Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Document
D-92-24

Kooperierende Agenten

Jürgen Müller, Donald Steiner (Hrsg.)

September 1992

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 20 80
D-6750 Kaiserslautern, FRG
Tel.: (+49 631) 205-3211/13
Fax: (+49 631) 205-3210

Stuhlsatzenhausweg 3
D-6600 Saarbrücken 11, FRG
Tel.: (+49 681) 302-5252
Fax: (+49 681) 302-5341

Deutsches Forschungszentrum für Künstliche Intelligenz

The German Research Center for Artificial Intelligence (Deutsches Forschungszentrum für Künstliche Intelligenz, DFKI) with sites in Kaiserslautern and Saarbrücken is a non-profit organization which was founded in 1988. The shareholder companies are Atlas Elektronik, Daimler Benz, Fraunhofer Gesellschaft, GMD, IBM, Insiders, Mannesmann-Kienzle, Philips, SEMA Group Systems, Siemens and Siemens-Nixdorf. Research projects conducted at the DFKI are funded by the German Ministry for Research and Technology, by the shareholder companies, or by other industrial contracts.

The DFKI conducts application-oriented basic research in the field of artificial intelligence and other related subfields of computer science. The overall goal is to construct *systems with technical knowledge and common sense* which - by using AI methods - implement a problem solution for a selected application area. Currently, there are the following research areas at the DFKI:

- Intelligent Engineering Systems
- Intelligent User Interfaces
- Intelligent Communication Networks
- Intelligent Cooperative Systems.

The DFKI strives at making its research results available to the scientific community. There exist many contacts to domestic and foreign research institutions, both in academy and industry. The DFKI hosts technology transfer workshops for shareholders and other interested groups in order to inform about the current state of research.

From its beginning, the DFKI has provided an attractive working environment for AI researchers from Germany and from all over the world. The goal is to have a staff of about 100 researchers at the end of the building-up phase.

Prof. Dr. Gerhard Barth
Director

Kooperierende Agenten

Jürgen Müller, Donald Steiner (Hrsg.)

DFKI-D-92-24

Diese Arbeit wurde finanziell unterstützt durch das Bundesministerium für Forschung und Technologie (FKZ ITW-9104).

Kooperierende Agenten

Jürgen Müller, Donald Steiner (Hrsg.)
German Research Center for Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3, D-6600 Saarbrücken, Germany
Tel.: ++49 681 302 5322
Fax : ++49 681 302 5341
E-Mail: mueller@dfki.uni-sb.de

Zusammenfassung

Das vorliegende Dokument gibt eine Übersicht über die Aktivitäten des DFKI im Arbeitsbereich 'Kooperierende Systeme'. Es werden die laufenden *Projekte* des Bereichs allgemein vorgestellt. Die *Systeme* (Entwicklungsumgebungen) der einzelnen Projekte werden eingeführt und *Anwendungen* unter Benutzung verschiedener Kooperationstechniken beschrieben. Ferner werden *Techniken* auf theoretischer Basis abgehandelt.

The document in hand displays an overview of the DFKI activities in the field of 'Cooperating Systems'. Current *projects* within the field will be presented. The *systems* and tools of the different projects will be introduced and the *applications* will be described with a special view on the different cooperation techniques. Eventually some specific *techniques* will be discussed on a more theoretical level.

Inhaltsverzeichnis

1 Vorwort

von H.J. Müller

Der folgende Bericht ist eine Gemeinschaftsarbeit, die aus einem projektübergreifenden Workshop zum Thema "Kooperierende Agenten" entstanden ist. Ziel des Berichtes ist es, die am DFKI behandelten und dem Thema zugehörigen

1. Projekte vorzustellen,
2. Grundlagenuntersuchungen darzulegen und
3. Systeme und Anwendungen zu präsentieren.

Dazu werden wir zunächst eine kurze informelle Einführung in die Gedankenwelt der Kooperierenden Agenten geben. Danach werden Projekte, Systeme und Anwendungen verschiedenster Art beschrieben. Mit den dabei entstandenen Ideen werden eine Reihe von Techniken beschrieben, die bei der Realisierung der Applikationen benutzt wurden bzw zur Zeit untersucht werden. Schließlich stellen wir ein Spektrum von Diensten vor, die das DFKI möglichen Kooperationspartnern in diesem Themenkomplex anbieten kann.

1.1 Die Vision

Stellen Sie sich vor, ihre Abteilung erhält eine Ladung frei programmierbarer Roboter (etwa 200 Stück), die an ihre Mitarbeiter zur Unterstützung verteilt werden. Die Roboter können sich frei in den Räumen bewegen, haben Greifarme, zwei Kameras (Stereovision), ein Mikrofon und einen Lautsprecher. Mit anderen Worten, sie sind, was die Mechanik und Elektronik betrifft, optimal ausgestattet. Die Frage, die sich stellt ist:

> Mit welchen Eigenschaften müssen sie intern ausgestattet sein, damit ein vernünfiges (Zusammen-) Arbeiten möglich ist?

Sie sollen Aufträge der Mitarbeiter erledigen, das heißt, daß sie zunächst eine Schnittstelle nach außen brauchen, und sie müssen verstehen, was sie tun sollen. Das letztere ist sicher sowohl von jedem Mitarbeiter als auch von der Voreinstellung abhängig. Nehmen wir also an es gibt eine Menge von Grundfunktionen und einige Spezialfähigkeiten. Der erste Punkt dagegen ist schwieriger. Optimal wäre das Verstehen von natürlicher Sprache, zumindest in einfacher Form:

> Vereinbare einen Termin für morgen vormittag mit Elisabeth.
> Hole mir einen Kaffee aus der Cafeteria.
> Bestelle 5 Päckchen Disketten.
> Hole 5 Stühle und einen kleinen Tisch aus dem Raum nebenan.

Natürlich sollen die Roboter nicht ständig in Fehlerloops enden oder alle 3 Minuten nachfragen, wenn sie ihre Aufgabe nicht direkt lösen können. Möglicherweise ist Elisabeth morgen nicht da, die Cafeteria ist zu, die Bestellzettel sind alle und der Tisch ist zu schwer. Nun dies ist alles kein Problem, wenn sich die Roboter k o o p e r a t i v verhalten. Sie stimmen einen anderen Termin mit Elisabeths Terminroboter ab, organisieren Kaffee und Bestellzettel und holen sich Hilfe für das Tragen. Hier taucht Kooperation in zwei Varianten auf: Kooperation mit dem menschlichen Auftraggeber und mit den Roboterkollegen.

Neben der Kooperation ist auch die Koordination der Roboter untereinander wesentlich (Können sie sich vorstellen, was passiert, wenn die Roboter unkoordiniert über den Flur gehen oder einen Tisch tragen sollen?). Möglicherweise müssen sie in Verhandungen treten, um ihre Pläne (Wie sehen die eigentlich aus?) abzustimmen, zu koordinieren. Oder sie müssen ihre Annahmen über das Wissen und Verhalten eines anderen Agenten revidieren, damit sie auf neue Situationen optimal reagieren können. (Früher ging der Robi von Markus immer an der Wand lang, jetzt läuft er mitten im Flur!?)

Am besten wäre es, wenn man zuerst mal in einer Simulation beobachten könnte wie sich unsere Roboter in ihrer Umwelt zurechtfinden. Dazu muß das Verhalten der Agenten, ihre Kommunikationsfähigkeit und ihr Wissen dargestellt und verarbeitet werden. Das heißt, wir brauchen nicht nur eine Experimentierumgebung, sondern auch eine mächtige Programmiersprache, mit der diese Aufgabe zu meistern ist.

2 Projekte

2.1 KIK

von D. Steiner

Dieser Abschnitt stellt Arbeiten vor, die im Projekt KIK (Künstliche Intelligenz & Kommunikationstechnologie) geleistet worden sind. KIK ist ein Kooperationsprojekt zwischen der Zentralabteilung Forschung und Entwicklung der Siemens AG und dem DFKI.

Ziel des KIK Projektes ist es, eine Technologie zu entwickeln, die durch synergetisches Zusammenwirken von KI- und Kommunikationstechnologien die Bearbeitung komplexer Aufgaben im distribuierten Mensch-Maschine Team unterstützt. Anwendungsbeispiele für diese Technologien sind u.a. (Luft-) Verkehrswesen, Terminvereinbarung, Wissensakquisition im Team, kooperative Schulung und kooperatives Bestellwesen.

Durch die Verknüpfung neuester Methoden aus den Bereichen der Verteilten Künstlichen Intelligenz (VKI) und Computer Supported Cooperative Work (CSCW) mit den derzeitigen und in naher Zukunft zu erwartenden Fortschritten der Kommunikations- und Netzwerktechnologie bestehen nun die Voraussetzungen zur Realisierung von Systemen zur computerunterstützten Teamarbeit.

In KIK wird das Gesamtproblem aus zwei unterschiedlichen Forschungsrichtungen angegangen; diese Aufteilung entspricht der Gliederung des KIK Projektes in die zwei Projekte KIK-TEAMWARE und KIK-TEAMKOM.

Im Projekt KIK-TEAMWARE werden Methoden zur Unterstützung der Kooperation in einem räumlich, zeitlich und kompetenzmäßig distribuierten Team aus Menschen und (teil-) autonomen maschinellen Agenten entwickelt.

Im Projekt KIK-TEAMKOM liegt der Forschungsschwerpunkt in der praktikablen und effektiven Realisierung kooperativen Arbeitens eines geographisch verteilten Teams. Die Telekooperation wird auf einem möglichst hohem Kommunikationsniveau realisiert. Um dies zu erreichen, wird im Rahmen von KIK-TEAMKOM eine multimediale, kooperative Kommunikationsshell zur Unterstützung von Telekooperationssystemen entwickelt.

In beiden Projekten erfolgt über die Erforschung grundlegender theoretischer Aspekte der skizzierten Fragestellungen hinaus anhand von Beispielapplikationen eine praktische Umsetzung der erzielten Ergebnisse in funktionsfähige Prototypen.

2.1.1 KIK-TEAMWARE

Ein Kennzeichen der heutigen Gesellschaft ist ein immer dichter werdendes Netz nationaler und internationaler Beziehungen. Weltweite Kontakte werden immer enger. Wichtige, kurzfristig zu treffende Entscheidungen verlangen eine effektive Zusammenarbeit kooperierender Organisationen und Personen über große Entfernungen hinweg.

Die steigende Komplexität der dabei anstehenden Probleme und die praktisch weltweite Verteilung eines Unternehmens verbieten klassische ('geschlossene') Lösungen und verlangen nach einer integrierten computertechnischen Unterstützung *kooperativen Arbeitens.* Da an dieser Stelle die ausgereiften (KI-) Lösungsmethoden und Systeme an ihre Grenzen gestoßen sind, ist die Entwicklung von Systemen mit völlig neuen Leistungsmerkmalen notwendig. Hierbei wird die Tatsache, daß bei gut funktionierender Kommunikation die Effektivität einer Gruppe größer ist als die Summe der Fähigkeiten der einzelnen Mitglieder, auf die Konzeption solcher Systeme übertragen.

Durch die Verknüpfung neuester Methoden aus dem Bereich der Verteilten Künstlichen Intelligenz (VKI) mit den aktuellen Fortschritten der Kommunikations- und Netzwerktechnologie besteht nun die Möglichkeit, neuartige Mensch-Maschine-Systeme zu konstruieren, die den oben genannten Problemen Rechnung tragen.

Ziel von **KIK-TEAMWARE**, eines gemeinsamen Forschungsprojektes der Siemens AG und des DFKI, ist es, *die Kooperation von Menschen und Maschinen als Mitglieder eines Teams* zu unterstützen. Dabei ist die Gruppe räumlich, zeitlich und kompetenzmäßig verteilt.

Wichtigstes Merkmal dieser Systeme ist, daß Menschen *und* Maschinen gleichermaßen integriert sind und gemeinsam kooperativ zusammenarbeiten. Die Kooperation läuft also weder vom Menschen losgelöst zwischen Maschinen alleine ab, noch findet sie ausschließlich zwischen Menschen statt, sondern bezieht Maschinen als sog. 'intelligente' Assistenten ein.

Um dieses neue Ziel von **'kooperativer Mensch-Maschine Arbeit'** zu erreichen, bedarf es der Entwicklung leistungsfähiger, flexibler Kooperationsmethoden und speziell dafür konstruierter intelligenter Unterstützungsmechanismen.

Grundlage dazu ist die Modellierung der Organisation des Teams allgemein und der speziellen Verbindungen zwischen den einzelnen Teammitgliedern. Es ist unabdingbar zu wissen, welche Rollen die einzelnen Teammitglieder im Rahmen des Kooperationsprozesses einnehmen, welche Beziehungen zwischen ihnen bestehen und ob und mit wem sie wie kooperieren können.

Als Beispiel eines Kooperationsprozesses verdeutliche man sich das Szenario, bei dem mehrere Personen ein Treffen vereinbaren wollen. Maschinelle Systeme können hier als intelligente Assistenten in den Kooperationsprozeß eingebunden werden. Diese erfüllen dabei Sekretariatsfunktionen wie z.B. die Kontaktaufnahme mit den betroffenen Personen, die Festlegung eines geeigneten Termins, die Belegung von Räumen, die Bereitstellung benötigter Materialien (z.B. Ausdrucken von Tischvorlagen) oder die Reservierung von Hotels für auswärtige Teilnehmer.

Die Formalisierung von solchen Kooperationsprozessen resultiert in der Entwicklung einer Kooperationssprache, auf deren Basis sich die einzelnen Teammitglieder -also Menschen und Maschinen- unterhalten und gemeinsam ein Problem lösen können. Die Repräsentation von Kooperationsprozessen, wie sie im Alltag vorkommen, stellt den innovativen Ansatz zur Lösung des Problems dar. Damit verbunden ist die Entwicklung intelligenter maschineller Komponenten, die vermöge der Kooperationssprache in ein Mensch-Maschine-Team eingebunden werden können.

Zur Verifikation des gewählten Ansatzes werden kooperative Anwendungsszenarien aus den

Bereichen Büro und Verkehrswesen der automatischen Auftrags- und Bestellabwicklung, der Luftverkehrskontrolle und des Netzwerkmanagements untersucht und prototypisch implementiert.

IMAGINE - Integrated Multi-Agent Interactive Environment

Zusätzlich vorangetrieben werden die Arbeiten in KIK-TEAMWARE durch die Konsortialführung von Siemens im von der Europäischen Gemeinschaft geförderten ESPRIT II Projekt **IMAGINE**. Die Zielsetzungen von IMAGINE sind

- die Entwicklung einer allgemeinen Architektur zum Entwurf heterogener Multi-Agenten Systeme; hierbei werden insbesondere Modelle zur Beschreibung der Agenten und des Kooperationsprozesses entwickelt,
- die Spezifikation und Implementierung einer formalen Sprache zur Interaktion zwischen Agenten, *MAIL* (Multi-Agent Interaction Language).

Die Modelle werden anhand konkreter Kooperationsszenarien aus den Bereichen Luftverkehrskontrolle und Netzwerkmanagement verifiziert und unter Verwendung von MAIL implementiert. Dabei wird ein zweistufiger Entwicklungsprozeß verfolgt: zunächst werden die Prototypen *in einer verteilten Parlog/Prolog Umgebung implementiert, um daran anschließend aus* Gründen der Performanzsteigerung in C++ umgeschrieben zu werden. Somit sind die im IMAGINE Projekt zu behandelnden Themen und Probleme auch für das TEAMWARE Projekt relevant; die dort erzielten Resultate können direkt in die Arbeiten von KIK-TEAMWARE einbezogen werden.

2.1.2 KIK-TEAMKOM

von D. Scheidhauer

TEAMKOM beschäftigt sich im Bereich verteilter kooperativer Teamarbeit einerseits mit Grundlagenforschung, was im Teilprojekt WiTek (Wissensbasierte Telekooperation) durchgeführt wird, andererseits mit der prototypischen Realisierung konkreter Anwendungen. Diese Aktivitäten werden teilweise mit Partnern aus Siemens-internen Bereichen, teilweise im Rahmen der europäischen Gemeinschaft geförderter Projekte durchgeführt. Außerdem bestehen Kontakte zu anderen DFKI-Projekten (DISCO, AKA-MOD).

Inhaltlich lassen sich diese Projekte in die drei unterschiedliche Anwendungsgebiete einordnen:

- Remote Troubleshooting bei der Flugzeugwartung
- Multi-Mediales verteiltes Lernen in einer vernetzten Umgebung
- Verteiltes Kooperatives Bestellwesen

In der Applikationsdomaine "verteiltes Lernen in einer vernetzten Umgebung" sind drei Projekte angesiedelt:

SETT - Selbstlernen und Tele-Tutoring

MALIBU - Multimedia and Distance Learning in Banking and Business Environments

ECOLE - European Collaborative Open Learning Environment

Remote Troubleshooting für Flugzeugwartung ist das Thema des Projekts ARAMIS (Airline Real Time Application for Maintenance Information Systems). MALIBU, ECOLE und ARAMIS sind Projekte, die von der Europäischen Gemeinschaft im Rahmen von DELTA und RACE gefördert werden.

SETT - Selbstlernen und Tele-Tutoring

Selbstlernen und Tele-Tutoring im SNI-Trainingscenter in München ist ein Pilotprojekt mit dem Ziel, die bereits vorhandenen computerunterstützten Ausbildungsmöglichkeiten (CBT - computer based training) von Servicetechnikern und Wartungspersonal zu verbessern und zu beschleunigen. Um dabei Kosten und Zeit einzusparen, wird in SETT die Realisierung einer geographisch verteilten Schulung angestrebt. Dabei kommen über ISDN vernetzte Multimedia-Workstations zum Einsatz, deren Kommunikationsfähigkeiten in eine einfach zu bedienende Benutzeroberfläche eingebunden sind.

