragen und die Planung nur sehr begrenzt. Wenn eine Anfrage behandelt wird, wird die Planstruktur daraufhin überprüft, ob sie ein Ziel enthält, das die konzeptuellen Elemente dieser Anfrage abdeckt. Ist dies der Fall, wird angenommen, daß im Laufe des weiteren Planungsprozesses die Information, die angefragt wurde, aktiviert wird. Das Element wird dem Strukturaktivator nicht sofort übergeben, was andernfalls getan wird.

Eine wünschenswerte Weiterentwicklung dieses einfachen ersten Ansatzes wäre es, die Anfrage als Ziel zu formulieren, das dann in den Planungsprozeß eingebracht wird. Der Planer kann es dann in die Planstruktur aufnehmen und bei der weiteren Bearbeitung berücksichtigen.

Untersuchung der Bedingungen in den Planoperatoren

Die Auswahl derjenigen Teile der konzeptuellen Wissensquelle, die verbalisiert werden sollen, wird durch die Prädikate in den Bedingungen der Planoperatoren realisiert. In Abschnitt 8.2 wurde z.B. das Prädikat `GET-RELEVANT-ROLES` vorgestellt, das die Rollen zu einem Konzept auswählt und sortiert. Auch wenn das Sortierungskriterium domänenunabhängig deklarativ dargestellt und verarbeitet werden könnte, bleiben die Bedingungen, welche Rollen ausgewählt werden, prozedural formuliert und sind teilweise domänenabhängig.

Die Bedingungen in den beiden anderen vorgestellten RST-Operationalisierungen sind ebenfalls zu einem großen Teil eng an die jeweilige Domäne gebunden. Weitere Untersuchungen sind notwendig, damit eine klarere Trennung in domänenabhängige und domänenunabhängige Bedingungen errreicht werden kann.

Wortwahl

Die Wahl der Inhaltswörter erfolgt derzeit dadurch, daß im Aktionsteil der Regeln, die die Abbildung von der CKB– in die FSS-Ebene festlegen, das Zielkonzept aus dem semantischen Lexikon stammen muß. Das bedeutet, daß ein Konzept der CKB für jede Regel mit einem fest zugeordneten Lexem verbalisiert wird, z.B. das CKB-Konzept `COMMUTE WITH PUBLIC TRANSPO` immer mit *fahren* oder *Fahrt*.

Dieses unflexible Verfahren kann verbessert werden, indem Regeln über die Subsumptionshierarchie vererbt werden und der Klassifikator von SB-ONE ausgenutzt wird. Kurz skizziert, wären folgende Modifikationen notwendig:

Abbildungsregeln für CKB-Konzepte enthalten in ihrem Aktionsteil immer dasjenige Konzept aus dem semantischen Lexikon, das das allgemeinste der möglichen Lexeme ist, z.B. *bewegen*. Wird eine Abbildung des CKB-Konzepts durchgeführt, dann kommen im Laufe der Verarbeitung Rollenfüller zu dem entsprechenden FSS-Konzept hinzu. Mit dem Einsatz des Klassifikators kann nun ein spezielleres FSS-Konzept bestimmt werden, z.B. *fahren*, wenn ein Rollenfüller für die Rolle `instrument` in der FSS-Ebene eintrifft. Wird der Füller dieser Rolle als *Flugzeug* spezialisiert, dann kann der Klassifikator ein noch spezielleres Konzept des semantischen Lexikons bestimmen, nämlich *fliegen*.

Diese Vorgehensweise entspricht einer inkrementellen Verwendung von Diskriminierungsnetzwerken (vgl. [Goldman, 1975]), jedoch muß keine neue, generatorspezifische Wissensquelle aufgebaut werden. Die Strukturierung der FSS und des semantischen Lexikons, zusammen mit dem Klassifikator, bieten alle notwendigen Daten.

Erweiterung multimodaler Ausgabe

In [Schmauks, 1991, p.150f] und [Schmauks and Wille, 1991] wird eine Zusammenfassung möglicher Weiterentwicklungen der multimodalen Interaktion gegeben.

Eine mögliche Weiterentwicklung ist die Generierung von Gesten auf bewegte (Real)-Bilder. Dafür ist der Simulationsansatz besonders geeignet, da die große Datenmenge nur eine grobe bzw. ungenaue Beschreibung der in der Bildsequenz dargestellten Objekte erlaubt. Allerdings tritt hier verstärkt das Problem auf, daß das System nicht direkt überprüfen kann, ob der Benutzer die Zeigegeste verfolgt und versteht. Vielmehr kann aus inadäquaten Benutzerreaktionen nur indirekt erschlossen werden, daß er die Geste nicht richtig interpretiert hat.