Mit der Teilnahme in diesem praxisorientierten Pilotprojekt soll Erfahrung gesammelt werden, um für die in TEAMKOM zu modellierende generische Kommunikationsshell einen möglichst realitätsnahen Anforderungskatalog zu erzielen.

ECOLE - European Collaborative Open Learning Environment

Das EG-Förderprojekt ECOLE beschäftigt sich mit der Entwicklung fortschrittlicher interaktiver Kommunikationssysteme für Fernlernen. Ziel von ECOLE ist die Entwicklung einer interaktiven, kosteneffizienten, ISDN-basierten Umgebung für pädagogisch verbessertes Fernlernen und flexibles Training. Diese distribuierte Lernumgebung soll als Prototyp realisiert und im Feldversuch ausgetestet werden. Insbesondere wird das Problem der Lehrer-Lerner Interaktion unter Echtzeitbedingungen und unter Zeitverzögerung betrachtet, sowie die Gruppenkommunikation als eine Komponente eines großen multimedialen Fernlerndienstes untersucht.

Die Beteiligung des EPOS Konsortiums, eines Zusammenschlußes von 7 europäischen Telekom-Gesellschaften zeigt, daß die entwickelte prototypische Umgebung in Zukunft als technologische Basis für einen operationalen, europaweiten Fernlerndienst dienen wird.

MALIBU - Multimedia and Distance Learning in Banking and Business Environments

Das von der Europäischen Gemeinschaft geförderte DELTA-Projekt MALIBU zielt auf das Austesten und die Validation von Pilotsystemen, welche Lerndienste bereitstellen. Hierbei wird die Kosteneffektivität und die pädagogische Effizienz von neuen, geographisch verteilten und flexiblen Lerndienstsystemen durch ein großangelegtes Experiment im Bankensektor untersucht. Drei unterschiedlich große Banken aus drei verschiedenen EG-Ländern nehmen direkt an diesem Experiment teil. Die von MALIBU zu entwicklten Lerndienste sollen kooperatives, kognitiv gefördertes Selbstlernen durch die Realisierung folgender Dienste unterstützen :

- Training Management
- Lernmaterialverteilung

- Informationsbeschaffung
- Auswahl geeigneter Werkzeuge
- Lehren(Tutoring) und Feedback Generierung
- Selbstlernen
- kooperatives Lernen in Echtzeit oder mit Verzögerung zwischen Schülern oder zwischen Schüler und Lehrer

Im Rahmen des SETT-Projektes wird 1992 ein Pilotversuch mit über ISDN vernetzten Lern- und Trainerarbeitsplätzen (in Frankfurt, Stuttgart und München) gestartet. Der Aufbau erfolgt innerhalb des SETT-Projektes und dient als Input für die beiden anderen Projekte MALIBU und ECOLE. Eingesetzt wird dafür eine erweiterte Version des bereits in TEAMKOM entwickelten multimedialen Prototyps. Innerhalb dieses Pilotversuches werden Wartungstechniker über Entfernungen geschult.

WiTek (Wissensbasierte Telekooperation)

Telekooperation ist ein Teilgebiet von CSCW. Es ist der Forschungsbereich, der computergestütztes kooperatives Arbeiten über Entfernungen hinweg untersucht. Geographisch und zeitlich voneinander getrennte Personen und intelligente maschinelle Systeme versuchen dabei gemeinsam Probleme zu lösen, vorausgesetzt, daß ihnen eine geeignete Infrastruktur zur Verfügung steht.

Unter dem Begriff Telekooperationssysteme verstehen wir kooperative Applikationen, die geographisch (und zeitlich) verteilt ablaufen können. Telekooperationssysteme sind Systeme, die verteiltes kooperatives Zusammenarbeiten von Menschen und Maschinen ermöglichen bzw. unterstützen. Die steigende Komplexität der Probleme und die weltweite Verteilung von Unternehmen erlauben keine isolierten, zentralen Lösungen, sondern sie erfordern integrierte, computerunterstützte Zusammenarbeit. Hier haben die Programme und Systeme für Single-User-Applikationen ihre Grenzen erreicht. Wenn sie auch weitgehend ausgereift sind und einen hohen Leistungsstandard besitzen, eignen sie sich nicht ausreichend zur Unterstützung kooperativer Tätigkeiten. Deshalb werden neue, andersartige Systeme benötigt, die für die Telekooperation konzipiert wurden.

Unser Ziel im Teilprojekt WiTek ist die Modellierung und Realisierung einer wissensbasierten multimedialen Kommunikations-Shell (KMC-Shell = Knowledge-based Multimedia Communication Shell). Die KMC-Shell wird dem Entwickler von Telekooperationssystemen eine generische, applikationsunabhängige Entwicklungsumgebung zur Verfügung stellen, um Telekooperationssysteme relativ einfach und komfortabel erstellen zu können. Als Grundlage für den Aufbau der KMC-Shell haben wir ein vierschichtiges Modell entworfen, das es uns erlaubt, Telekooperation auf unterschiedlichen Abstraktionsniveaus zu beschreiben.

Dieses Modell illustriert die verschiedenen Abstraktionsebenen des Wissens, das für eine Kooperation notwendig ist. Es umfaßt Wissen über Kommunikation und Informationsaustausch und besteht aus:

Generische Telekooperationsebene:
allgemeine Beschreibung, hohes Abstraktionsniveau, domainunabhängig; d.h. unabhängig von einem bestimmten Anwendungsbereich (Applikationsdomain)

Domain-Ebene:
domainabhängig, Abbildung der generischen Telekooperationsebene auf ein bestimmtes Fachgebiet (Applikationsdomain)

Applikations-Ebene:
applikationsabhängig, Beschreibung eines speziellen Applikationsszenarios

Runtime-Ebene:
applikations- und situationsabhängig, Implementierung eines konkreten Applikationsszenarios

Dieses Modell dient als Grundlage bei der Spezifikation und Definition der Funktionalitäten und Merkmale der KMC-Shell. Die Realisierung der KMC-Shell, basierend auf diesem 4-Ebenen Modell der Telekooperation wird uns eine intelligente Entwicklungsumgebung und einen mächtigen Werkzeugkasten zur Verfügung stellen, zur Unterstützung kooperativer Teamarbeit von Menschen und Maschinen in einer verteilten, heterogenen Umgebung.

Die Ergebnisse und Ideen, die während der Entwicklung der KMC-Shell entstehen, werden in verschiedenen im Rahmen des KIK-Projektes laufender Applikationsszenarien, die zuvor kurz beschrieben wurden, getestet. Darüber hinaus wird die Erfahrung aus diesen Applikationen wiederum für die Modellierung der KMC-Shell im Detail genutzt.

2.2 AKA

Im DFKI-Forschungsbereich Kooperierende System beschäftigen wir uns mit Grundlagen, Anwendungen, und natürlich mit der Implementierung von verschiedenen Aspekten der Verteilten Künstlichen Intelligenz (CSCW, Multi-Agenten-Systeme), d.h. es geht um die Erforschung der Methoden des verteilten Problemlösens, der Kooperation, Kommunikation und Koordination beim Lösen von Aufgaben durch mehrere KI-Systeme.

Speziell im Projekt *Autonome Kooperierende Agenten (AKA)* werden die Grundlagen für Wissensrepräsentationsformalismen, Inferenz-, Kommunikations- und Kontrollmechanismen bei der Kooperation mehrerer autonom agierender Systeme untersucht und die Resultate an verschiedenen Anwendungsbeispielen erprobt .

2.2.1 AKA-Mod: Modellierung Kooperativer Agenten

von H.J. Müller

PROBLEMSTELLUNG

Im Projekt AKA-Mod werden die Prinzipien, die den Übergang von alleinstehenden Systemen zu kooperierenden Systemfamilien realisieren, untersucht. Dabei liegen die Schwerpunkte bei der Agentenmodellierung und den Planungsaktivitäten der Agentengesellschaft einerseits und der Realisierung von Multi-Agenten Szenarien andererseits. Unter einem Agenten verstehen wir prinzipiell ein wissensbasiertes System mit explizit repräsentiertem Wissen. Die Hauptfragestellung ist nun, welche Eigenschaften kooperierende Agenten haben müssen, damit sie in einer Agentengesellschaft *sozial* agieren. Dabei müssen Fähigkeiten künstlicher Agenten wie die Kommunikation, die Kooperation und die Synchronisation untersucht und modelliert werden. Dabei soll die Kommunikation festlegen, wie technisch kommuniziert wird und welcher Art die *Informationspakete* sind. Die Kooperation behandelt sozio-kognitive Aspekte, die zu kooperativem Verhalten führen und soll Fragen der Repräsentation im Agentenmodell (implizit oder explizit) behandeln. Unter der Synchronisation verstehen wir sowohl die technischen Probleme, die beim Arbeiten mit verteilten Systemen üblicherweise vorkommen, als auch die "bewußte"

Synchronisation der Agenten untereinander, wenn sie sich Ressourcen verschiedener Art (also auch ihren Operationsraum) teilen. Zusammenarbeiten heißt zusammen Probleme lösen und damit auch zusammen einen Lösungsweg zu planen. Fragen wie z.B. welche Mechanismen einem Planen im Team zugrunde liegen und der Zusammenhang zwischen der Komplexität der Probleme zu der Planungsmethode sollen beantwortet werden. Dabei sind sowohl die Dekomposition der Probleme wie auch die Synthese der Lösungen beim verteilten Problemlösen zentrale Punkte. Ferner sollen auch Untersuchungen über das "Emergieren" einer Problemlösung aus nicht zentral geplanten Einzelaktionen untersucht werden.

LÖSUNGSANSATZ

Die prinzipielle Vorgehensweise ist die wissensbasierte Modellierung der Agenten. Das heißt, die Agenten werden über hierarchisch strukturierte Wissensbasen modelliert. Dabei ist es notwendig, gemeinsame Zeit- und Raummodelle für die Agenten vorauszusetzen, damit eine gemeinsame Basis zur Zusammenarbeit gewährleistet werden kann. Kooperative Fähigkeiten, Kommunikationsstrukturen und die Möglichkeit der Partnermodellierung werden explizit in den Agenten bzw. in deren Wissensbasen verankert.

Zur Wissensrepräsentation werden logikbasierte Wissensrepräsentationssprachen eingesetzt (s. nächstes Kapitel). Dies sind beispielsweise für die Repräsentation von Zeit Variationen der Modallogik und für allgemeines Wissen über die Objekte in der Welt Konzeptlogik.

Für die Kommunikation zwischen den Agenten wurde ein Ansatz basierend auf Sprechakten entwickelt. Dieser Ansatz gestattet den Agenten über verschiedene Objekte zu verhandeln. Die Verhandlungsführung wiederum ist ein wesentlicher Faktor bei dem Problemlöseverhalten der Agenten.

Planungsaufgaben werden durch verzögertes, bidirektionales Planen gelöst. Das heißt, daß Pläne von den Agenten nicht vollständig berechnet werden, sondern nur gewisse Teilpläne total ausgearbeitet werden, um auf neue Gegebenheiten der Welt flexibel reagieren zu können. Ferner können damit auch Teilpläne an andere Agenten delegiert werden, die als abstrakte Aktionen im Plan repräsentiert werden. "Bidirektional" bezieht sich dabei auf die Planentwicklung, die sowohl top-down -von abstrakten zu spezifischen Aktionen-, als auch bottom-up -von den Einzelfähigkeiten zu dem Gesamtplan-, stattfinden kann.

Als Implementierungssystem wird die Entwicklungsumgebung MAGSY (s. Kapitel Systeme) verwendet. Es stellt die wesentlichen Basisfunktionen, wie Prozeßgenerierung, Interprozeßkommunikation und globale Kontrollstrukturen zur Verfügung. Einfache Spielszenarien wie Towers von Hanoi oder Scrabble wurden auf SEMAW, einer Experimentierplattform für Multi-Agentensysteme, realisiert.

SZENARIEN

Ausgehend von einfachen Szenarien, wie dem der Türme von Hanoi und von Blockswelten, in denen fundamentale Fragestellungen der VKI untersucht wurden, werden in einer zweiten Phase dann Realweltprobleme angegangen. Als Szenarien sind hier insbesondere die Simulation einer Gruppe von Spediteuren mit unterschiedlichen Spezialisierungen in Transportmitteln und Transportmöglichkeiten, sowie einer Gruppe von Robotfahrzeugen mit Ladeflächen und Manipulatoren vorgesehen.

Spedition

Die Spediteure erhalten Transportaufträge, die über die Möglichkeiten und Kapazitäten des einzelnen Spediteurs hinausgehen und sollen die Aufträge gemeinsam planen und ausführen. Dabei können neben den Transportkapazitäten und unterschiedlichen Transportmöglichkeiten (LKW, Bahn, Schiff, Flugzeug) insbesondere auch zeitliche Beschränkungen (verderbliche Ware o.ä.) und eventuell räumliche Randbedingungen (Sitz der Spedition, Transportstrecken) eine Rolle spielen. In diesem Szenario muß also eine komplexere Umwelt modelliert werden. Man benötigt Wissen über die Topologie des Landes, die Verkehrsverbindungen usw., ökonomisches Wissen (Kosten, Kapazitäten), sowie Wissen rechtlicher Art (z.B. Sonntagsfahrverbot). Die Agenten sind wesentlich komplizierter, da die einzelnen Speditionen unterschiedlich organisiert und spezialisiert sind. Dies stellt im Gegensatz zu den Spielszenarien bereits hohe Anforderungen an die Wissensbasen der Agenten.

Verladestation

Hier soll zunächst als Simulation eine Gruppe von autonom agierenden Fahrzeugen in einem Verladehof, z.B. einer Speditionsfirma, modelliert werden, wobei aber durchaus das Ziel einer realen - wenigstens in kleinem Mastab - Durchführung angestrebt wird. Zu dieser Verladestation gehören reine Transportvehikel wie verschiedene LKWs, ebenso wie mit Manipulatoren bestückte, freibewegliche Fahrzeuge (etwa Gabelstapler) und auf Leitschienen bewegte Kräne. Der Verladehof als ganzes erhält einen oder mehrere Be-, Ent- oder Umladeaufträge, die dann verteilt durch die einzelnen Agenten geplant und ausgeführt werden. Dabei bewegen sich neben den Gabelstaplern und Kränen auch die LKWs - sie fahren an eine andere Stelle im Hof, fahren weg oder es kommen neue hinzu. Hier treten neben den Zuordnungsproblemen (welche Palette an welche Stelle auf welchen LKW usw.) insbesondere räumliche Beschränkungen auf: begrenzte Flächen, sich überschneidende Wegstrecken, Ladeflächen etc.

2.2.2 AKA-WINO: Logische Grundlagen der Wissensrepräsentation und Wissensverarbeitung bei autonomen kooperierenden Agenten

von H.-J. Bürckert

Aufgabenstellung: Formalisierung der Wissensdarstellung im Computer.

Das Projekt WINO ist ein Grundlagenforschungsprojekt in der Forschungsgruppe AKA (Autonome Kooperierende Agenten), in dem Fragen der Kooperation und Kommunikation in Gruppen interagierender KI-Systeme untersucht werden. Im WINO-Projekt befassen wir uns speziell mit der Frage, wie Wissen im Computer repräsentiert werden kann und wie dieser daraus Schlußfolgerungen ziehen kann. Dabei geht es nicht darum, den Computer mit Daten, etwa über Autos, zu füttern, sondern ihm echtes Wissen darüber zur Verfügung zu stellen, was Autos sind, welche Eigenschaften sie haben und was man damit anfangen kann. Ein mit solchen Fähigkeiten und solchem Wissen versehenes System sollte dann in der Lage sein, aus seinem Wissen selbständig Schlüsse zu ziehen: Wenn das System etwa Informationen über die Verfügbarkeit eines speziellen Fahrzeuges mit großer Ladefläche und Geländegängigkeit hat, sollte es damit folgern können, daß dieses Fahrzeug zum Transport von Salzsäcken durch die Wüste eingesetzt werden kann. Wichtig ist dabei, daß das System weiß, daß es über solches Wissen verfügt, und insbesondere, daß es auch weiß, wovon es keine Ahnung hat: Wenn es über keinerlei Wissen über, sagen wir, Tiere verfügt, sollte es einen Benutzer auf eine entsprechende Anfrage auch darüber informieren,

daß es nicht weiß, was eine Kamelkarawane ist und daher auch nicht weiß, ob eine solche für den Salztransport durch die Wüste eingesetzt werden könnte. WINO (Wissens- & Inferenz-Objektivierung) zielt dabei auf eine Objektivierung von Wissensrepräsentationsformalismen für die Darstellung insbesondere von begrifflichem Wissen und dessen Verarbeitung mit Methoden der Logik:

1. Erforschung der logischen Grundlagen von Wissensrepräsentationsformalismen,
2. Entwicklung adäquater Inferenzmechanismen für die Wissensrepräsentation,
3. Operationalisierung durch effizient realisierbare Kalküle,
4. Prototypische Implementierung eines Wissensrepräsentations- und Inferenzsystems

Für die Entwicklung eines Wissensrepräsentations- und Inferenzsystems sind zwei prinzipielle Vorgehensweisen möglich: Einerseits können bewährte taxonomische Wissensrepräsentationssprachen wie KL-ONE benutzt und um darauf zugeschnittene Inferenzverfahren erweitert werden. Andererseits ist es möglich, hocheffiziente Inferenzkomponenten, wie sie für logische Programmiersprachen entwickelt wurden, zum Ausgangspunkt zu nehmen und um Datenstrukturen zu erweitern, die zur Wissensrepräsentation eingesetzt werden:

1. Erweiterung logischer Programmiersprachen um Wissensstrukturen.
2. Kombination taxonomischer Wissensstrukturen mit Inferenzmechanismen.