Besonders interessant aus der Sicht der Generierung ist die Ausdehnung des visuellen Kontexts auf künstliche, dreidimensionale Szenen, den sog. *Cyberspace*. Dem Benutzer wird dabei durch eine stereoskopische Darstellung eine virtuelle Realität vorgespiegelt, in der er mit einem sog. *Data-Suit* agieren kann. Begleitet ein Dialogsystem als sichtbarer Mitagent den Benutzer in dieser künstlichen Welt, kann dieser, je nach Visualisierung[1] in dieser Welt virtuelle Extremitäten bewegen, um Gesten zu produzieren.

[1] Beispielsweise in der Form des freundlichen kleinen Roboters C3-PO aus dem Film "Krieg der Sterne".

Verwendung einer vollständig bidirektionalen grammatischen Komponente

In Abschnitt 3.3 wurde dargestellt, in wieweit in XTRA und damit in POPEL die Verwendung von bidirektionalen Methoden und Wissensquellen möglich ist. Besonders im Bereich der grammatischen Verarbeitung wird seit einiger Zeit untersucht, wie diese vollständig bidirektional durchgeführt werden kann (vgl. [Neumann, 1991a]).

POPEL stellt bestimmte Bedingungen sowohl an die Grammatik, z.B. diejenige der Lexikonzentrierung, als auch an die Verarbeitungsverfahren, z.B. die Unabhängigkeit von der Eingabereihenfolge, damit die inkrementelle und parallele Verarbeitung möglich ist (siehe Abschnitt 6.2). An die interne Repräsentation und deren Verarbeitung hingegen stellt das Modell keine Anforderungen.

Es ist deshalb möglich, die derzeit in der DBS– bzw. ILS-Ebene implementierten Komponenten gegen die einer bidirektionalen grammatischen Komponente auszutauschen, wie sie z.B. in [Neumann, 1991b] vorgeschlagen wird. Das dort vorgestellte Modell der grammatischen Verarbeitung basiert auf dem Ansatz der *head driven phrase structure grammar* (HPSG) [Pollard and Sag, 1987]. Die HPSG vereinigt semantische, syntaktische und phonologische Merkmale in einer einheitlichen Struktur. Der Ansatz, daß das Kopfelement einer Phrase deren Struktur einschließlich der abhängigen Elemente bestimmt, ist vergleichbar mit der dependentiellen Strukturierung in POPEL.

Implementation auf einem Parallelprozessor

Mit der Realisierung der parallelen Verarbeitung gehört POPEL in das Teilgebiet der *Distributed Artificial Intelligence* [Bond and Gasser, 1988]. Im Bereich der Generierungssysteme ist das einzige, zumindest teilweise vergleichbare Modell der *Incremental Parallel Formulator* [DeSmedt, 1990], der ausschließlich die syntaktische Verarbeitung in einer ähnlichen Art und Weise realisiert. In diesem System werden für alle Elemente der Eingabe, d.h. für jede syntaktische Phrase und jedes Wort Prozesse erzeugt, die parallel zueinander arbeiten. Massiv parallele Ansätze wie die *Spreading-Activation* Verfahren in [Kitano, 1990] oder [Ward, 1990] arbeiten direkt auf den Wissensquellen. Ward [Ward, 1990, p.15] nennt diese Verfahren 'in-part' Parallelität im Gegensatz zu der 'part-wise' Parallelität bei DeSmedt oder in POPEL. Insofern sind diese Ansätze nur bedingt mit POPEL zu vergleichen.

Diese Systeme sind, soweit sie implementiert wurden, auf Einprozessoranlagen realisiert.[2] In der Literatur konnte kein System gefunden werden, das auf einem Parallelrechner lauffähig implementiert wurde.

Die maximale Anzahl parallel laufender Prozesse in den Beispielsätzen aus Abschnitt 8.2 beträgt 23. Derzeit sind Parallelrechner mit 32–64 Prozessoren und darauf abgestimmte Common-Lisp Systeme erhältlich, deren Kapazität für POPEL vollkommen ausreichen würde. Eine Portierung auf ein solches System ist also möglich. Die einzige größere Änderung, die notwendig wäre, ist die Definition von Bedingungen, die verhindern, daß parallel-

[2] Von dem Dialogsystem, das in [Kitano, 1990] beschrieben wird, wurde ein Teil des Parsers auf einem VLSI-Chip realisiert.

laufende Prozesse gleichzeitig auf dieselben Daten zugreifen und diese verändern. Zusammen mit einer Optimierung der implementierten Algorithmen, besonders der Beseitigung der erkannten Schwachpunkte, kann eine Verarbeitungsgeschwindigkeit im Sekundenbereich erwartet werden.

Literaturverzeichnis

[Allgayer and Reddig, 1990] J. Allgayer and C. Reddig. What kl-one lookalikes need to cope with natural language. In K. Bläsius, U. Hedstück, and C.R. Rollinger, editors, *Sorts and Types in Artificial Intelligence*, pages 240–285. Springer, Berlin, 1990.

[Allgayer and Schmitt, 1991] J. Allgayer and R. Schmitt. $\updownarrow_{\mathbf{SB-ONE^+}}$: Die zugangssprache zu **sb-one**$^+$. Memo, Lehrstuhl für Informatik IV, Universität des Saarlandes, Saarbrücken, 1991. In Vorbereitung.

[Allgayer *et al.*, 1989a] J. Allgayer, K. Harbusch, A. Kobsa, C. Reddig, N. Reithinger, and D. Schmauks. Xtra: A natural-language access system to expert systems. *International Journal of Man-Machine Studies*, 31:161–195, 1989.

[Allgayer *et al.*, 1989b] J. Allgayer, R.M. Jansen-Winkeln, A. Kobsa, C. Reddig, N. Reithinger, and D. Schmauks. Xtra – ein natürlichsprachliches zugangssystem zu expertensystemen. Memo Nr. 39, Lehrstuhl für Informatik IV, Universität des Saarlandes, Saarbrücken, 1989.

[Allgayer *et al.*, 1989c] J. Allgayer, R.M. Jansen-Winkeln, C. Reddig, and N. Reithinger. Bidirectional use of knowledge in the multi-modal nl access system xtra. In *11. IJCAI*, pages 1492–1497, Detroit, 1989.

[Allgayer, 1986] J. Allgayer. Eine graphikkomponente zur integration von zeigehandlungen in natürlichsprachliche ki-systeme. In *16. GI-Jahrestagung*, pages 284–298, Berlin, 1986. Springer.

[Allgayer, 1990] J. Allgayer. Sb-one$^+$ – dealing with sets efficiently. In *9. ECAI*, pages 13–18, Stockholm, 1990.

[Appelt, 1985] D.E. Appelt. *Planning English Sentences*. University Press, Cambridge, 1985.

[Appelt, 1987] D.E. Appelt. Bidirectional grammars and the design of natural language generation systems. In *3. TINLAP*, pages 185–191, Las Cruces, NM., 1987.

[Aue *et al.*, 1989] D. Aue, S. Heib, and A. Ndiaye. Sb-one matcher: Systembeschreibung und benutzeranleitung. Memo Nr. 32, Lehrstuhl für Informatik IV, Universität des Saarlandes, Saarbrücken, 1989.

[Bateman *et al.*, 1989] J.A. Bateman, B. Kasper, J.B. Moore, and R. Whitney. The penman user guide. Technical report, Information Sciences Institute, Marina del Rey, CA., 1989. Penman Development Note; Draft.

[Bateman, 1990] J.A. Bateman. Upper modeling: Organizing knowledge for natural language processing. In *Fifth International Workshop on Natural Language Generation*, pages 54–61, Dawson, PA., 1990.

[Beiche, 1989] H.-P. Beiche. Lst–1: Ein expertensystem zur unterstützung des benutzers bei der durchführung des lohnsteuerjahresausgleichs. Master's thesis, Lehrstuhl für Informatik IV, Universität des Saarlandes, Saarbrücken, 1989.

[Bock, 1987] K. Bock. Exploring levels of processing in sentence production. In G. Kempen, editor, *Natural Language Generation*, pages 351–363. Nijhoff, Dordrecht, 1987.

[Bolz and Weber, 1990] M. Bolz and T. Weber. Baby-bone: Integration von babylon-frames in sb-one. Praktikums-Abschlußarbeit, Lehrstuhl für Informatik IV, Universität des Saarlandes, Saarbrücken, 1990.

[Bond and Gasser, 1988] A.H. Bond and L. Gasser, editors. *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann, San Mateo, CA., 1988.

[Bosch, 1988] G. Bosch. Incas, ein interpreter für kaskadierte systeme. Master's thesis, Lehrstuhl für Informatik IV, Universität des Saarlandes, Saarbrücken, 1988.

[Brown and Yule, 1983] G. Brown and G. Yule. *Discourse Analysis*. University Press, Cambridge, 1983.

[Bußmann, 1983] H. Bußmann. *Lexikon der Sprachwissenschaft*. Kröner, Stuttgart, 1983.

[Busemann, 1984] S. Busemann. Surface transformations during the generation of written german sentences. In D.D. McDonald and L. Bolc, editors, *Natural Language Generation Systems*, pages 98–165. Springer, Berlin, 1984.

[Busemann, 1990] S. Busemann. *Generierung natürlicher Sprache mit Generalisierten Phrasenstruktur–Grammatiken*. PhD thesis, Lehrstuhl für Informatik IV, Universität des Saarlandes, Saarbrücken, 1990.

[Butterworth, 1980] B. Butterworth. Some constraints on models of language production. In B. Butterworth, editor, *Language Production. Volume 1: Speech and Talk*, chapter 15, pages 423–459. Academic Press, London, 1980.

[Dale *et al.*, 1990] R. Dale, C. Mellish, and M. Zock, editors. *Current Research in Natural Language Generation*. Academic Press, London, 1990.