Vorrangig wurde und wird in WINO jedoch der zweite Ansatz verfolgt. Hier wurden Inferenzverfahren für terminologische Logiken -das sind Konzeptbeschreibungssprachen- entwickelt, die auch bereits eine prototypischen Realisierung durch das System KRIS (Knowledge Representation and Inference System) fanden. KRIS ermöglicht die Darstellung und Verarbeitung von Begriffswissen durch effiziente Verfahren.

Lösungsansatz: Klassenorientierte Darstellung von Begriffen und Begriffsbeziehungen

In WINO haben wir uns zunächst mit formalen Methoden zur Repräsentation des rein begrifflichen Wissens von Anwendungsbereichen, der Strukturierung solchen Wissens durch auf den Begiffen und ihren Eigenschaften basierenden Klassenbildung und die sich daraus ergebenden Teilklassen-Hierarchien, sowie mit den solche Strukturierung unterstützende bzw. ausnützende Schlußfolgerungsverfahren beschäftigt.

Dabei wird begriffliches Domänenwissen durch Klassen (Konzepte) und Instanzen der Klassen (Objekte) dargestellt, wobei als Beschreibungselemente die üblichen Mengenoperationen (Vereinigung, Schnitt, Differenz oder Komplement), sowie funktionale und relationale Attribute (Features und Rollen) verwendet werden. Es gilt nun Verfahren zu finden, die zum einen die Darstellung solchen Wissens im Computer unterstützen: durch Feststellen der Teilklassenbeziehungen, die sich aus definierenden Eigenschaften der Begriffe ergeben; durch Überprüfung der Sinnhaftigkeit definierter Begriffe (enthalten die dadurch beschrieben Klassen überhaupt irgendwelche Objekte), bzw. der Sinnhaftigkeit des repräsentierten Wissens an sich (enthält es auch keine widersprüchlichen Definitionen). Zum anderen braucht man Verfahren, die aus diesem Begriffswissen evtl. nur implizit vorhandenes Wissen ableiten können, um etwa Anfragen beantworten zu können oder neue Begriffe in die vorhandene Begriffshierarchie einzuordnen, d.h.

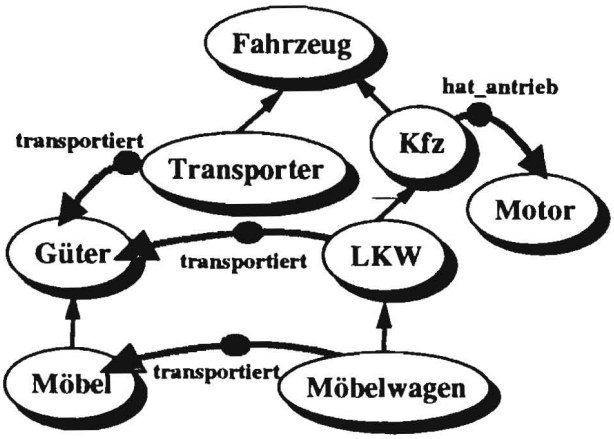


Abbildung 1: Ausschnitt aus einer Wissensbasis für das Speditionsszenario mit hierarchisch strukturiertem Begriffswissen über Transportfahrzeuge mit Unterklassenbeziehungen und definierenden Eigenschaften. Unterklassen erben die Eigenschaften der Oberklassen und werden über die Unterklassenbeziehungen und ihre angegebenen Eigenschaften definiert. So sind z.B. LKWs definiert als Kfz, die Güter transportieren können, und erben natürlich die Eigenschaft einen Motorantrieb zu besitzen. Das Wissensverarbeitungssystem muß an diesem einfachen Beispiel etwa folgern können, daß LKWs eine Teilklasse der Transporter sind: LKWs sind als Kfz auch Fahrzeuge. Zusammen mit der Eigenschaft, daß LKWs Güter transportieren, folgt daraus, daß LKWs spezielle Transporter sind.

zu klassifizieren. Zusammengenommen hat man also etwa folgende Hauptinferenzen: Zum einen sollen definierte Klassen daraufhin getestet werden, ob sie überhaupt Instanzen besitzen können. Zum anderen muß natürlich die Wissenbasis als solche auf Widerspruchsfreiheit überprüft werden. Darüberhinaus sind Klassenbeziehungen bzw. speziell die Zugehörigkeit gewisser Objekte zu Klassen festzustellen (Klassifikation), gemeinsame Eigenschaften der Objekte einer Klasse zu finden (Vererbung), sowie alle Instanzen einer Klasse zu errechnen (Retrieval). In WINO haben wir für diese zunächst unterschiedlichsten Inferenzformen ein einheitliches, auf einem korrekten und vollständigen Kalkül basierendes Inferenzschema entwickelt, das für ausdrucksstarke Begriffsbeschreibungssprache mit obigen (und zum Teil weiteren) Beschreibungselementen effizient realisierbare Entscheidungsverfahren zur Verfügung stellt. Wir befassen uns im Projekt auch mit der Einbeziehung räumlicher und zeitlicher Begriffe, sowie mit der Modellierung von Wissen über vorhandenes (Begriffs-)Wissen. Ferner untersuchen wir Methoden und Verfahren zur Verarbeitung unvollständigen Wissens, etwa nur partiell definierbarer Begriffe, unsicherer oder vager Informationen (wie ‚hohes Fieber', oder ‚heute hat es mal wieder viel geregnet'). Dabei stützen wir uns insbesondere auf Methoden und Verfahren aus der Logik, in der es eine ganze Reihe geeigneter Modellierungsansätze für solches Wissen gibt. Unsere Aufgabe besteht darin, für diese Ansätze geeignete und vor allem effiziente Schlußfolgerungsverfahren zur Verarbeitung des Wissens zu entwickeln und diese in ein entsprechendes Wissensrepräsentations- und Inferenzsystem einzubauen.

Anwendungen: Wissensbasen für begriffliches Wissen

Das eingangs erwähnte Beispiel über den Transport von Salzsäcken durch die Wüste dürfte auch schon andeuten, wo und wie solche Wissensrepräsentationssysteme eingesetzt werden können: natürlich überall dort, wo nicht nur abgespeicherte Daten ausgegeben werden, und deshalb auch als Kern-System für die Wissensverarbeitung in KI-Systemen. Aber auch in der Datenbankforschung zeigt sich immer mehr die Notwendigkeit, komplex strukturierte Daten in Datenbanken

darzustellen (Sprachverarbeitung, CAD, Ingenieursysteme). Daraus hat sich eine Konvergenz mit Ansätzen aus der Künstlichen Intelligenz und hier besonders mit den Forschungen zur Wissensrepräsentation ergeben.

- Nachteile des relationalen Modells:
  - "flache", unstrukturierte Daten
  - Information muß im Regelfall vollständig sein
- Vorzüge unseren Ansatzes:
  - Strukturierung durch Hierarchien und Attribute
  - Darstellung unvollständiger Information ist möglich
  - ebenfalls logikbasiert (wie der relationale Ansatz), daher einfach mit logischen Schlußregeln wie in PROLOG zu verbinden

Übliche, etwa objektorientierte Ansätze, verwenden Information über die Struktur der Domäne (wie Hierarchien, Vererbung, Existenz und Nichtexistenz von Attributen) als Integritätsbedingungen, d.h. das System überprüft, ob diese vordefinierten Bedingungen eingehalten sind oder nicht und meldet im zweiten Fall einen Fehler. Die Information werden also nur passiv genutzt. Wir arbeiten daran, sie auch aktiv zu nutzen für Aufgaben wie Feststellen von Vererbungsbeziehungen, Beantworten von Anfragen (Stichwort: Intensionale Antworten) und Anfrage-Optimierung. Wissensdarstellung mit Konzeptsprachen ermöglicht flexible Anfragesprachen - der Benutzer kann schon Anfragen stellen, wenn er die grundlegenden Beziehungen und Attribute zwischen Objekten kennt. Sie geben darüberhinaus eine Grundlage für intensionale Antworten auf Datenbank-Anfragen (Beschreibung einer Menge von Antworten durch ihre Eigenschaften statt Aufzählung aller Elemente). Für intensionale Antworten ist eine ausdrucksstarke Sprache zur Beschreibung der Strukture der jeweiligen Domäne zentral. Hier könnten die Untersuchungen im WINO-Projekt Anwendung bei der Erstellung eines Wissensbank-Systems für die unterschiedlichsten Wissensdomänen finden. Voraussetzung wäre lediglich die Beschreibbarkeit des zu verwendenden Wissens durch Konzeptbeschreibungssprachen. Hier ergibt sich auch der wesentliche Bezug zur AKA-Forschungsgruppe, in der die einzelnen kooperierenden Systeme Wissen über ihr eigenes Wissen und das Wissen der Partnersysteme haben müssen, um überhaupt zu sinnvoller Kooperation und Kommunikation fähig zu sein.

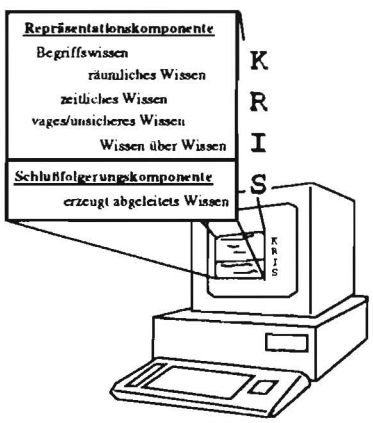


Abbildung 2: KRIS als Kern-System zur Wissensverarbeitung in den AKAs.

Ein wichtiges Anwendungsszenario in AKA soll dabei eine Gruppe kooperierender Speditionsagenturen sein (vgl. auch AKA-Mod): Verschiedene Agenturen, spezialisiert auf unterschiedliche Transportmittel bzw. Güter und beschränkt auf unterschiedliche Regionen Deutschlands oder Europas arbeiten bei der Planung und Ausführung von Transportaufträgen zusammen. Auf lange Sicht hat dieses Szenario Anwendungen von verkehrspolitischer Bedeutung, etwa beim Einsatz im Sinne elektronischer Frachtenbörsen, mit dem gemeinsamen Ziel der optimalen Transportmittel-Nutzung und-Auslastung.

3 Systeme

3.1 MEKKA: Eine Entwicklungsumgebung zur Konstruktion kooperativer Anwendungen

von D. Steiner

3.1.1 Einleitung

Mensch-Maschine Kooperation wird weder im Bereich der Verteilten Künstlichen Intelligenz (VKI) noch im Bereich Computer Supported Cooperative Work (CSCW) wirklich thematisiert.

Zur Überbrückung dieser Lücke ist es notwendig, einen Rahmen zu schaffen, der das Zusammenwirken von Menschen und maschinellen Subsystemen als Partner in der Kommunikation, in der Kooperation, beim Problemlösen und bei der Ausführung verteilter Aufgaben unterstützt. Ein solcher Ansatz erweitert und kombiniert bestehende Ansätze zu einem neuen Paradigma der kooperativen Mensch-Maschine Arbeit, HCCW (Human Computer Cooperative Work) [SMH90, G+91]. Eine wesentliche Herausforderung an solche intelligenten HCCW-Systeme stellt die Einbettung von leistungsfähigen und flexiblen Kooperationsmethoden und speziell dafür konstruierten Unterstützungsmechanismen dar.

Die Anwendungszenarien, für deren Beherrschung die Methoden der Mensch-Maschine Kooperation notwendig erscheinen, sind durch folgende Eigenschaften charakterisiert:

Verteilung Die Zusammenarbeit im Team erfordert die Berücksichtigung unterschiedlicher Dimensionen der Verteilung:

- **Raum** Die Teammitglieder arbeiten an verschiedenen Orten.
- **Zeit** Die Teammitglieder arbeiten in verschiedenen, u.U. nicht-überlappenden Zeitintervallen.
- **Kompetenz** Die Fähigkeit, die globale Aufgabe zu lösen, ist über das gesamte Team verteilt, jedoch können einzelne Gruppen bestimmte Teilaufgaben lokal lösen.
- **Verantwortung** Die Verantwortung für verschiedene Aspekte des Lösungsprozesses kann auf einzelne Mitglieder oder Gruppen im Team verteilt sein.

Heterogenität Ein Team kann sich aus einer Vielzahl unterschiedlicher Mitglieder zusammensetzen: einfache Sensoren, komplexe Softwaresysteme und Menschen; ihre Funktionalitäten und Fähigkeiten unterscheiden sich drastisch.

Dynamik Die kooperativen Rollen der einzelnen Agenten sind nicht vollständig auf der Basis einer festen organisatorischen Struktur vorbestimmt. Vielmehr kristallisieren sie sich dynamisch im Laufe des Arbeitsprozesses heraus, je nach Fähigkeiten der Beteiligten und der Struktur der zu erfüllenden Aufgabe.

Replikation Die Kompetenz der verschiedenen Agenten kann überlappen oder sogar konkurrieren, was zu erhöhten Anforderungen an die Modellierung des Kooperationsprozesses führt.

Stabilität Das Team soll auch dann noch zusammenarbeiten können, wenn einzelne Kommunikationskanäle oder einzelne Teammitglieder ausfallen; natürlich müssen dann Einschränkungen bzgl. der Arbeitsweise und der Effizienz in Kauf genommen werden.

Um die logischen Aspekte der Verteilung und der Kooperation in solchen Systemen zu unterstützen, wird die Umgebung *MEKKA* (Mehr-Agenten Entwicklungsumgebung für die Konstruktion Kooperativer Anwendungen) entworfen [DH+90]. MEKKA wird die Beschreibung der kommunikativen, kooperativen und funktionalen Fähigkeiten eines Agenten, sowie die Spezifizierung der kooperativen und kommunikativen Struktur eines Mehr-Agentensystems (MAS) erlauben. Die Arbeiten werden auch im Rahmen des ESPRIT II Projektes 5362, IMAGINE (Integrated Multi-Agent Interactive Environment) durchgeführt.

3.1.2 Agentenmodell

Zu Beginn des Projektes wurde ein **Agentenmodell** entworfen, mit dem die zwei wesentlichen Rollen von Agenten im Kontext von MASen beschrieben werden können:

- die Ausführung zugeteilter Aufgaben
- die Teilnahme am Kooperationsprozeß

Aus der Anforderung, daß ein Agent mit anderen Agenten kommunizieren können muß und in geeigneter Weise (kooperativ) auf empfangene Nachrichten reagieren muß, läßt sich die Struktur eines Agenten unmittelbar ableiten (vgl. auch Abbildung).

Körper Im Körper des Agenten ist seine individuelle Problemlösefähigkeit manifestiert. Hier werden grundlegende Funktionen beschrieben, die er auch ohne Einbettung in eine Kooperationsstruktur ausführen kann. Ein Körper kann eine spezielle Hardware (z.B. ein Sensor oder ein Drucker), Software (z.B. ein Finanzberatungsprogramm oder eine Datenbank) oder auch ein Mensch sein. Ein Körper kann gleichzeitig Teil verschiedener Systeme sein; Wiederverwendbarkeit und Duplizierung von (maschinellen) Agentenkörpern ist also allgemein möglich.

Kommunikator Um mit anderen Agenten kommunizieren zu können, benötigt ein Agent Zugang zu geeigneten Kommunikationskanälen und zu Informationen über das Netzwerk, wie z.B. die Agentenadressen. Der Kommunikator stellt diese Kanäle bereit und verwaltet die Netzwerkinformationen. Seine Hauptaufgabe ist das Empfangen, Vorverarbeiten und ggf. Weiterleiten von Nachrichten an den Kopf, sowie das Aussenden der Nachrichten, die vom Kopf kommen.

Kopf Der Kopf des Agenten ermöglicht dem Agenten, an kooperativen Prozessen teilzunehmen. Maschinelle Agenten sind hierzu mit dedizierten Softwarekomponenten ausgestattet. Für die Einbindung von Menschen in einen kooperativen Prozeß ist deren eigenes Wissen in Kombination mit geeigneten Schnittstellen zum System relevant.

Da wir von einem zielgerichteten Ansatz kooperativen Arbeitens ausgehen, benötigen wir ein Modell des Problemlöseprozesses. Dieser läßt sich allgemein durch eine Folge *abstrakter generischer Aktivitäten* beschreiben [BCM90], die unabhängig von der konkreten Anwendungsdomäne sind:

Initialisierung resultiert in der Etablierung eines Ziels für einen oder mehrere Agenten. Umfangreiche Kommunikation zur Abstimmung von Interessen und Ansichten kann dazu nötig sein.

Planung beinhaltet die Entwicklung und Bewertung alternativer Vorgehensweisen, die Festlegung eines Aktionsplanes, die Zuordnung von Ressourcen usw.

Ausführung meint die Umsetzung eines Plans, wozu auch die Überwachung der Aktionen zwecks Ermittlung von Diskrepanzen zwischen erzielten und erwarteten Effekten gehört.

Bewertung erzielter Ergebnisse eines Plans oder einzelner Aktionen hilft, Schwierigkeiten aktueller Pläne aufzudecken und kann zur Aufstellung neuer Ziele führen.

Jeder Agent (Problemlöser) arbeitet nach diesem Ablaufschema. Ein völlig autonomer Agent (ohne Einbettung in eine kooperative Umgebung) benötigt hierzu Wissen über:

- seine eigenen Fähigkeiten (sog. autoepistemisches Wissen) und
- die aktuelle Aufgabe.