[Dale, 1988] R. Dale. *Generating Referring Expressions in a Domain of Objects and Processes*. PhD thesis, Department of Artificial Intelligence, University of Edinburgh, 1988.

[Dale, 1990] R. Dale. Generating recipes: An overview of epicure. In R. Dale, C. Mellish, and M. Zock, editors, *Current Research in Natural Language Generation*, pages 229–255. Academic Press, London, 1990.

[Danks, 1977] J.H. Danks. Producing ideas and sentences. In S. Rosenberg, editor, *Sentence Production*, chapter 10, pages 229–257. Lawrence Erlbaum Associates, Hillsdale, NJ., 1977.

[Danlos, 1987] L. Danlos. A french and english syntactic component for generation. In G. Kempen, editor, *Natural Language Generation*, pages 191–218. Nijhoff, Dordrecht, 1987.

[DeSmedt and Kempen, 1991] K. DeSmedt and G. Kempen. The representation of grammatical knowledge in a model for incremental sentence generation. In C.L. Paris, W.R. Swartout, and W.C. Mann, editors, *Natural Language Generation in Artificial Intelligence and Computational Linguistics*, chapter 13, pages 329–349. Kluwer, Norwell, MA., 1991.

[DeSmedt, 1990] K. DeSmedt. *Incremental Sentence Generation*. PhD thesis, Nijmegen Institute for Cognition Research and Information Technology, Katholieke Universiteit, Nijmegen, 1990.

[Dymetman *et al.*, 1990] M. Dymetman, P. Isabelle, and F. Perrault. A symmetrical approach to parsing and generation. In *13. COLING*, pages 90–96, Helsinki, 1990.

[Ehrich, 1987] V. Ehrich. The generation of tense. In G. Kempen, editor, *Natural Language Generation*, pages 423–440. Nijhoff, Dordrecht, 1987.

[Engel, 1982] U. Engel. *Syntax der deutschen Gegenwartssprache*. Schmidt, Berlin, 1982.

[Engel, 1988] U. Engel. *Deutsche Grammatik*. Groos, Heidelberg, 1988.

[Finkler and Neumann, 1988] W. Finkler and G. Neumann. Morphix – a fast realization of a classification-based approach to morphology. In H. Trost, editor, *4. Österreichische Artificial-Intelligence-Tagung, Wiener Workshop Wissensbasierte Sprachverarbeitung*, pages 11–19, Berlin, 1988. Springer.

[Finkler, 1989] W. Finkler. Popel–how: Ein verteiltes, paralleles modell zur inkrementellen generierung natürlichsprachlicher sätze aus konzeptuellen einheiten, teil 1. Master's thesis, Lehrstuhl für Informatik IV, Universität des Saarlandes, Saarbrücken, 1989.

[Garrett, 1980] M.F. Garrett. Levels of processing in sentence production. In B. Butterworth, editor, *Language Production. Volume 1: Speech and Talk*, chapter 8, pages 177–220. Academic Press, London, 1980.

[Gazdar *et al.*, 1985] G. Gazdar, E. Klein, G. Pullum, and I. Sag. *Generalized Phrase Structure Grammar*. Basil Blackwell, Oxford, 1985.

[Goldman, 1975] N.M. Goldman. Conceptual generation. In R.C. Schank, editor, *Conceptual Information Processing*, pages 289–371. North-Holland, Amsterdam, 1975.

[Görz, 1988] G. Görz. *Strukturanalyse natürlicher Sprache.* Addison-Wesley, Bonn, 1988.

[Grice, 1975] H.P. Grice. Logic and conversation. In P. Cole and J. Morgan, editors, *Syntax and Semantics 3: Speech Acts.* Academic Press, New York, NY., 1975.

[Grosz and Sidner, 1986] B.J. Grosz and C.L. Sidner. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204, 1986.

[Hertzberg, 1988] J. Hertzberg. *Planen.* BI Wissenschaftsverlag, Mannheim, 1988.

[Herzog *et al.*, 1989] G. Herzog, C.-K. Sung, E. André, W. Enkelmann, H.-H. Nagel, T. Rist, W. Wahlster, and G. Zimmermann. Incremental natural language description of dynamic imagery. In W. Brauer and C. Freksa, editors, *Wissensbasierte Systeme*, pages 153–162. Berlin, 1989.

[Hoeppner *et al.*, 1983] W. Hoeppner, T. Christaller, H. Marburger, K. Morik, B. Nebel, M. O'Leary, and W. Wahlster. Beyond domain-independence: Experience with the development of a german language access system to highly diverse background systems. In *8. IJCAI*, pages 588–594, Karlsruhe, 1983.

[Horacek, 1990] H. Horacek. The architecture of a generation component in a complete natural language dialogue system. In R. Dale, C. Mellish, and M. Zock, editors, *Current Research in Natural Language Generation*, pages 193–227. Academic Press, London, 1990.

[Hovy *et al.*, 1989] E.H. Hovy, D.D. McDonald, and S.R. Young. Current issues in natural language generation: An overview of the aaai workshop on text planning and realization. *AI Magazine*, 10(3):27–29, 1989.

[Hovy, 1985] E.H. Hovy. Integrating text planning and production in generation. In *9. IJCAI*, pages 848–851, Los Angeles, CA., 1985.

[Hovy, 1987] E.H. Hovy. *Generating Natural Language Under Pragmatic Constraints.* PhD thesis, Department of Computer Science, Yale University, New Haven, CT., 1987.

[Hovy, 1988] E.H. Hovy. *Generating Natural Language Under Pragmatic Constraints.* Lawrence Erlbaum Associates, Hillsdale, NJ., 1988.

[Hovy, 1990a] E.H. Hovy. Parsimonious and profligate approaches to the question of discourse structure relations. In *Fifth International Workshop on Natural Language Generation*, pages 128–136, Dawson, PA., 1990.

[Hovy, 1990b] E.H. Hovy. Pragmatics and natural language generation. *Artificial Intelligence*, 43:153–197, 1990.

[Hovy, 1990c] E.H. Hovy. Unresolved issues in paragraph planning. In R. Dale, C. Mellish, and M. Zock, editors, *Current Research in Natural Language Generation*, pages 17–45. Academic Press, London, 1990.

[Hovy, 1991] E.H. Hovy. Approaches to the planning of coherent text. In C.L. Paris, W.R. Swartout, and W.C. Mann, editors, *Natural Language Generation in Artificial Intelligence and Computational Linguistics*, chapter 3, pages 83–102. Kluwer, Norwell, MA., 1991.

[Jablonski *et al.*, 1990] K. Jablonski, A. Rau, and J. Ritzke. *Wissensbasierte Textgenerierung.* Narr, Tübingen, 1990.

[Jackendoff, 1987] R. Jackendoff. *Consciousness and the Computational Mind.* MIT Press, Cambridge, MA., 1987.

[Jacobs, 1987] P.S. Jacobs. King: A knowledge-intensive natural language generator. In G. Kempen, editor, *Natural Language Generation*, pages 219–230. Nijhoff, Dordrecht, 1987.

[Jameson and Wahlster, 1982] A. Jameson and W. Wahlster. User modelling in anaphora generation: Ellipsis and definite description. In *6. ECAI*, pages 222–227, Orsay, Frankreich, 1982.

[Jansen-Winkeln *et al.*, 1991] R.M. Jansen-Winkeln, A. Ndiaye, and N. Reithinger. Fsswastl – interactive knowledge acquisition for a semantic lexicon. In E. Ardizzone, S. Gaglio, and F. Sorbello, editors, *Trends in Artificial Intelligence, 2nd AI*IA Congress*, pages 108–116, Palermo, 1991. Springer.

[Jansen-Winkeln, 1988] R.M. Jansen-Winkeln. Wastl: An approach to knowledge acquisition in the natural language domain. In *European Knowledge Acquisition Workshop*, pages 1–22, Bonn, 1988.

[Jansen-Winkeln, 1990] R.M. Jansen-Winkeln. Hc2lc: Horn clause to lisp compiler. Manuskript, Lehrstuhl für Informatik IV, Universität des Saarlandes, Saarbrücken, 1990.

[Jung *et al.*, 1989] J. Jung, A. Kresse, N. Reithinger, and R. Schäfer. Das system zora – wissensbasierte generierung von zeigegesten. In D. Metzing, editor, *13. GWAI*, pages 190–194, Berlin, 1989. Springer.

[Kalmes, 1990] J. Kalmes. Sb-graph: Eine graphische benutzerschnittstelle für die wissensrepräsentationswerkbank sb-one. Master's thesis, Lehrstuhl für Informatik IV, Universität des Saarlandes, Saarbrücken, 1990.

[Karlin, 1985] R.F. Karlin. Romper mumbles. Technical Report MS-CIS-85-41, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA., 1985.

[Karttunen, 1986] L. Karttunen. D-patr. a development environment for unification-based grammars. In *11. COLING*, pages 74–80, Bonn, 1986.

[Kass and Finin, 1987] R. Kass and T. Finin. The need for user models in generating expert system explanations. Technical Report MS-CIS-87-86, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA., 1987.

[Kay, 1979] M. Kay. Functional grammar. In *Fifth Annual Meeting of the Berkeley Linguistics Society*, 1979.

[Keene, 1989] S.E. Keene. *Object-Oriented Programming in COMMON LISP*. Addison-Wesley, Reading, MA., 1989.

[Kempen and Hoenkamp, 1987] G. Kempen and E. Hoenkamp. An incremental procedural grammar for sentence formulation. *Cognitive Science*, (11):201–258, 1987.

[Kempen, 1987a] G. Kempen. A framework for incremental syntactic tree formation. In *10. IJCAI*, pages 655–660, Mailand, 1987.