In einer Multiagentenumgebung ist dieses Schema ebenso gültig, jedoch ist der gesamte Prozeß auf mehrere Agenten verteilt und es können gleichzeitig mehrere Problemlöseprozesse ablaufen. Der Kopf des Agenten vermittelt zwischen der Funktionalität des Agentenkörpers und den allgemeinen Aufgabenstellungen (den Kooperationskontexten). Um aktiv zu kooperieren, muß der Kopf über das oben erwähnte Wissen hinaus zusätzliches Wissen über relevante Aspekte der Kooperation besitzen. Dies ist zum einen global definiert, wie etwa:

- a priori vorhandene Kooperationsmethoden und -primitive[1], aus denen er bei Bedarf neue Kooperationsstrukturen konstruieren kann,

oder wird durch die Partizipation an einem spezifischen kooperativen Kontext bestimmt:

- Fähigkeiten anderer Agenten,
- Kommunikationsmöglichkeiten,
- seine eigene(n) Rolle(n),
- Rollen der Agenten, mit denen er kooperiert,
- die Kooperationsstrukturen, an denen der Agent teilnimmt.

Dieses Wissen eines Agenten ist bei seiner Erzeugung entweder bereits vorgegeben, oder es wird dynamisch akquiriert, z.B. durch Übermittlung zwischen Agenten im Verlauf einer kooperativen Aufgabenbearbeitung. Wissen, insbesondere über andere Agenten, kann unvollständig sein.

3.1.3 Kooperationsmodell

Das **Kooperationsmodell** stellt den zweiten Pfeiler dar, auf dem MEKKA basiert. Es besteht aus zwei unterschiedlichen Ebenen, den *Kooperationsprimitiven* und den *Kooperationsmethoden.*

[1] Zur Definition dieser Begriffe siehe Paragraph über das Kooperationsmodell.

Kooperationsprimitive Die im Problemlösungsprozess in einem Mehragentensystem auftretenden Objekte, z.B. Absichten, Ziele, Aufgaben, Resourcenzuweisungen, Ergebnisse, werden durch die Verteilung in MASen zu Verhandlungsgegenständen. Die Verhandlung über diese sog. Kooperationsobjekte wird durch typisierte und strukturierte Nachrichten, sog. *Kooperationsprimitive*, unterstützt. Sie setzen sich aus Nachrichtentyp und Verhandlungsgegenstand zusammen [Ste91, HS91]. Die folgende Aufzählung ist nicht vollständig; sie dient zur Veranschaulichung der intendierten Funktionalität:

`propose`: Ein Vorschlag bezüglich eines bestimmten Gegenstandes (Ziel, Aufgabenzuordnung, Ressourcenzuordnung, ...) wird an einen oder mehrere Agenten übermittelt. Randbedingungen können spezifiziert werden, um Verklemmungen zu vermeiden, Zeitbeschränkungen einzuhalten oder den Kooperationsprozeß zu steuern. Ein `propose` startet einen Verhandlungsprozeß bezüglich des Verhandlungsgegenstandes.

`accept`: Ein `propose` wird in der zuletzt vorgeschlagenen Form akzeptiert, die Verhandlung darüber beendet.

`reject`: Ein `propose` wird abgelehnt. Durch eine Modifikation des abgelehnten Vorschlags kann eine Verhandlung weitergeführt werden.

`request`: Ein `request` übermittelt eine Forderung, ohne diese zum Gegenstand weiterer Verhandlungen zu machen.

`refine`: Ein `refine` verfeinert das Kooperationsobjekt eines vorangegangenen Vorschlags.

`tell`: Informationen werden an Agenten übermittelt, ohne die Absicht, diese zum Gegenstand einer Verhandlung zu machen.

`support`: Ein vorgeschlagenes Ziel wird unterstützt.

`oppose`: Einem vorgeschlagenen Ziel wird widersprochen.

Kooperationsmethoden Durch die Komposition von Kooperationsprimitiven entstehen die sog. Kooperationsmethoden. Die Kooperationsmethoden stellen einen gemeinsamen Rahmen zur Teilnahme der Agenten innerhalb eines Kooperationsprozesses dar. Man kann sie als Regeln/Vorschriften ansehen, nach denen die Agenten effizient eine Kooperation durchführen können. Die aus den Bereichen VKI und CSCW bestens bekannten Kooperationsmethoden wie *master-slave, contract net, negotiation*, usw. können ebenso leicht aus den Kooperationsprimitiven gebildet werden wie neue Kooperationsmethoden (z.B. zur Terminvereinbarung, s. Vortrag A. Lux). Kooperationsmethoden haben Parameter zur Initialisierung der beteiligten Agenten und des Kooperationsobjektes.

Vordefinierte Kooperationsmethoden werden in Bibliotheken gespeichert und können von dort von den Agenten abgerufen werden. Auf diese Weise können auch maschinelle Agenten leicht darauf zugreifen und über die ablaufende Kooperation Schlußfolgerungen ziehen.

Darüberhinaus soll es möglich sein, daß Menschen dynamisch eigene Kooperationsmethoden definieren oder modifizieren. Dadurch ist gewährleistet, daß der Ablauf innerhalb eines Kooperationsvorgangs den menschlichen Bedürfnissen angepaßt ist.

Das Hauptproblem des vorgestellten Kooperationsmodells besteht darin, wie eine *einzelne* Nachricht im Kontext einer gerade ablaufenden Koperation klassifiziert werden kann. Zu diesem Zweck müssen Regeln festgelegt werden, die beschreiben, wie Kooperationsmethoden aus Kooperationsprimitiven definiert werden können.

3.1.4 Ausblick

Zusammenfassend läßt sich festhalten, daß der in MEKKA gewählte Ansatz folgende Hauptvorteile bietet:

- Vordefinierte organisatorische Strukturen und Kooperationsformen können leicht mittels der Kooperationsprimitive und -methoden modelliert und unterstützt werden.
- Menschen und Maschinen können reibungslos in einen gemeinsamen Arbeitsfluß integriert werden.

Zur Zeit werden das Agenten- und Kooperationsmodell auf ihre praktische Anwendbarkeit hin überprüft. Auf einer vernetzten Umgebung von Unix Workstations werden auf der Basis von PARLOG/Prolog (siehe Vortrag A. Burt) kooperative Szenarien aus den Bereichen Verkehrswesen und Terminvereinbarung (siehe Vortrag A. Lux) prototypisch implementiert. Die dort gewonnenen Erkenntnisse werden im Projekt IMAGINE zur Realisierung von Mensch-Maschine-Systemen in den Bereichen Luftverkehrskontrolle und Netzwerkmanagement eingebracht.

Literatur

[BCM90] N. Bhandaru, W. B. Croft, and D. E. Mahling. A Task-Centered Theory of Cooperative Work. Technical Report 90-25, COINS, University of Massachusetts, Amherst MA 01003, March 1990.

[DH+90] C. Dietel, H. Haugeneder, et al. KIK Projektbeschreibung (Version 2). Technical report, Siemens AG, 1990.

[G+91] Paul De Greef et al. Analysis of Human-Computer Cooperative Work. Technical Report 4, IMAGINE (Esprit Project 5362), 1991.

[HS91] Hans Haugeneder and Donald Steiner. Cooperation Structures in Multi-Agent Systems. In W. Brauer and D. Hernández, editors, *Verteilte Künstliche Intelligenz und kooperatives Arbeiten*. 4. Internationaler GI-Kongreß Wissensbasierte Systeme, Springer-Verlag, 1991.

[SMH90] Donald Steiner, Dirk Mahling, and Hans Haugeneder. Human Computer Cooperative Work. In M. Huhns, editor, *Proc. of the 10th International Workshop on Distributed Artificial Intelligence*. MCC Technical Report Nr. ACT-AI-355-90, 1990.

[Ste91] Donald Steiner, editor, A Preliminary Agent Model. Technical Report 7, IMAGINE (Esprit Project 5362), 1991.

3.2 MAGSY

von K. Fischer

Der Abschnitt stellt das regelbasierte Multiagentensystem MAGSY vor. Die Agenten dieses Multiagentensystems kommunizieren und kooperieren, indem sie Faktenwissen in Form von Nachrichten austauschen und gegenseitig Dienstleistungen in Anspruch nehmen und erbringen. Das globale Faktenwissen ist in einer objektorientierten Wissensbasis abgelegt, die allen Agenten gleichermaßen zugänglich ist. Bei der Beschreibung wird zuerst erläutert, für welches

Anwendungsgebiet das System primär konzipiert wurde. Dann wird das theoretische Konzept präsentiert und daran anschließend erläutert, welche Vorteile das System für den praktischen Einsatz bietet.

3.2.1 Einleitung

Der Agentenbegriff, wie er im weiteren verwendet wird, trifft die in [Martial 92] gegebene Definition. Ein Agent wird als abstrakte (Rechen-)Einheit mit Problemlösungsfähigkeiten definiert. Weil in MAGSY der Kern eines Agenten durch einen vollständigen Regelinterpreter gegeben ist, hat jeder Agent potentiell die Problemlösungsfähigkeiten eines Expertensystems.

Von den einzelnen Agenten einer Multiagentenanwendung müssen müssen in der Regel komplexe Einzelaufgaben geplant werden können, bei deren Lösung sie miteinander kooperieren müssen. Dazu ist es notwendig, daß die Agenten Wissen austauschen und auf die Aktionen der jeweiligen Kooperationspartner unmittelbar reagieren können. Dies erfordert für die einzelenen Agenten die Fähigkeit auf externe Ereignisse asynchron reagieren zu können, also externes Wissen asynchron verarbeiten zu können. Konkret verstehen wir unter asynchroner Wissensverarbeitung, daß in ein Agent zu einem beliebigen Zeitpunkt neue Fakten entstehen können, die sofort in den Planungsvorgang einbezogen werden, damit der Agent in der Lage ist, auf die veränderte Situation zu reagieren.

Das Multiagentensysstem MAGSY unterstützt genau die eben motivierte asynchrone Wissensverarbeitung und bietet die Möglichkeit zur Parallelprogrammierung. Das allen Agenten zugängliche globale Faktenwissen ist in einer objektorientierten Wissensbasis abgelegt. Ein erster Prototyp von MAGSY wurde mit Hilfe der Regelprogrammiersprache OPS5 für einen VAX/VMS-Cluster implementiert [Fischer 91]. Die einzelnen Agenten sind Regelprozesse, die beliebig auf die Knoten des Cluster verteilt werden können. Sie können untereinander Faktenwissen austauschen, nehmen gegenseitig Dienstleistungen in Anspruch und erbringen Dienstleistungen. Die Möglichkeiten zur Parallelverarbeitung waren nur schwer in das OPS5-System zu integrieren, da es dafür in OPS5 keine eigenen sprachlichen Ausdrucksmittel gibt. Der erste Prototyp bot deshalb für die Implementierung von Anwendungen nicht viel Komfort. Aus diesem Grund wurde in [Windisch 91] ein Regelinterpreter für eine UNIX-Umgebung entworfen, der die vorwärtsschließende Semantik von OPS5 beibehält, in der aber Sprachelemente für die Parallelverarbeitung eingebunden wurden. Dieser Regelinterpreter ist der Kern eines Agenten von MAGSY.

3.2.2 Definition des Agentensystems

Ein Agent A ist in MAGSY definiert als ein Tripel $A := (\mathcal{F}, \mathcal{R}, \mathcal{D})$, mit:

- $\mathcal{F}$ ist eine Menge von Fakten, die das lokale Faktenwissen des Agenten repräsentiert. Ein Faktum ist dabei als Eingenschaftenliste zu verstehen, die bestimmten Bezeichnern symbolische Werte zuweist.
- $\mathcal{R}$ ist eine Menge von Regeln, die Strategien für das allgemeine Verhalten des Agenten vorgeben.
- $\mathcal{D}$ ist eine Menge von Diensten, die der Agent als Schnittstelle nach außen anbietet.

Für einen Agenten A soll durch $A_{\mathcal{F}}, A_{\mathcal{R}}$ und $A_{\mathcal{D}}$ die jeweilige Menge angesprochen werden können. A kann Nachrichten empfangen, die entweder die lokale Faktenmenge $A_{\mathcal{F}}$ verändern

oder ein Dienst der Menge $A_{\mathcal{D}}$ aktivieren. Jeder Dienst eines Agenten besteht aus einer Menge von Regeln, die genau dann aktiv sind, wenn der zugehörige Dienst durch das Empfangen einer Botschaft aktiviert wurde. Bei Ausführung eines Dienstes des Agenten A, kann sich sein Faktenwissen $A_{\mathcal{F}}$ und sein Regelwissen $A_{\mathcal{R}}$ verändern. Der Agent lernt also, indem er Dienstleistungen erbringt, wobei das Lernen auch darin bestehen kann, hinfällig gewordene Fakten und Regeln zu löschen. Weiter kann durch das Ausführen eines Dienstes des Agenten A eine Menge von neuen Agenten entstehen. Die bei einem Agenten eingehenden Dienstbotschaften werden nach einer prioritätengesteuerten FIFO-Strategie abgearbeitet. Dabei ist immer der älteste Dienst mit der höchsten Priorität aktiv. Alle weiteren Dienste werden in einer Warteschlange verwaltet. Trifft eine Dienstbotschaft für einen Dienst mit einer höheren Priorität als der des aktuell aktiven Dienstes ein, so wird der gerade aktive Dienst unterbrochen und der neue höherpriorisierte Dienst wird aktiv. Der so unterbrochene Dienst wird wieder aktiv, wenn alle höherpriorisierten Dienste beendet sind.

Zur Modellierung des die Problemlösung beschreibenden Fakten- und Regelwissens wird eine objektorientierte Wissensbasis verwendet [Boc87, Bo/Me 88], weil sich hier das Wissen kompakt strukturieren läßt. Objekte sind die Basiseinheit, aus denen die Wissensbasis aufgebaut wird. In ihrer Struktur gleiche Objekte werden zu Klassen zusammengefaßt. Die Beschreibung einer Klasse ist damit ein prototypisches Objekt, das angibt, wie die Elemente (man nennt diese Elemente Instanzen) dieser Klasse strukturiert sind. Die Eigenschaften der Instanzen werden durch Attribute beschrieben, die ihrerseits durch Facetten beschrieben werden. Die Klassen sind in einer Hierarchie angeordnet, in der Attribute von übergeordneten, allgemeinen Klassen an untergeordnete, spezielle Klassen vererbt werden. Es lassen sich Integritätsbedingungen formulieren, die objektübergreifende Konsistenz zusichern können, weil sie bei Veränderungen automatisch geprüft werden.

Zur Modellierung der Problemlösung für eine bestimmte Aufgabenstellung wird eine Menge von kooperierenden Agenten spezifiziert (siehe Abbildung 3). Die statische Beschreibung der Problemlösung ist dabei durch das Prototypensystem $\mathcal{AS}_{prot} := (\mathcal{A}_{prot}, W_{init})$ gegeben, mit:

$\mathcal{A}_{prot}$ ist die Menge der prototypischen Agenten; das sind die Agenten, die dynamisch im Agentensystem entstehen können, d.h. die als Problemlöser dem System potentiell zur Verfügung stehen. Es können dabei mehrere Inkarnationen von einem bestimmten prototypischen Agenten in das System eingebracht werden.

W_{init} beschreibt den Initialzustand der Wissensbasis.

Die Aufgabenbearbeitung beginnt mit dem Initialsystem $\mathcal{AS}_0 := ((A_1, ..., A_n), W_{init})$, mit $A_1, ..., A_n \in \mathcal{A}_{prot}$. Aus dem Initialsystem $\mathcal{AS}_0$ entsteht nach t Zeiteinheiten dynamisch das System $\mathcal{AS}_t$ (siehe Abbildung 3). Bei der Problemlösung erbringen die Agenten Dienste und nehmen Dienste anderer Agenten in Anspruch. Die Agenten kooperieren dabei, indem sie sich gegenseitig Nachrichten zusenden. Neue Agenten können dynamisch in das System eingebracht werden — dabei können beliebig viele Inkarnationen eines prototypischen Agenten in das System eingebracht werden — und wieder aus dem System entfernt werden. Die Identifikation von Agenten kann Inhalt von Botschaften sein, so daß sich beliebige Kommunikationsstrukturen aufbauen lassen. Das Wissen der Wissensbasis ist allen Agenten zugänglich. Damit kann ein Agent seine Identifikation allgemein bekanntmachen, indem er sie in die Wissensbasis einträgt.