[Kempen, 1987b] G. Kempen, editor. *Natural Language Generation*. Nijhoff, Dordrecht, 1987.

[Kitano, 1990] H. Kitano. Parallel incremental sentence production for a model of simultaneous interpretation. In R. Dale, C. Mellish, and M. Zock, editors, *Current Research in Natural Language Generation*, pages 321–351. Academic Press, London, 1990.

[Kobsa and Wahlster, 1988] A. Kobsa and W. Wahlster, editors. *User Models in Dialog Systems*. Springer, Berlin, 1988.

[Kobsa, 1985] A. Kobsa. *Benutzermodellierung in Dialogsystemen*. Springer, Berlin, 1985.

[Kobsa, 1989] A. Kobsa. The sb-one representation workbench. In *Workshop on formal aspects of semantic networks*, Santa Catalina Island, CA., 1989.

[Kobsa, 1990] A. Kobsa. Modeling the user's conceptual knowledge in bgp-ms, a user modeling shell system. *Computational Intelligence*, 6(4), 1990.

[Koskenniemi, 1984] K. Koskenniemi. A general computational model for word-form recognition and production. In *10. COLING*, pages 178–181, Stanford, CA., 1984.

[Kresse, 1991] A. Kresse. Konsens: Kontextsensitive generierung von referenzausdrücken. Master's thesis, Lehrstuhl für Informatik IV, Universität des Saarlandes, Saarbrücken, 1991.

[Krishnamurthy, 1989] E.V. Krishnamurthy. *Parallel Processing: Principles and Practice*. Addison-Wesley, Sydney, 1989.

[Lakoff, 1987] G. Lakoff. *Women, Fire, and Dangerous Things*. University Press, Chicago, IL., 1987.

[Levelt, 1989] W.J.M. Levelt. *Speaking – From Intention to Articulation.* MIT Press, Cambridge, MA., 1989.

[Mann and Thompson, 1987a] W.C. Mann and S.A. Thompson. Rhetorical structure theory: A theory of text organisation. Technical Report ISI/RS-87-190, Information Sciences Institute, Marina del Rey, CA., 1987.

[Mann and Thompson, 1987b] W.C. Mann and S.A. Thompson. Rhetorical structure theory: Description and construction of text structures. In G. Kempen, editor, *Natural Language Generation*, pages 85–95. Nijhoff, Dordrecht, 1987.

[Mann, 1987] W.C. Mann. What is special about natural language generation research? In *3. TINLAP*, pages 206–210, Las Cruces, NM., 1987.

[Marcus, 1987] M. Marcus. Generation systems should choose their words. In *3. TINLAP*, pages 211–214, Las Cruces, NM., 1987.

[McCoy, 1987] K.F. McCoy. Contextual effects on responses to misconceptions. In G. Kempen, editor, *Natural Language Generation*, pages 43–54. Nijhoff, Dordrecht, 1987.

[McDonald, 1981] D.D. McDonald. Natural language generation as a computational problem: an introduction. Technical Report 81-33, Department of Computer and Information Science, University of Massachusetts, Amherst, MA., 1981.

[McDonald, 1987] D.D. McDonald. No better, but no worse, than people. In *3. TINLAP*, pages 200–205, Las Cruces, NM., 1987.

[McDonald, 1991] D.D. McDonald. On the place of words in the generation process. In C.L. Paris, W.R. Swartout, and W.C. Mann, editors, *Natural Language Generation in Artificial Intelligence and Computational Linguistics*, chapter 9, pages 229–248. Kluwer, Norwell, MA., 1991.

[McKeown *et al.*, 1990] K.R. McKeown, J.D. Moore, and S. Nirenburg, editors. *Proceedings of the Fifth International Workshop on Natural Language Generation*, Dawson, PA., 1990.

[McKeown, 1985a] K.R. McKeown. Discourse strategies for generating natural-language text. *Artificial Intelligence*, 27(1):1–41, 1985.

[McKeown, 1985b] K.R. McKeown. *Text generation.* University Press, Cambridge, 1985.

[Meteer, 1990a] M.W. Meteer. Abstract linguistic resources for text planning. In *Fifth International Workshop on Natural Language Generation*, pages 62–69, Dawson, PA., 1990.

[Meteer, 1990b] M.W. Meteer. *The "Generation Gap" – The Problem of Expressibility in Text Planning.* PhD thesis, Department of Computer and Information Science, University of Massachusetts, Amherst, MA., 1990. BBN Report No. 7347.

[Moore and Paris, 1989] J.D. Moore and C.L. Paris. Planning text for advisory dialogues. In *27. ACL*, pages 203–211, Vancouver, 1989.

[Moore, 1980] R.C. Moore. Reasoning about knowledge and action. Technical Report 191, SRI International, Menlo Park, CA., 1980.

[Moore, 1989] J.D. Moore. *A Reactive Approach to Explanation.* PhD thesis, Computer Science Department, University of California, Los Angeles, CA., 1989.

[Müller, 1986] B.S. Müller. Lehrmaterialien babylon. die beispielswissensbasen zur pilzbestimmung. Arbeitspapier 221, GMD, Birlinghoven, 1986.

[Ndiaye, 1990] A. Ndiaye. Fss-wastl: Ein interaktives wissensakquisitionssystem zur erweiterung des semantischen lexikons in xtra. Master's thesis, Lehrstuhl für Informatik IV, Universität des Saarlandes, Saarbrücken, 1990.

[Neumann and Finkler, 1990] G. Neumann and W. Finkler. A head-driven approach to incremental and parallel generation of syntactic structures. In *13. COLING*, pages 288–293, Helsinki, 1990.

[Neumann, 1989] G. Neumann. Popel–how: Parallele, inkrementelle generierung natürlichsprachlicher sätze aus konzeptuellen einheiten, teil 2. Master's thesis, Lehrstuhl für Informatik IV, Universität des Saarlandes, Saarbrücken, 1989.

[Neumann, 1991a] G. Neumann. A bidirectional model for natural language processing. In *5. EACL*, pages 245–250, Berlin, 1991.

[Neumann, 1991b] G. Neumann. Reversibility and modularity in natural language generation. In *ACL Workshop on Reversible Grammar in Natural Language Processing*, Berkeley, CA., 1991.

[Novak, 1987] H.-J. Novak. *Textgenerierung aus visuellen Daten: Beschreibungen von Straßenszenen.* Springer, Berlin, 1987.

[Paris and McKeown, 1987] C.L. Paris and K.R. McKeown. Discourse strategies for describing complex physical objects. In G. Kempen, editor, *Natural Language Generation*, pages 97–115. Nijhoff, Dordrecht, 1987.

[Paris *et al.*, 1991] C.L. Paris, W.R. Swartout, and W.C. Mann, editors. *Natural Language Generation in Artificial Intelligence and Computational Linguistics.* Kluwer, Norwell, MA., 1991.

[Pechmann and Zerbst, 1990] T. Pechmann and D. Zerbst. Konzeptualisierungs- und formulierungsprozesse bei der produktion komplexer nominalphrasen. Arbeiten der Fachrichtung Psychologie 150, Universität des Saarlandes, Saarbrücken, 1990.

[Pollard and Sag, 1987] C. Pollard and I. Sag. *Information-based Syntax and Semantics.* CSLI, Stanford, CA., 1987.

[Profitlich, 1990] H.-J. Profitlich. Sb-one: Ein wissensrepräsentationssystem basierend auf kl-one. Master's thesis, Lehrstuhl für Informatik IV, Universität des Saarlandes, Saarbrücken, 1990.

[Reichman, 1985] R. Reichman. *Getting Computers to Talk Like You and Me.* MIT Press, Cambridge, MA., 1985.

[Reinert, 1990] J. Reinert. Sb-trans: Conflictfree net-to-net transformation based on incrementally classified rules. Manuskript, Lehrstuhl für Informatik IV, Universität des Saarlandes, Saarbrücken, 1990.

[Reithinger, 1984] N. Reithinger. Antwortgenerierung für das erlanger spracherkennungssystem. Master's thesis, Institut für mathematische Maschinen und Datenverarbeitung V, Universität Erlangen-Nürnberg, 1984.

[Reithinger, 1987] N. Reithinger. Generating referring expressions and pointing gestures. In G. Kempen, editor, *Natural Language Generation*, pages 71–81. Nijhoff, Dordrecht, 1987.

[Reithinger, 1989] N. Reithinger. Dialogstrukturen und dialogverarbeitung in xtra. Memo Nr. 38, Lehrstuhl für Informatik IV, Universität des Saarlandes, Saarbrücken, 1989.

[Reithinger, 1991] N. Reithinger. Popel- an incremental and parallel natural language generation system. In C.L. Paris, W.R. Swartout, and W.C. Mann, editors, *Natural Language Generation in Artificial Intelligence and Computational Linguistics*, chapter 7, pages 179–200. Kluwer, Norwell, MA., 1991.

[Rösner, 1986] D. Rösner. *Ein System zur Generierung von deutschen Texten aus semantischen Repräsentationen.* PhD thesis, Institut für Informatik, Universität Stuttgart, 1986.

[Sacerdoti, 1978] E. Sacerdoti. *A Structure for Plans and Behaviour.* North-Holland, New York, NY., 1978.

[Schäfer, 1990] R. Schäfer. Sprediac – intelligente pfadsuche und -bewertung auf vererbungsnetzen zur verarbeitung impliziter referenzen. In H. Marburger, editor, *14. GWAI*, pages 231–235, Berlin, 1990. Springer.

[Schank, 1975] R.C. Schank, editor. *Conceptual Information Processing.* North-Holland, Amsterdam, 1975.