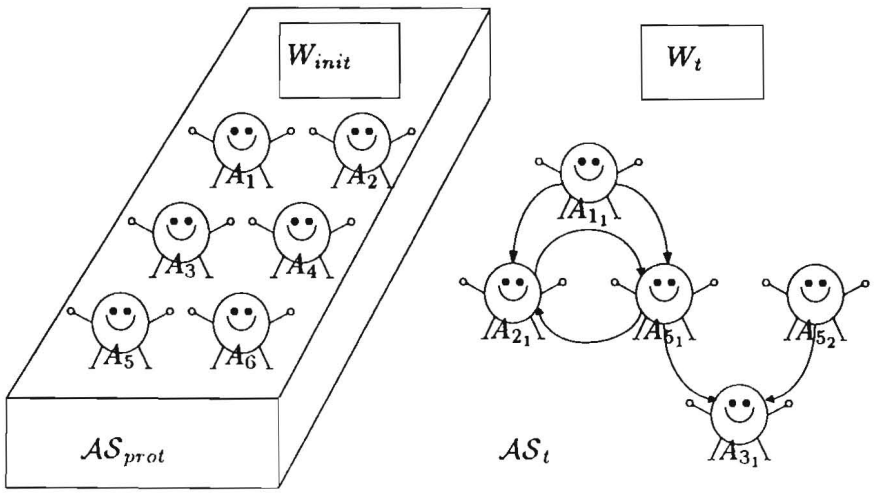


Abbildung 3: Statische Beschreibung der Problemlösung $\mathcal{AS}_{prot}$ und dynamische Ausführung $\mathcal{AS}_t$

3.2.3 Anforderungen an den Regelinterpreter von MAGSY

MAGSY besteht neben der objektorientierten Wissensbasis aus einer Menge von prototypischen Agenten. Jeder dieser Agenten wird durch ein Regelprogramm mit einer entsprechenden initialen Regel- und Faktenmenge realisiert. Die Verwendung von Regelprogrammen für die Implementierung der einzelnen Agenten setzt jedoch voraus, daß der Regelinterpreter bestimmte Anforderungen bezüglich seiner Möglichkeiten zur Kommunikation mit anderen Regelprogrammen erfüllt. Sollen echtzeitfähige Anwendungen realisiert werden, so ist die Ausführungsgeschwindigkeit der Regelprogramme ein wichtiges Kriterium für die Tauglichkeit eines Regelinterpreters. Zur Vermeidung langer Laufzeiten von Regelprogrammen können nach [Boc89] drei Ansätze verfolgt werden:

1. Parallelisierung des Mustervergleichs
2. Paralleles Feuern mehrerer Regeln
3. Aufteilung eines Regelprogramms in Teilaufgaben, die parallel abgearbeitet werden

Die ersten beiden Ansätze lassen sich nicht ohne aufwendige Betrachtung des zu implementierenden Programms bzw. nicht ohne Verwendung von Multiprozessorsystemen sinnvoll realisieren. Der dritte Ansatz erfordert jedoch lediglich eine Aufteilung des Regelprogramms in unabhängig zu lösende Teilstücke und kann auch bei wenigen vorhandenen Prozessoren zu einer deutlichen Beschleunigung führen.

Echtzeitfähigkeit bedeutet, daß ein Programm innerhalb einer Zeitspanne auf Eingaben reagiert, die die Erfüllung der Aufgabe des Programmes gewährleistet. Typischerweise handelt es

sich bei den Eingaben nicht um statisch festgelegte Daten, sondern um Daten, die dynamisch zur Laufzeit in externen Quellen (z.B. Sensorsubsystemen) entstehen. Diese werden zumeist asynchron an die Regelprogramme gesendet. Sollen viele asynchron eingehende Nachrichten verarbeitet werden, so kann die Echtzeitfähigkeit des Systems durch folgende Maßnahmen erhalten bleiben:

- Effiziente Verarbeitung asynchron eingehender Daten durch den Regelinterpreter.
- Aufteilung des Programmes in mehrere Teilprozesse (die u.U. auf verschiedenen Prozessoren ablaufen), so daß bestimmte Teilaufgaben auch während der Abarbeitung von Nachrichten erfüllt werden können.

Eine Anwendung, die aus vielen parallel arbeitenden Regelprogrammen besteht, birgt eine enorme Komplexität in sich. Daher ist es unbedingt erforderlich, komfortable Möglichkeiten zur Überwachung einer solchen Anwendung zu haben. Aus dem bisher Gesagten wurden für den dem Multiagentensystem MAGSY zugrundeliegenden Regelinterpreter folgende Anforderungen abgeleitet:

- **Schnittstelle zur Prozeßkommunikation**
  Ermöglicht das Austauschen von Nachrichten zur Synchronisation der Agenten sowie zur Aktivierung von Diensten.
- **Senden und Empfangen von Fakten**
  Neben Nachrichten sollten auch Fakten unter Erhaltung ihrer Semantik auf effiziente Weise zwischen den Agenten ausgetauscht werden können.
- **Überwachung von parallel arbeitenden Agenten**
  Zur Suche nach Fehlern in der Interaktion der Agenten muß es möglich sein, auch während des Programmablaufs eines Agenten dessen Status abzufragen, sowie dessen Verhalten zu beeinflussen.

3.2.4 Kommunikation zwischen Agenten

Der Mechanismus zum Nachrichtenaustausch zwischen Agenten sollte ein integraler Bestandteil des verwendeten Produktionensystems sein, um

- eine eindeutig definierte Schnittstelle anzubieten,
- zusätzlichen Programmieraufwand für die Kommunikation zu vermeiden und
- die Überprüfung der Parameter der Kommunikationsaufrufe zu ermöglichen.

In [Weikert 89] wurde ein Konzept zur Kommunikation für parallel arbeitende Regelprogramme vorgestellt. Die Kommunikation zwischen den Regelprogrammen geschieht dabei über das Senden und Empfangen von Nachrichten. Eine Nachricht ist in diesem Zusammenhang eine freie Botschaft, eine parametrisierte Dienstanforderung, eine Rückmeldung nach der Ausführung eines Dienstes oder ein in den Faktenspeicher des Empfängers einzufügendes Faktum. Durch die Möglichkeit, Nachrichten asynchron, d.h. ohne Wartezeit, zu senden, können Prozesse andere Aufgaben verrichten, solange die Rückmeldung eines angeforderten Dienstes nicht erfolgt ist. Zusätzlich ist auf diese Weise auch eine Synchronisation von Prozessen möglich. Dazu wird ein Auftrag an den Adressaten verschickt, wobei der Sender auf die Rückmeldung des

Empfängers wartet. Erhält ein Agent mehrere Dienstanforderungen, so werden diese nach einer FIFO-Strategie abgearbeitet.

Es folgt eine Aufzählung der wichtigsten Prozeduren, die die Kommunikationsschnittstelle bereitstellt. In MAGSY können diese Prozeduren als Aktionen im Aktionsteil der Regeln verwendet werden:

- CREATE-AGENT <*name*> <*node*>
  Ein neuer, parallel zum aufrufenden Agenten ablaufender Agent wird gestartet. Ein Faktum, das die Identifikation des neuen Agenten enthält, wird in den Faktenspeicher des Erzeugers eingetragen; dadurch kann dieser sofort mit dem neu entstandenen Agenten kommunizieren. Der Parameter <*name*> bezeichnet die Datei, die das ausführbare Regelprogramm enthält. Durch den Parameter <*node*> kann angegeben werden, auf welchem Knoten im Netz der Agent erzeugt werden soll. Es darf auch LOCAL oder NOT-LOCAL angegeben werden. Der Agent entsteht im ersten Fall auf dem gleichen Knoten wie der Aufrufer bzw. im zweiten Fall nach einer bestimmten Strategie auf einem anderen Knoten im Netz.

- REQUEST-SERVICE <*agent-id*> <*service-name*> <*mode*> <*fact*>
  Diese Prozedur sendet einen Auftrag namens <*service-name*> an den Agenten mit der Identifikation <*agent-id*>. Wird <*mode*> = WAIT angegeben, so wartet die Prozedur, bis ein Anwortfaktum des Agenten empfangen wird. Andernfalls kann der Auftraggeber weiterarbeiten, ohne auf eine Rückmeldung zu warten. Für <*fact*> darf eine Symbolfolge angegeben werden, die ein Faktum beschreibt, das als Parameter bei der Dienstaktivierung in den lokalen Faktenspeicher des Agenten <*agent-id*> eingetragen wird.

- SEND-FACT <*agent-id*> <*fact*>
  Diese Prozedur fügt ein Faktum asynchron in den Faktenspeicher des Agenten mit der Identifikation <*agent-id*> ein. Die Symbolfolge <*fact*> beschreibt das einzufügende Faktum. Durch die Kommunikationsschicht wird sichergestellt, daß die Semantik des Faktums im Sender und Empfänger die gleiche ist.

- SEND-ANSWER <*client-id*> <*service-id*> <*fact*>
  Diese Prozedur sendet ein Antwortfaktum an den Auftraggeber (er hat die Identifikation <*client-id*>) des Dienstes <*service-id*>. Wurde der Dienst mit <*mode*> = WAIT aufgerufen, so wird durch das Eintreffen eines Antwortfaktums der Wartezustand beendet, in dem sich der Auftraggeber seit der Anforderung des Dienstes befindet. Die Symbolfolge <*fact*> beschreibt das Antwortfaktum.

3.2.5 Asynchrones Empfangen von Nachrichten

Die gerade beschriebene Kommunikationsschnittstelle wurde von [Weikert 89] für die Regelprogrammiersprache OPS5 implementiert. Das Empfangen von Fakten funktioniert hier jedoch nur, solange der Recognize-Act Zyklus durchlaufen wird, also Regeln abgearbeitet werden. Der Recognize-Act Zyklus besteht aus den Phasen Mustervergleich, Regelauswahl und Regelausführung und arbeitet nach jeder Regelausführung eine Prozedur ab, die empfangene Nachrichten als Fakten in den Arbeitsspeicher des Empfängers einträgt. Um sicherzustellen, daß diese Prozedur durchlaufen wird, wird eine sogenannte rien-ne-va-plus-Regel eingesetzt [Kr/Ra 87], die stets dann zündet, wenn keine andere Regel ausgeführt werden kann. Diese Regel ist jedoch rein technischer Natur, d.h. sie führt keine für das jeweilige Regelprogramm wichtigen Aktionen aus. Ein Weg zur Vermeidung einer solchen Regel ist gegeben, wenn der Regelinterpreter die Fähigkeit besitzt, auch dann asynchron eingehende Fakten zu empfangen, wenn gerade keine

Regel zünden kann. Neu eingehende Fakten müßten in diesem Fall auch Regelinstantiierungen, d.h. Einträge in die Konfliktmenge, und das Anstoßen der Konfliktlösungsstrategie bewirken, so daß Regelausführungen durch die empfangenen Fakten möglich sind.

Bei OPS5 muß der Benutzer selbst durch geeignete Programmierung systemnaher Dienste für das Eintragen gesendeter Fakten in den Faktenspeicher des Regelinterpreters sorgen. Diese Verfahrensweise bietet zwar den Vorteil flexibler Gestaltung des Empfangsmechanismus, jedoch auch drei Nachteile:

- zusätzlicher Programmieraufwand
- geringerer Datendurchsatz des Empfangsmechanismus wegen der Verwaltung zusätzlicher Datenpuffer
- der Toplevel (siehe nächster Abschnitt) ist über diese Schnittstelle nicht ansprechbar

Eine in den Regelinterpreter integrierte Schnittstelle zum Empfangen von asynchron eingehenden Nachrichten behebt diese Nachteile. Der Vorteil einer flexiblen Behandlung von freien Botschaften kann dadurch bewahrt werden, daß nur ausgezeichnete Nachrichten als Fakten in den Faktenspeicher eingetragen werden. Nachrichten, die dieser Syntax nicht genügen, können dann vom Benutzer in der von OPS5 bekannten Weise, die am Anfang dieses Abschnitts beschrieben wurde, verarbeitet werden.

3.2.6 Überwachung von parallel arbeitenden Agenten

Bei OPS5 besitzt der Benutzer durch die Toplevel-Routine die Möglichkeit zur Suche und Behebung von Fehlern, indem er den Systemzustand inspiziert oder das Programm schrittweises abarbeitet. Dabei kann nur entweder die Toplevel-Routine oder der Regelinterpreter aktiv sein, nie jedoch beide gleichzeitig. Bei der Aufteilung eines Multiagentensystems in parallel arbeitende Regelprogramme bringt dieser Umstand einige Nachteile mit sich. Da ein zu untersuchender Agent angehalten werden muß, um Toplevel-Anweisungen ausführen zu können, kann er keine asynchron eingehenden Nachrichten mehr empfangen. Dadurch werden unter Umständen auch andere Agenten in ihrem Verhalten beeinflußt, so daß die Interaktion von Agenten kaum untersucht werden kann.

Ein weiteres Problem in diesem Zusammenhang stellen die Ausgaben der Toplevel-Kommandos dar. Bei OPS5 ist es zwar möglich, WATCH[2]-Ausgaben auf Dateien bzw. andere Ausgabegeräte umzuleiten, jedoch besteht diese Möglichkeit nicht für alle Toplevel-Ausgaben. Die Folge ist, daß auf dem Bildschirm angezeigte Masken überschrieben werden oder durch das Vermischen von Programmausgaben und Toplevel-Ausgaben die Übersichtlichkeit verloren geht.

Abhilfe schafft hier die Trennung von Toplevel und Regelinterpreter in zwei unabhängige, parallele Prozesse. Abbildung 4 stellt die prinzipielle Arbeitsweise des vom Regelinterpreter getrennten Toplevel-Prozesses dar. Der Toplevel-Prozeß muß dabei nicht zugleich mit dem Regelinterpreter gestartet werden, sondern erst dann, wenn gewisse Informationen über dessen Ablauf nötig sind oder dieser angehalten werden soll. Da eine Verbindung zu jedem beliebigen aktiven Regelprozeß hergestellt und auch wieder unterbrochen werden kann, ist es möglich, nacheinander mehrere solcher Prozesse zu untersuchen.

[2]Über die WATCH-Anweisung kann festgelegt werden, wie detailliert OPS5 die Abarbeitung eines Regelprogramms protokolliert.

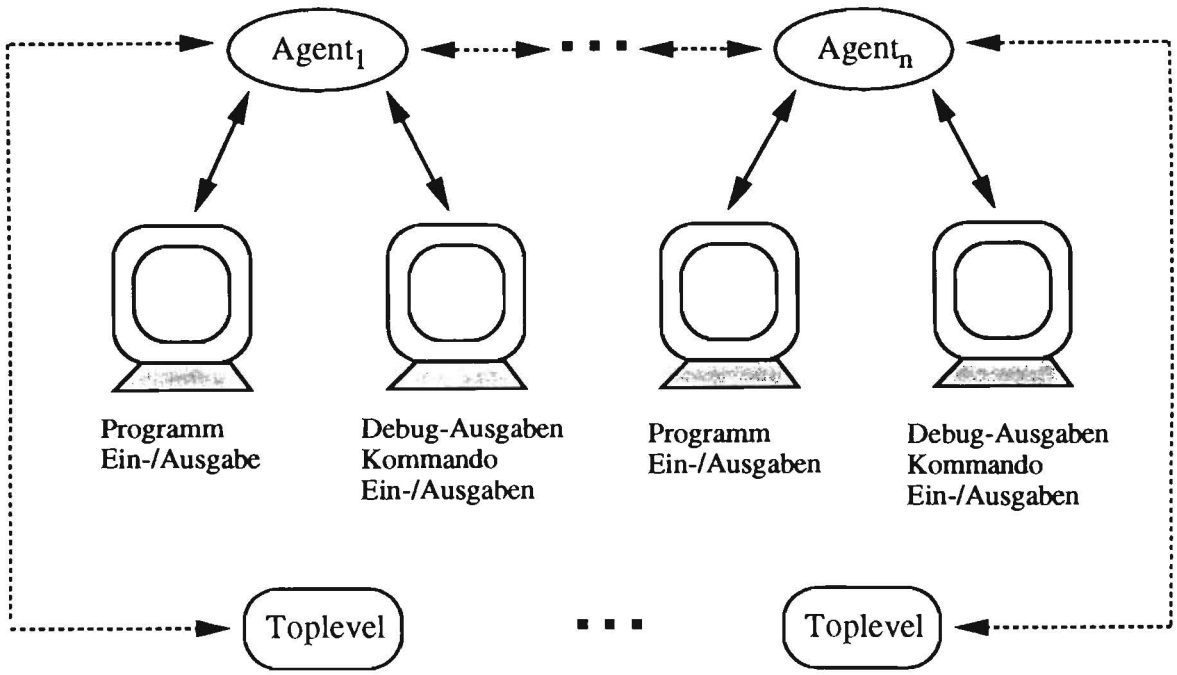


Abbildung 4: Kommunikation zwischen den Toplevel- Prozessen und den Agenten.

Nach dem Starten eines Toplevel-Prozesses meldet sich dieser bei dem gewünschten Regelprozeß an und teilt diesem das Ausgabegerät für Kommandoausgaben mit. Ab diesem Zeitpunkt kann der Benutzer an der Toplevel-Konsole Kommandos eingeben, die an den Regelprozeß weitergeleitet werden. Gleichgültig, ob der Regelprozeß aktiv oder angehalten ist, werden die Kommandos empfangen und bearbeitet. Die Ausgaben werden an die Toplevel-Konsole geschickt und der Toplevel-Prozeß erhält eine kurze Rückmeldung über die Ausführung der Kommandos. Hat der Benutzer genügend Informationen über den Regelprozeß erhalten, so kann er sich vom Toplevel-Prozeß aus abmelden und die Verbindung zum Regelinterpreter unterbrechen. Soweit erforderlich, kann zuvor der Regelprozeß terminiert werden.

Die Vorteile, die sich durch die Trennung von Kommandoebene und Regelinterpreter ergeben, kann man folgendermaßen zusammenfassen:

- Die Eingabe von Kommandos und deren Abarbeitung ist auch während des Interpreterlaufs möglich. Zum Beispiel können der Name der Regel die gerade abgearbeitet wird oder der Inhalt des Arbeitsspeichers abgefragt werden.
- Die vollständige Trennung der Ausgaben des Regelprogramms und der Antworten auf Toplevel-Kommandos ist durch die Umlenkung der Kommandoausgaben auf beliebige Ausgabegeräte gewährleistet.
- Die Möglichkeit, Verbindungen zu beliebigen Regelprozessen einzugehen und wieder zu beenden, erlaubt die Überwachung von mehreren Regelprozessen zur gleichen Zeit.