[Scherer, 1990] J. Scherer. Sb-part: Ein partitionsmechanismus für die wissensrepräsentationssprache sb-one. Master's thesis, Lehrstuhl für Informatik IV, Universität des Saarlandes, Saarbrücken, 1990.

[Schmauks and Reithinger, 1988] D. Schmauks and N. Reithinger. Generating multimodal output – conditions, advantages and problems. In *12. COLING*, pages 584–588, Budapest, 1988.

[Schmauks and Wille, 1991] D. Schmauks and M. Wille. Simulation of communicative hand movements in human-computer-interaction. *Computers and the Humanities*, 25(2), 1991. To appear.

[Schmauks, 1991] D. Schmauks. *Deixis in der Mensch-Maschine Interaktion.* Niemeyer, Tübingen, 1991.

[Schmolze and Brachman, 1982] J.G. Schmolze and R.J. Brachman. Summary of the kl-one language. In J.G. Schmolze and R.J. Brachman, editors, *1981 KL-One Workshop*, pages 233–260. Bolt, Beranek and Newman, Boston, MA., 1982.

[Scott and de Souza, 1990] D.R. Scott and C. Sieckenius de Souza. Getting the message across in rst-based text generation. In R. Dale, C. Mellish, and M. Zock, editors, *Current Research in Natural Language Generation*, pages 47–73. Academic Press, London, 1990.

[Searle, 1983] J.R. Searle. *Sprechakte: Ein Sprachphilosophischer Essay.* Suhrkamp, Frankfurt, 1983.

[Shieber *et al.*, 1983] S.M. Shieber, H. Uszkoreit, F.C.N Pereira, J.J. Robinson, and M. Tyson. The formalism and implementation of patr-ii. In *Research on interactive acquisition and use of knowledge*, chapter 4, pages 39–79. SRI International, Menlo Park, CA., 1983.

[Shieber, 1988] S.M. Shieber. A uniform architecture for parsing and generation. In *12. COLING*, pages 614–619, Budapest, 1988.

[Sidner, 1979] C.L. Sidner. Towards a computational theory of definite anaphora comprehension in english discourse. Technical Report 157, Artificial Intelligence Laboratory, MIT, Cambridge, MA., 1979.

[Simmons and Slocum, 1972] R.F. Simmons and J. Slocum. Generating english discourse from semantic networks. *Communications of the ACM*, 15(10):891–905, 1972.

[Steele, 1984] G.L. Steele. *Common LISP: The Language.* Digital Press, Burlington, MA., 1984.

[Swartout, 1983] W.R. Swartout. Xplain: a system for creating and explaining expert consulting programs. *Artificial Intelligence*, 3(21):285–325, 1983.

[Symbolics, 1990] Symbolics. *Lisp Machine Manuals.* Cambridge, MA., 1990.

[Uszkoreit, 1986] H. Uszkoreit. Categorial unification grammar. In *11. COLING*, pages 187–194, Bonn, 1986.

[van Dijk and Kintsch, 1983] T.A. van Dijk and W. Kintsch. *Strategies of Discourse Comprehension.* Academic Press, New York, NY., 1983.

[von Polenz, 1985] P. von Polenz. *Satzsemantik: Grundbegriffe des Zwischen-den-Zeilen-Lesens.* de Gruyter, Berlin, 1985.

[Wahlster, 1982] W. Wahlster. Natürlichsprachliche systeme. In W. Bibel and J.H. Siekmann, editors, *Frühjahrsschule Künstliche Intelligenz*, pages 203–283. Springer, Berlin, 1982.

[Ward, 1989] N. Ward. Capturing intuitions about human language production. In *11. Annual Conference of the Cognitive Science Society*, pages 956–963, Hillsdale, NJ., 1989. Lawrence Erlbaum Associates.

[Ward, 1990] N. Ward. A connectionist treatment of grammar for generation: Relying on emergents. In *Fifth International Workshop on Natural Language Generation*, pages 15–22, Dawson, PA., 1990.

[Wick, 1989] M.R. Wick. The 1988 aaai workshop on explanation. *AI Magazine*, 10(3):22–26, 1989.

[Wille, 1989] M. Wille. Evaluation und aufbau einer analysekomponente für zeigegesten. Master's thesis, Lehrstuhl für Informatik IV, Universität des Saarlandes, Saarbrücken, 1989.

[Winograd, 1983] T. Winograd. *Language as a Cognitive Process. Volume 1: Syntax.* Addison-Wesley, Reading, MA., 1983.

[Wong, 1975] H.K.T. Wong. Generating english sentences from semantic structures. Master's thesis, Department of Computer Science, University of Toronto, 1975.

[Yonezawa and Tokoro, 1987] A. Yonezawa and M. Tokoro, editors. *Object-Oriented Concurrent Programming.* MIT Press, Cambridge, MA., 1987.

[Zock and Sabah, 1988] M. Zock and G. Sabah, editors. *Advances in Natural Language Generation.* Pinter, London, 1988.