3.2.7 Ausblick

In diesem Abschnitt haben wir das regelbasierte Multiagentensystem MAGSY vorgestellt, das sich hervorragend für die Implementierung von verteilten KI-Anwendungen eignet. Die wichtigsten Eigenschaften dieses Multiagentensystems sind die komfortablen Möglichkeiten zur Kommunikation zwischen den Agenten, die weitgehenden Semantikprüfung innerhalb der einzelnen Agenten und die komfortable Möglichkeit zur Überwachung der einzelnen Agenten. Das Multiagentensystem wird in unserer Forschungsgruppe hauptsächlich für die Modellierung der Planungs- und Überwachungskomponenten einer flexiblen Fertigungssteuerung eingesetzt. Über die in [Fischer 89, Fischer 91] vorgestellten Ansätze wird hier versucht weitere Konzepte zu finden, die das Modellieren von intelligentem Verhalten bei autonomen Systemen erleichtern. Wichtig erscheint uns hier, mehr Flexibilität bei der Wissensdarstellung zu erreichen. Deshalb sollen die objektorientierten Ansätze, die bisher nur in der globalen objektorientierten Wissensbasis verwirklicht worden sind, auch in die Wissensdarstellung der einzelnen Agenten übernommen werden. Das in dieser Wissensbasis abgelegte deklarative Faktenwissen soll in Zukunft auch bei der Erzeugung von Agenten für die Semantikprüfung der Kommunikationsbeziehungen zwischen den einzelnen Agenten eingesetzt werden.

Literatur

[Bocionek 87] Bocionek, Siegfried: Dynamic Flavors, TU München, Institut für Informatik, Technischer Bericht, TUM-I8708, Juni 1987.

[Bo/Me 88] Bocionek, Siegfried und Meyfarth, Ralph: Aktive Wissensbasen und Dämonenkonzepte, TU München, Institut für Informatik, Technischer Bericht, TUM-I8811, September 1988.

[Bocionek 89] Bocionek, Siegfried: Modularisierung als Grundkonzept zur Entwicklung systemunterstützer Programmierumgebungen für parallele Regelprogramme, TU München, Institut für Informatik, Mai 1989.

[Fischer 89] Fischer, Klaus: Knowledge-Based Task Planning for Autonomous Mobile Robot Systems. Proc. of the 2nd Inter. Conf. on Intelligent Autonomous Systems, Amsterdam, Dezember 1989, S. 761–771.

[Fischer 91] Fischer, Klaus: Ein Agentensystem für eine flexible Fertigungssteuerung, Prozeßrechensysteme '91, Springer-Verlag, Berlin, Februar 1991, S. 140–149.

[Kr/Ra 87] Krickhahn, Reinhard und Radig, Bernd: Die Wissensrepräsentationssprache OPS5, Vieweg, Braunschweig 1987.

[Martial 92] Martial, Frank v.: Einführung in die Verteilte Künstliche Intelligenz, KI 1/92, 1992.

[Weikert 89] Weikert, Petra: Parallelisierung von OPS5 durch Nachrichtenübermittlung, TU München, Institut für Informatik, Februar 1989.

[Windisch 91] Windisch, Hans: Entwurf und Implementierung eines Regelinterpreters für Echtzeitplanungsaufgaben, TU München, Institut für Informatik, Mai 1991.

3.3 SEMAW

von S. Bussmann

Im folgenden wird das Simulationsentwicklungstool für Multi-Agenten Systeme SEMAW *(Simulation Environment for Multi-Agent Worlds)* in einem Überblick der wesentlichen Grundeinheiten und Funktionalitäten vorgestellt. Um unnötige Reproduktionen zu vermeiden, sollen nur auf die Hauptkonzepte **Agentendarstellung, Kommunikation** zwischen den Agenten und die **Basisfunktionalität** eingegangen werden. Die detailierte Beschreibung findet man bei Bedarf in [Bus92].

3.3.1 Einführung

Bei der Entwicklung von SEMAW wurden zwei Ziele verfolgt: Erstens sollte man in der Lage sein, daß Verhalten der Agenten deklarativ zu beschreiben (was von SEMAW nur teilweise erfüllt wird, wie man aus der Definition der Regelsätze erkennen kann). Zudem gehört auch die Anforderung, das Verhalten der Agenten in verschiedene, selbständige Bereiche aufzuteilen, und diese getrennt zu simulieren, sprich den Agenten zu modularisieren. Zu diesem Zweck wurde das Konzept der Komponenten eingeführt, welche getrennt programmiert und überwacht werden können. Zweitens sollte die Modellierung der Agenten streng von der des Szenarios getrennt werden. Das bedeutet einmal, daß eine klar definierte Schnittstelle zur Welt existiert, damit die Agenten keine unübersichtlichen, trickreichen Veränderungen des Szenarios vornehmen, und desweiteren, daß die Agenten sich nicht gegenseitig verändern, sondern nur über die Welt miteinander kommunizieren. Natürlich könnte man einwenden, daß solche programmiertechnischen Tricks (insbesondere aus Effizienzgründen) sinnvoll sein können, was auch hier nicht bestritten werden soll, doch ist die Umgebung entwickelt worden, um das Verhalten der Agenten zu untersuchen, und deshalb soll so etwas vermieden werden. In dem Rest dieses Paragraphen soll näher auf die einzelnen Konzepte von SEMAW eingegangen werden. Die Benutzeroberfläche, Hinweise auf deren Handhabung, Implementierungsbeispiele etc. sind in [Bus92] zu finden.

3.3.2 Darstellung der Agenten

Ein Agent besteht aus einer Menge von Komponenten, wobei jede Komponente aus einem Regelsatz besteht, der das Programm symbolisiert. Bei der Definition des Agenten werden die Komponenten durch Namensvergabe erzeugt. Alle Komponenten sind gleich aufgebaut, sie bestehen aus einer Kopie desselben Regelinterpreters, können aber mit verschiedenen Regelsätzen geladen werden, wobei eine Komponente nur einen Regelsatz haben kann. Die Komponenten sind also unabhängig von einander und können nur über Nachrichten miteinander kommunizieren.

Regelsätze bestehen aus einem Namen, der zur Identifikation dient, einer Menge von Variablen, einer Menge von Regeln und einer Menge von Regelsätzen. Die Definition ist also rekursiv; ihre genaue Syntax und Semantik wird in [Bus92] genauer definiert. Variablen sind einfache (Lisp-) Symbole, die einen beliebigen Wert annehmen können. Lediglich ihre Zuweisung muß durch den Interpreter vorgenommen werden. Eine Regel besteht aus einer Bedingung und einer Folge von Befehlen. Die Bedingung und jeder Befehl sind Lisp-Ausdrücke, in denen die Variablen referenziert werden können. Dies ermöglicht es, die Berechnung von Werten auf das unterliegende Lisp-System [3] abzuwälzen und dadurch eine möglichst mächtige Ausdruckskraft der Regelsätze zu erreichen. Lediglich die Kontrolle des Programmflusses wird durch den

[3] SEMAW ist in Common-Lisp implementiert

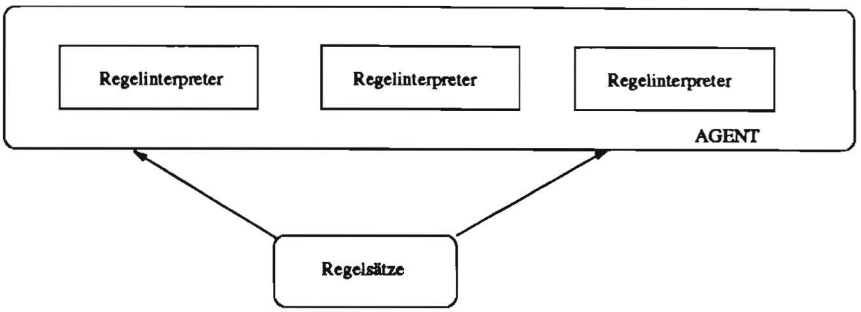


Abbildung 5: Agentenmodell

Interpreter explizit durchgeführt. Der Interpreter arbeitet in Zyklen. Am Anfang eines jeden Zyklus bestimmt er die Menge der Regeln, deren Bedingungsteile zu wahr (in Common-Lisp zum Symbol t) ausgewertet werden. Die zu den feuernden Regeln gehörenden Befehle werden in der Reihenfolge, wie sie aufgeschrieben wurden, zu einer Liste verkettet und ausgeführt. Dabei kann der Interpreter nach jeder Ausführung eines Befehls unterbrochen werden; insbesondere also inmitten der Abarbeitung einer Regel. Sind alle Befehle ausgeführt, beginnt der Zyklus von vorne. Feuern keine Regeln mehr, stoppt der Interpreter und die Komponente wird inaktiv. Der Interpreter einer Komponente ist genau dann aktiv, wenn er Befehle ausführen soll. Er wird aktiv gesetzt, wenn die Komponente mit einem Regelsatz geladen wird, oder eine Nachricht erhält. Kann der Interpreter keine feuernde Regel mehr finden, also keinen Befehl ausführen, so wird er inaktiv. Die Kontrolle, wann welche Komponente Rechenzeit erhält, kann in drei verschiedenen Modi durchgeführt werden: **sequential, agent-parallel oder component-parallel.** Sequential bedeutet, daß das System einen Agenten sucht, der (mindestens) eine aktive Komponente (also eine Komponente, die rechnen will) hat, und den Interpreter dieser Komponente unbegrenzt arbeiten läßt, bis dieser von selbst stoppt. Danach wird eine aktive Komponente im selben Agenten gesucht, dann ein neuer Agent, der eine aktive Komponente hat, bis keine mehr vorhanden sind, und das System stoppt. Bei den parallelen Modi kann eine Komponente nur solange Befehle ausführen, bis eine vom Benutzer festgelegte Grenze erreicht wird. Der Interpreter muß die Kontrolle abgeben, und sein Zähler wird zu null gesetzt. Der Vorteil paralleler Modi ist, daß man dadurch Parallelität simulieren kann, die gröber als die Ausführung eines Befehls ist. Eine Komponente im parallelen Modus muß damit rechnen, nach jedem Befehl unterbrochen zu werden. Agent-parallel bewirkt, daß ein Agent die Kontrolle behält, bis keine Komponente mehr aktiv ist. Die Komponenten des Agenten rechnen parallel, die Menge der Agenten aber sequentiell. Im Modus component-parallel rechnen alle Komponenten (sämtlicher Agenten) parallel.

Das Szenario kann gestartet werden, indem einmal die Interpreter direkt angestoßen werden. In diesem Fall sucht das System eine aktive Komponente (im entsprechenden Modus) bis keine mehr gefunden werden, und das System hält. Oder eine Benutzer-definierte Prozedur wird aufgerufen, die den Kontrollfluß übernimmt und die Interpreter selbst anstößt. Während eines Szenario-Laufs können Agenten und ihre Kommunikationsbemühungen überwacht werden, indem monitor-flags gesetzt werden. Einmal kann bestimmt werden von welchen Komponenten (welcher Agenten) send-Befehle ausgegeben werden; also wann eine Komponente an wen eine Nachricht geschickt hat. Desweiteren kann der aktuelle Regelsatz, in dem sich der Interpreter gerade befindet, ausgegeben werden. Diese Information kann durch die Ausgabe der ausgeführten Befehle und deren Argumente erweitert werden.

3.3.3 Simulation des Szenarios

Der Ansatz, der für dieses System gewählt wurde, sollte keine Einschränkungen bezüglich des zu simulierenden Szenarios beinhalten. Aus diesem Grunde wird das Szenario durch beliebige Lisp- Funktionen simuliert, ohne jegliche explizite Information über die Diskurs-Welt zu repräsentieren. Das System erfordert eine eindeutige Schnittstelle der Funktionen, und nur über die darf zwischen Agenten und Szenario kommuniziert werden. Diese Maßnahme soll sicherstellen, daß die Interaktion der Agenten mit dem Szenario vollständig überwacht werden kann, und keine "heimlichen"Veränderungen sowohl des Szenarios, als auch der Agenten durchgeführt wird. Die Funktionen haben also nur die Schnittstelle (die später genau definiert wird) zur Verfügung zu stellen; inwieweit diese realisiert wird, ist ohne Belang (für das System). Die Simulationsfunktionen müssen folgende Funktionalitäten bereitstellen:

1. Konfigurieren des Szenarios: Die notwendigen Festlegungen müssen getroffen werden, z.B. wieviele Agenten vorhanden sind - dies ist eine Größe, die oft variiert. Diese Funktion muß auch die Liste erstellen, anhand der das System erkennen kann, wieviele Agenten es erzeugen muß.

2. Rücksetzen des Szenarios: Da ein laufendes Szenario in seinen Startzustand zurücksetzbar sein muß.

3. Ausgeben von Bildschirm-Information: Dies dient zur Darstellung des Szenario-Zustandes.

4. Kommunikationsschnittstelle: Eine Funktion muß die Nachrichten der Agenten an die Welt verarbeiten, so z.B. Nachrichten weiterleiten, die an andere Agenten gerichtet sind. Dabei kann es nur Nachrichten an den Agenten an sich schicken, nicht aber an einzelne Komponente, weil sie diese nicht ßiehtünd nicht ansprechen kann. Das bedeutet insbesondere, daß auch Agenten andere Agenten höchstens als ganze Agenten sehen, sofern sie diese Agenten im Szenario überhaupt als solche erkennen.

3.3.4 Kommunikation zwischen Agenten und Szenario

Da die einzelnen Komponenten eines Agenten nicht die Möglichkeit haben, auf gemeinsame Datenstrukturen zuzugreifen, benötigen sie einen Mechanismus, durch den sie miteinander kommunizieren können. Dies wird durch das Nachrichten-Konzept bewerkstelligt. Eine Komponente ist in der Lage jeder anderen Komponente desselben Agenten, sofern sie den Namen dieser Komponente weiß, eine Nachricht zu schicken. Vom System aus gesehen handelt es sich lediglich um eine Liste von Lisp-Werten, die der adressierten Komponente in die (vordefinierte) Variable message als Listenelement eingefügt wird. Technisch läuft es so ab, daß der Nachrichteninhalt in die Variable der Komponente geschrieben wird, der Interpreter dieser aktiv wird (falls er dies noch nicht ist), und die sendende Komponente weiterrechnet. Die adressierte Komponente verarbeitet die Nachricht also erst, wenn sie wieder Rechenzeit erhält.

Derselbe Mechanismus wird verwendet, um mit dem Szenario zu kommunizieren. Eine Komponente schickt eine Liste von Lisp-Werten an die Welt (world), welche von einer Funktion des Szenarios (send-world) verarbeitet wird, und erhält die Antwort in der (vordefinierten) Variable send abgelegt. Die Antwort, die möglicherweise nil sein kann, wird direkt erzeugt, bevor die Komponente weiterrechnet. Man sollte aber in dem Fall der Kommunikation zwischen Welt und Agent, den Begriff der Nachricht nicht überstrapazieren. Agenten müssen zwar Nachrichten benutzen, wenn sie mit anderen Agenten kommunizieren wollen - insofern ist der Ausdruck Nachricht wohl gerechtfertigt, sie benutzen aber Nachrichten auch, wenn sie Aktionen oder Sinneswahrnehmung in der Welt durchführen, also z.B. sich in der Welt bewegen oder Informationen

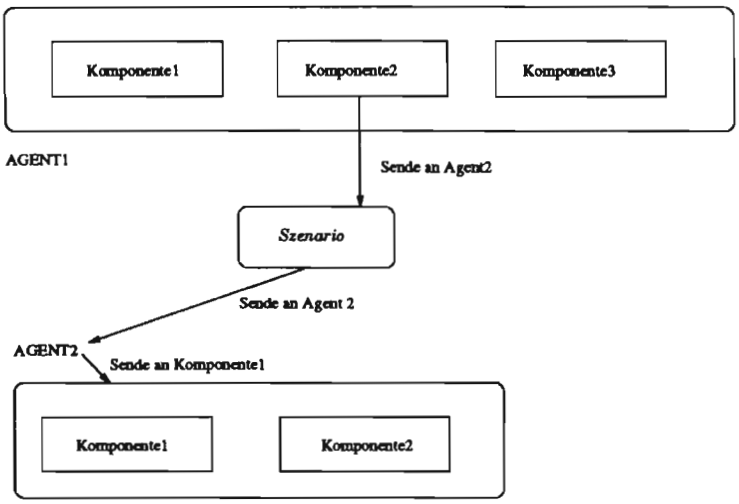


Abbildung 6: Kommunikation zwischen Agenten

durch Sehen sammeln. Der zweite Punkt ist eher ein Handeln (in der Umwelt), soll aber auch über die wohldefinierte Schnittstelle zum Szenario laufen, und daher überwachbar sein. Aus Gründen der Allgemeinheit wurde hier keine Unterscheidung getroffen.

Literatur

[Bus92] S. Bussmann. Simulation Environment for Multi-Agent Worlds. Technical Report DFKI Document D-92-01, DFKI, Saarbrücken, January 1992.

4 Anwendungen

4.1 MARS: Ein Multiagentensystem zur Simulation Kooperierender Transportunternehmen

von J. P. Müller and M. Pischel

Im folgenden Beitrag wird das System MARS, ein Multiagentensystem zur Simulation Kooperierender Transportunternehmen beschrieben. Dabei wird eine Gesellschaft von Speditionen modelliert, in der die Speditionsunternehmen versuchen, mit den ihnen zur Verfügung stehenden Transportmitteln die von Kunden vorgegebenen Transportaufträge möglichst kostengünstig auszuführen. Um dieses Ziel erreichen zu können, müssen die Speditionen - in der Simulation ebenso wie im wirklichen Leben - kooperieren.

In diesem Text skizzieren wir die wichtigsten Kooperationsformen, die Architektur des MARS Systems, und die Verwendung von DAI-Techniken wie dem Contract Net und Möglichkeiten der *Verhandlungsführung bei der Modellierung des Szenarios.*

4.1.1 Einleitung

Seit geraumer Zeit ist in der DAI-Gemeinde eine lebhafte Diskussion darüber zu beobachten, welche Eigenschaften Probleme geeignet für eine Modellierung als Multiagentensystem erscheinen lassen (siehe [FKMM92] als Überblick). Als wesentliches Kriterium dabei wird dabei die **inhärente Verteilung** des Problems angesehen. Auf das von uns zur Modellierung gewählte Speditionsszenario trifft dies in vollem Maße zu. Es gibt verschiedene Transportunternehmen, die über lokale Transportmittel verfügen und für sich gesehen autonom sind. Eine zentrale Lösung des Transportproblems durch eine übergeordnete Entscheidungsinstanz würde zum einen der Natur des Problems in keiner Weise entsprechen. Die strukturellen Gegebenheiten für eine zentrale Planung und Entscheidungskompetenz sind in der Praxis (zumindest momentan) nicht gegeben.

Zum anderen macht die mathematische Komplexität des Transportproblems eine optimale zentrale Lösung unmöglich. Eine verteilte Problemlösung lässt durch Ausnutzung parallel zu lösender Teilprobleme ein Effizenzgewinn erwarten, der sich zum einen in der schnelleren Erarbeitung von Problemlösungen wie auch in der einfacheren Modellierung der einzelnen Problemlösekomponenten (Agenten) widerspiegelt.

Zusätzlich zu den bisher genannten Kriterien gibt es eine Reihe volkswirtschaftlicher und ökologischer Erwägungen, die dafür sprechen, Techniken aus der Verteilten KI zur Modellierung der Speditionsdomäne zu verwenden. Man denke dabei an die immensen Kosten, die der Volkswirtschaft durch das permanent ansteigende Verkehrsaufkommen und die dadurch verursachte ständig steigende Umweltverschmutzung erwachsen. Durch die zunehmende Transparenz und das stetige Zusammenwachsen der internationalen Märkte[4] ist das Transportgewerbe mit einem immer härter werdenden Konkurrenzdruck konfrontiert, so daß es sich keine Spedition auf Dauer leisten kann, daß ca. 40% ihrer LKWs leer auf der Straße unterwegs sind, entweder, um neue Fracht abzuholen oder auf dem Rückweg zu ihrem Heimatort- eine Zahl, die von aktuellen Statistiken belegt wird [Rit91].

In diesem Text wollen wir das im AKA-Mod Projekt am DFKI entwickelte MARS System zur Modellierung autonomer kooperierender Speditionsgesellschaften beschreiben. In Kapitel

[4] Ab 1993 ist die Einführung eines europäische Binnenmarkt geplant.

4.1.2 geben wir einen Überblick über die Systemarchitektur, in Kapitel 4.1.3 stellen wir die verwendeten Kooperationsstrategien vor.

4.1.2 Das MARS Transportszenario

Die einzelnen Agenten des Transportszenarios verkörpern Transportunternehmen, die über lokale Resourcen verfügen, und die eingebettet sind in eine Umgebung, deren Infrastruktur durch eine Landkarte mit Verladestationen bzw. Städten, Transportwegen mit ihren Entfernungen und Zeiten gegeben sind. Diese Umgebung ist statisch, alles andere hingegen als dynamisch anzusehen, d.h. die Anzahl der Agenten, die Menge ihrer lokalen Ressourcen (Transportvehikel u.a.) und die zu bearbeitenden Transportaufträge können sich im Verlauf der Simulation ändern. Es gibt verschiedene Kommunikationskanäle und Verfahren, derer sich die Agenten bedienen können, um Möglichkeiten der Zusammenarbeit auszunutzen. Der Benutzer seinerseits kann die Simulation über eine graphische Schnittstelle überwachen und interaktiv steuern.

Die Architektur von MARS

Im folgenden werden der Aufbau des Systems, die verwendeten Kooperationsmuster sowie die Schnittstellen zum Benutzer im Mittelpunkt unserer Betrachtungen stehen. Bild 7 zeigt die Architektur von MARS. Darauf sind verschiedene Komponenten zu erkennen. Es wird dabei unterschieden in benutzerdefinierte *Agenten*, *Benutzerschnittstellen* und der *Simulationsumgebung* (World Environment). Betrachtet man das System eher vom Aspekt der Agentenmodellierung, kann man eine Gliederung nach Transportunternehmen (TGs), Transportvehikeln (TVs), Maklern und dem Weltagenten mit angegliederter Wissensbasis (World DB) vornehmen.

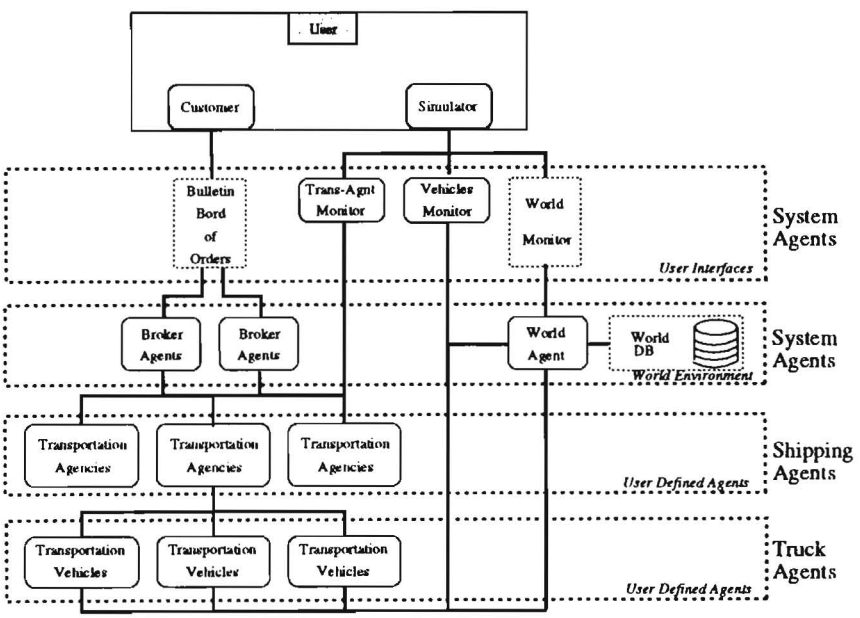


Abbildung 7: Die Architektur des MARS-Systems

Agenten sind autonome, intelligente Einheiten, die physikalisch verteilt über eigene Prozessoren realisiert werden können. Bei Speditionsagenten besteht das globale Wissen im wesentlichen aus der Kenntnis der Weltkarte mit Städten und Transportwegen, der Kenntnis von der Existenz eines Benutzers, eines Makleragenten und einer Auftragsbörse. Sie besitzen daneben auch

lokales Wissen über ihre eigenen Problemlösungsstrategien (wie werden LKWs beladen? Wie Wege geplant?), ihre Ressourcen wie speditionseigenene Transportmittel (LKWs), Modelle von anderen Speditionen und eventuell auch vom Benutzer (d.h. von möglichen Kunden) etc.

Die Agenten können untereinander *kommunizieren* und sich bestimmte Informationen aus der Welt beschaffen. Daneben müssen gewisse *Aktionen*, die Veränderungen im Environment hervorrufen, wie die Durchführung von Fahrten durch LKWs, der Welt bekanntgegeben werden. Die Welt soll in diesem Fall die Simulation übernehmen und eine Rückmeldung über den erfolgreichen oder erfolglosen Abschluß der entsprechenden Aktion an den jeweiligen Agenten abgeben.

In der Endversion soll der Benutzer mittels der Simulationsumgebung individuell eigene Agenten aus den vorgegebenen Klassen nach seinen jeweiligen Wünschen so leicht und sicher wie möglich generieren können. Gewisse Agenten sind aber von vorneherein existent und gehören zum internen Kern des Systems, wie z.B. ein Weltagent, ein Makleragent etc. Geplant ist, daß der Benutzer selbst die eigentlichen Aufträge als Eingabe an das System gibt. Im folgenden wollen wir kurz die Komponenten des Systems erläutern.

Die **Transportgesellschaften** (TGs) sind Spezialisten bei der Verhandlungsführung mit Benutzer, Makleragenten und anderen Speditionsagenten. Sie beherrschen verschiedene Kommunikations- und Kooperationsmechanismen für unterschiedliche Agentenklassen. Weiterhin planen sie die Auftragsabarbeitung auf der Speditionsebene. Daneben fällt ihnen die Entscheidungskompetenz über die Verteilung auf ihre Ressourcen (z.B. LKWs) zu. Da Ressourcen ebenfalls als Agenten modelliert sind, müssen die Speditionen über Möglichkeiten zur Kommunikation mit den selben verfügen, und in der Lage sein, auch deren Ausführungsverhalten nach gewissen Kriterien zu koordinieren.

Ebenso wie die TGs sind die von ihnen eingesetzten **Transportvehikel** (TVs) als eine eigene Klasse autonomer Agenten modelliert. Die TVs sind Spezialisten für Wegeplanung und Ladungszusammenstellung. Gerade die Aufgabe der Wegeplanung kann durchaus als losgelöst vom Rest der Aufgaben modelliert werden, die eine Spedition zu bewältigen hat. Die TVs haben ein höheres Maß an Autonomie und sind im Fall der Rücktransportauslastung durch Partnerspeditionen auch in der Lage, dezentral zu entscheiden, ob sie Rückfrage bei ihrer Speditionen nehmen sollen oder nicht. Die echte Parallelität der Prozesse bringt hierbei einen zusätzlichen Effizienzgewinn. Weiterhin haben die TVs in der Simulation die Aufgabe, neben der Planung ebenfalls die zeitabhängige Ausführung jeder einzelnen Fahrt von einem Ort der Region zum anderen durchzuführen.

Neben den benutzerdefinierbaren Agenten existieren außerdem eine Reihe systeminterne Agenten. Die erste Gruppe sind die **Makleragenten**. Der Anspruch hierbei war, dem Benutzer ein Hilfsmittel zur Verfügung zu stellen, daß ihm bei der Auftragsvergabe an die Speditionen unterstützend unter die Arme greifen kann. Zwar soll der Benutzer prinzipiell die Auftragsvergabe bis ins kleinste Detail steuern können, aber diese Möglichkeit wird er wahrscheinlich eher nur selten in Anspruch nehmen wollen, weshalb die Vergabe von Aufträgen an Speditionsagenten von einem Makleragenten im Sinne des Benutzers durchgeführt wird.

Zu den Aufgaben des **Weltagenten** gehört u.a. die Repräsentation der Aktivitäten der Agenten sowie deren Auswirkungen auf die Welt. Außerdem sollten globale Daten über den aktuellen Ort von Gütern oder eine synchronisierte Zeit verfügbar sein. Uber diesen Weltagenten und seine angegliederte Datenbank **World DB** kann in einem **World Monitor** eine Visualisierung der Abarbeitung von Aufträgen durch Transportmittel erreicht werden, bei der die Transportvehikel als Objekte über eine Landkarte bewegt und vom Benutzer visuell verfolgt werden können. Außerdem soll sich der Benutzer durch Anklicken von Objekten mit der Maus zusätzliche Informationen über sie beschaffen können.

Die aktuelle Version des MARS Simulationssystems ist in der auf OPS-5 basierenden Programmiersprache MAGSY [FW92] auf einer UNIX SUN Workstation - Umgebung implementiert. MAGSY bietet Hilfsmittel zur Definition von Basisagenten und zur Interprozess-Kommunikation. Die Erweiterung des Systems auf andere (z.B. PROLOG oder LISP) Agenten ist geplant.

4.1.3 Kooperationsformen im MARS–Szenario

In diesem Kapitel wollen wir die für das Speditionsszenario relevanten Kooperationsformen beschreiben. Nach einem allgemeinen Überblick wollen wir auf zwei Kooperationsformen genauer eingehen. Im zweiten Teil des Kapitels wollen wir die wesentlichen konzeptuellen Aspekte der Implementierung des MARS-Szenariosschildern.

Warum kooperieren Speditionen?

Die im Speditionsszenario auftretenden Agenten, die Speditionen, sind in eine marktwirtschaftliche Umgebung eingebettet und verhalten sich diesem Umstand gemäß. Ihr erstes Ziel ist es, ihren Profit zu maximieren. Dieses Ziel kann durch Erhöhung der Einnahmen oder durch Senkung der Kosten erreicht werden. Da eine Erhöhung der Einnahmen aufgrund des strengen Wettbewerbs nur in sehr begrenztem Maße möglich ist, müssen sich die Speditionen im wesentlichen auf die Verringerung der Kosten begrenzen. In der Tat bietet sich hier ein guter Angriffspunkt für Kooperation. Dies umfaßt sowohl interne Kooperationsstrukturen (*vertikale Kooperation*, s.u.) als auch Kooperation mit anderen Speditionen (*horizontale Kooperation*). Vor dem Hintergrund der eingangs angeführten Probleme lässt sich feststellen, daß eine Spedition ihre Kosten erheblich senken kann, wenn sie ihre Leerfahrten anderen Speditionen anbietet. Auch in der Praxis kann man wachsende Kooperations- und Vereinheitlichungsbemühungen auf dem Speditionsmarkt erkennen. Die auftretenden Kooperationsformen werden im nächsten Abschnitt detaillierter betrachtet.

Kooperationsformen zwischen Speditionen

In der Speditionsdomäne lassen sich eine Anzahl typischer Kooperationsformen identifizieren, wobei wir zwischen vertikalen und horizontalen Kooperationsformen unterscheiden. Vertikale Kooperationsformen umfassen insbesondere die Koordination einer Spedition mit ihren LKWs. Hier sind u.a. zu nennen:

- Austausch von Informationen und Beantworten von Anfragen (Frage-Antwort-Schema).
- Kooperative Zerlegung und Verteilung von Aufgaben (*task decomposition*, *task allocation*).
- Der LKW meldet Fahrten mit schwacher Auslastung an seine Spedition, unterstützt damit also die unten beschriebene horizontale Kooperation.

Da wir LKWs nicht einfach als Ressourcen, sondern insbesondere als eigenständige Agenten auffassen, scheint uns der Begriff der vertikalen Kooperation für die zwischen Spedition und LKWs ablaufenden Interaktionen sinnvoll.

Die horizontale Kooperation ist dadurch charakterisiert, daß es sich bei den Kooperationspartner grundsätzlich um autonome Agenten handelt, die in keinem hierarchischen Abhängigkeitsverhältnis zueinander stehen. In unserem Szenario könnte man darunter die Kooperation zwischen dem Kunden und einer Spedition, Speditionen sowie LKWs untereinander, oder

zwischen einer Spedition und einem LKW einer anderen Spedition verstehen. Momentan beschränken wir uns dabei auf die Kooperation zwischen Speditionen (und zweitrangig auf die Interaktionen zwischen Kunde und Spedition). Gemeinsame Aktionen können in der Regel nur unter Einverständnis aller Beteiligten zustande kommen. Hierbei spielt der Begriff der *Verhandlung* als das zentrale Mittel zum Erzielen von Übereinkünften zwischen autonomen Agenten eine zentrale Rolle (siehe auch [ZR89, S. 92]). Die wichtigsten horizontalen Kooperationsmuster sind:

- Der Austausch von Informationen und die Beantwortung von Anfragen zwischen Speditionen (z.B. Frage-Antwort-Schema).
- Vermeidung von Leerfahrten durch Anbieten an andere Speditionen, oder durch Weitergabe eines Teilauftrags.
- Aushandeln der Bedingungen für die Ausführung eines Auftrags zwischen Kunde und Spedition.
- Rückführungs-Kooperation: anstatt die Rückfahrt eines LKWs zum Heimatort selbst zu planen, kann die Spedition den LKW einer anderen Spedition übergeben, die die Rückladung für den LKW plant. Dabei kann sie ihn unter Umständen auch für eigene Zwecke mit verwenden (siehe auch unten).
- Wir unterscheiden beim Güterverkehr zwischen Langstrecken- und Regionalverkehr. Die Verbindung zwischen beiden Typen kann in kooperativer Weise geschehen, indem sich manche Speditionen auf Langstrekken-, andere auf Kurzstreckenverkehr spezialisieren (siehe auch unten).

Die letzten beiden Kooperationstypen wollen wir etwas detaillierter betrachten:

Rückladung von LKWs

Das Problem, für LKWs nach einer Fahrt vom Haupthaus der Spedition in eine andere Stadt eine Rückladung zu finden, d.h. möglichst eine Leerfahrt von dieser Stadt zum Haupthaus zu vermeiden, konnte bis vor einigen Jahren nur unter erheblichem Aufwand seitens der Disponenten gelöst werden, da die Rückladung zentral geplant wurde. Durch Einführung des Linienverkehrs, d.h. des Verkehrs zwischen dem Haupthaus A und der Partnerspedition B, konnte die Aufgabe der Rückführung auf die Partnerspedition verlagert werden.

In Gebieten ohne festen Kooperationspartner wird irgendeine bekannte Spedition kontaktiert. Ein dynamischer Aufbau temporärer Partnerbeziehungen ist also auch möglich.

Nahverkehrsdisposition

Eine weitere Form der Kooperation ergibt sich im Bereich der Kopplung von Fern- und Nahverkehr. *Fährt ein LKW der Firma X&Y nach Hamburg, um im Raum Hamburg mehrere* kleinere Aufträge verschiedener Kunden auf einmal zu erledigen (Sammelverkehr), so fährt er in der Regel nicht jeden einzelnen Auftraggeber an, sondern die Distribution im Nahbereich wird wiederum vom Partner P&Q in Hamburg besorgt. Kommt der LKW in Hamburg an, so wird die Ware umgeladen und im Nahverkehr zugestellt. Die Routenplanung für den Fernverkehr wird dadurch erheblich vereinfacht, denn der LKW, der von Saarbrücken nach Hamburg fährt, muß nur die Route von X&Y zu P&Q planen. Zudem kann durch die Verteilung der Güter von

einem großen auf mehrere kleinere LKWs die Zustellung der Güter schneller und auch günstiger erfolgen.

Die oben beschriebenen Kooperationsformen sollen im folgenden durch einige Beispiele näher erläutert werden.

Beispiel 1: Ein LKW der Spedition S (in Saarbrücken) fährt mit Stückgutladung von Saarbrükken nach Frankfurt. Nachdem er in Frankfurt beim Kunden K entladen hat, fährt er zur Partnerspedition P. P besorgt ihm Sammelladung, anschließend fährt er mit Sammelladung zurück nach Saarbrücken, wo die Teilladungen von der Nahverkehrsabteilung übernommen werden (Abb. 8).

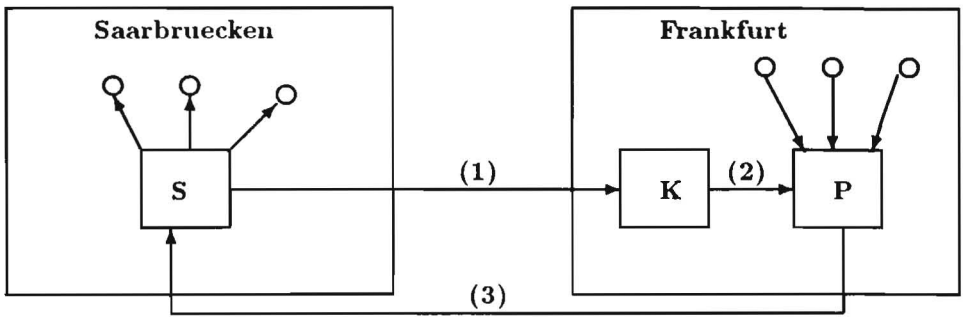


Abbildung 8: Beispiel 1

Beispiel 2: Ein LKW der Spedition S fährt mit Sammelladung von Saarbrücken nach Neuß. Die Partnerspedition X (in Neuß) übernimmt die Verteilung der Sammelladung. Der leere LKW bekommt neue Sammelladung vom Partner X. Er fährt mit der Ladung nach Hannover, Partner Y in Hannover übernimmt wiederum die Verteilung der Sammelladung. Y besorgt ihm schließlich Rückladung und er kann zurück nach Saarbrücken fahren (Abb. 9).

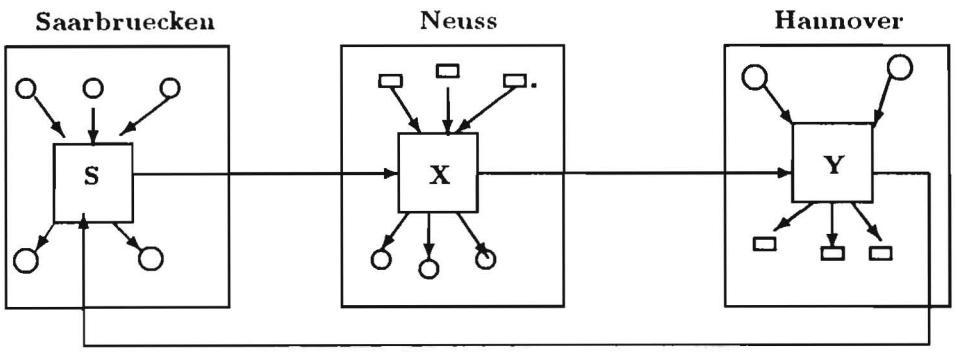


Abbildung 9: Beispiel 2

Beispiel 3: Dieses Beispiel ist eine Mischform von Beispiel 1 und Beispiel 2. Spedition S (in Saarbrücken) hat Güter (X), die nach Dortmund zu befördern sind. Spedition P in Dortmund hat Waren (Y), die nach Trier transportiert werden müssen. Beide Speditionen schicken jeweils

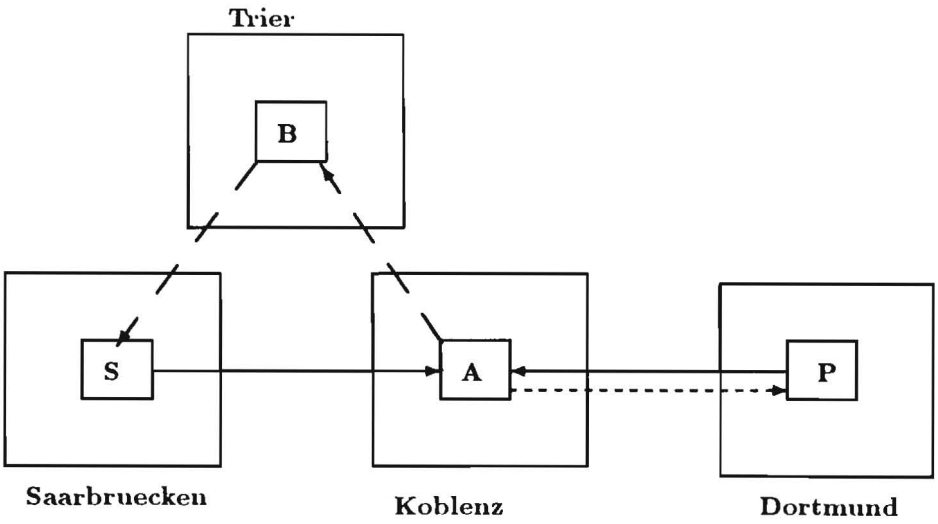


Abbildung 10: Beispiel 3

einen LKW (aus Saarbrücken LKW Nr. 1, aus Dortmund LKW Nr. 2) nach Koblenz. In Koblenz werden die Waren umgeladen. LKW Nr. 2 fährt mit Waren X zurück nach Dortmund. LKW Nr. 1 fährt zunächst mit Waren Y nach Trier. Dort wird er entladen und bekommt von der Partnerspedition B eine neue Ladung Z, die er dann anschließend nach Saarbrücken transportiert (Abb. 10).

Im nächsten Abschnitt werden wir die Implementierung von Kooperationsformen im MARS System näher beschreiben.

Kooperations- und Kommunikationstypen in MARS

In diesem Abschnitt wollen wir eine Beschreibung der in MARS implementierten Kooperationstypen geben [BKMP92, MPS92]. Dabei wollen wir uns auf die zugrundeliegenden Kooperationsprotokolle konzentrieren.

Aufgabenzerlegung und Aufgabenverteilung

Die zentrale Aufgabe einer Spedition ist es, vom Kunden erhaltene Aufträge zu bearbeiten. Dazu müssen Aufträge in Teilaufträge zerlegt und die Teilaufträge auf geeignete LKWs verteilt werden. Dies wird in MARS über das Contract Net (siehe [Smi80, DS83]) realisiert. Hierbei handelt es sich um eine typische Form vertikaler Kooperation, die durch die hierarchische Beziehung zwischen einer Spedition und ihren LKWs gegeben ist. Für die Realisierung ergeben sich zwei Alternativen, die in Bild 11 dargestellt sind. Alternative a) zeigt ein verschachteltes Contract Net, in dem die Speditionen vom Makler erhaltene Aufträge zunächst an ihre LKWs weitergeben, die selbst wiederum Angebote abgeben. Erst wenn die Spedition so einen möglichen Plan (und eine damit verbunden Kostenabschätzung) gefunden hat, bewirbt sie sich bei dem Makler. Bekommt sie den Auftrag, gibt sie ihren LKWs "grünes Licht", und die Ausführung kann beginnen.

Ein Nachteil dieser Methode ist die sich durch die Verflechtung der beiden Contract Nets ergebenden Wartezeiten. Der LKW muß einen Plan für einen Auftrag erstellen und pflegen, bevor

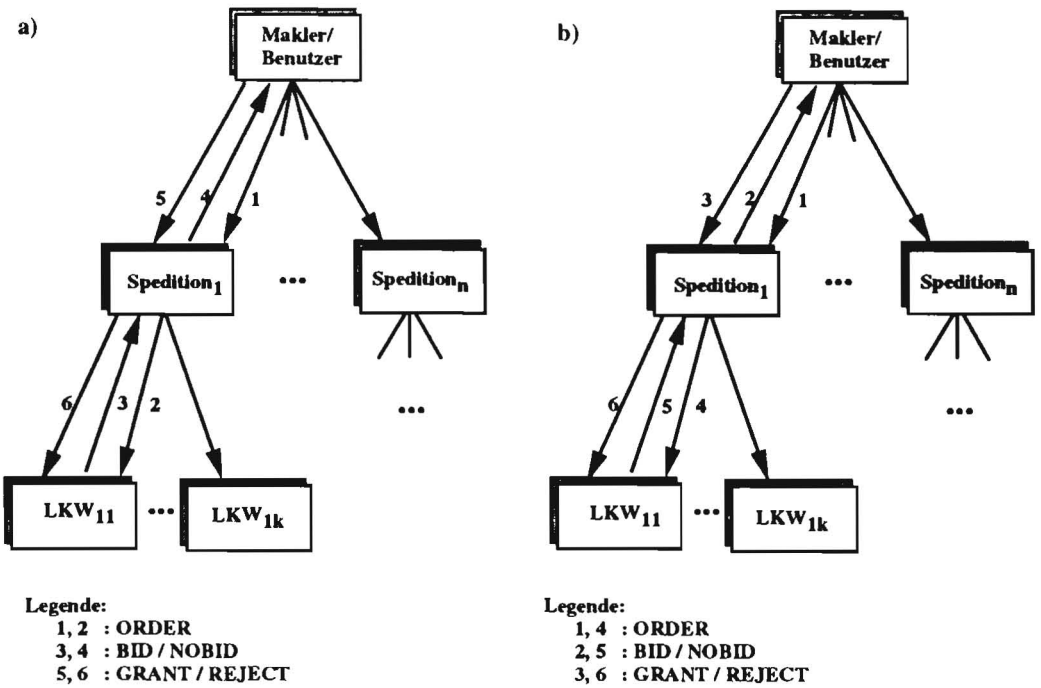


Abbildung 11: Zerlegung und Verteilung von Aufgaben durch das Contract Net

der Auftrag sicher ist. Dies kann unter Umständen zu langen bedingten Plänen und zu unzuverlässigen Kostenabschätzungen führen. Alternative b) zeigt deshalb ein entkoppeltes Contract Net. Die Spedition schätzt selbst (über Heuristiken oder Frachttabellen) die voraussichtlichen Kosten des Auftrags ab, ohne in dieser Phase bereits ihre LKWs einzuschalten. Erst wenn der Makler ihr den Auftrag zusichert, verteilt sie den Auftrag auf ihre LKWs.

Horizontale Kooperation: Vermeidung von Leerfahrten

Nun möchten wir unsere Implementierung der oben bereits beschriebenen Leerfahrtenkooperation zwischen Speditionen beschreiben. Die zugrundeliegende Idee dabei ist, daß LKWs ihren Speditionen unrentable Fahrten (d.h. Fahrten mit zu geringer Auslastung) melden. Die Spedition wählt basierend auf ihrem Wissen über andere Speditionen mögliche Interessenten aus, denen sie die "Mitfahrgelegenheit" anbietet. Nun muß sich die Spedition mit Interessenten über die genauen Bedingungen der Kooperation einigen.

Zwei Alternativen für diesen Einigungsprozeß sind in Bild 12 illustriert. Alternative a) stellt eine Basislösung dar. Hierbei kann sich die Spedition nur um die Leerfahrt bewerben (wenn sie daran interessiert ist), es erfolgt aber keine echte Verhandlung über die Rahmenbedingungen wie z.B. Preis, Zeitpunkt oder Interesse nur an einer Teilstrecke der ursprünglich angebotenen Leerfahrt. Alternative b) zeigt einen echten Verhandlungsprozeß, bei dem durch Vorschläge und Gegenvorschläge die unterschiedlichen Positionen sich einander annähern, bis eine Übereinkunft erzielt ist.

Die Spedition wählt durch den oben beschriebenen Verhandlungsprozeß einen geeigneten Interessenten aus (dies kann unter Umständen auch sie selbst sein), und teilt ihrem LKW mit, daß eine Spedition gefunden wurde, die die Mitfahrgelegenheit wahrnehmen möchte. Da es möglich ist, daß z.B. durch lokale Planoptimierungen des LKW eine zuvor angebotene Leerfahrt

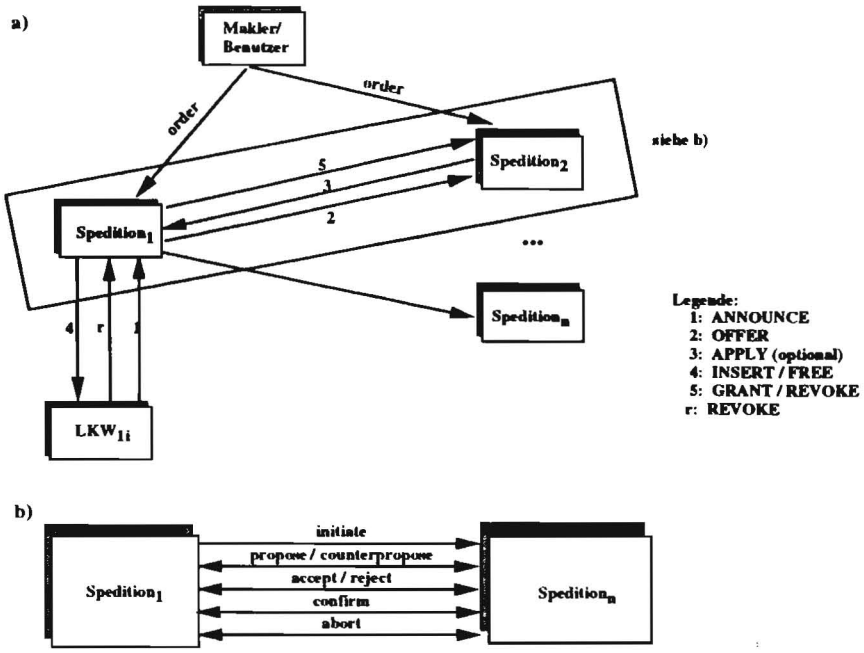


Abbildung 12: Horizontale Kooperation: Anbieten von Leerfahrten

ganz wegfällt, muß der LKW die Möglichkeit haben, eine Leerfahrtenankündigung an seine Spedition zu widerrufen (REVOKE). Die Spedition muß diese Information an die Speditionen weitergeben, denen sie die Leerfahrt bereits angeboten hat.

Die Leerfahrtenkooperation umfaßt sowohl horizontale als auch vertikale Kooperation, denn, wie aus Bild 12 ersichtlich, ist die Spedition auf die Mitarbeit ihrer LKWs angewiesen, die Schwachstellen in ihren lokalen Plänen erkennen und diese ihrer Spedition mitteilen.

Literatur

[BKMP92] M. Buchheit, N. Kuhn, J. P. Müller, and M. Pischel. Mars: Modelling a multiagent scenario for shipping companies. In *Proceedings of ESS92, Dresden.* Forthcoming, 1992.

[DS83] R. Davis and R.G. Smith. Negotiation as a metaphor for distributed problem solving. In *Artificial Intelligence, 20(1)*, pages 63–109, 1983.

[FKMM92] K. Fischer, N. Kuhn, H. J. Müller, and J. P. Müller. Sophisticated and distributed: The transportation domain. Submitted for CAIA-93, 1992.

[FW92] K. Fischer and H. M. Windisch. Magsy- ein regelbasiertes Multi-agentensystem. In *KI 1/92.* 1992.

[MPS92] J. P. Müller, M. Pischel, and A. Schroth. MARS: Eine Multiagenten-Testumgebung mit Graphischer Benutzerschnittstelle. Internal Working Paper, 1992.

[Rit91] R. Rittmann. Die Macht der Trucks. *Bild der Wissenschaft*, 9:112–114, 1991.

[BM92] S. Bussmann and H.J. Müller. A Negotiation Framework for Cooperating Agents. Proc. of the 2nd Workshop on Cooperating Knowledge Based Systems, September 1992.

[Smi80] R.G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. In *IEEE Transaction on Computers*, volume C-29, pages 1104–1113, 1980.

[ZR89] G. Zlotkin and J. S. Rosenschein. Negotiation and task sharing among autonomous agents in cooperative domains. In *Proc. of the Eleventh IJCAI*, pages 912–917. Detroit, Michigan, August 1989.

4.2 Anwendungen unter MAGSY

von K. Fischer

4.2.1 Verladehof

Beim 'Verladehof'-Beispiel, wird ein Lastwagen von einer Menge von Gabelstaplern mit Kisten beladen, wobei die Kisten zunächst in einem Lager untergebracht sind. Das Anwendungsbeispiel 'Verladehof' war das erste Anwendungsbeispiel, das mit Hilfe der UNIX-Version von MAGSY implementiert wurde. Es sollte zunächst vor allem die prinzipiellen Möglichkeiten von MAGSY exemplarisch aufzeigen. Aus diesem Grund ist die Modellierung der einzelnen Agenten sehr grob.

Abbildung 13 zeigt eine mögliche Situation bei der Problemlösung. In dem Anwendungsbeispiel sind bisher nur die Gabelstapler als MAGSY-Agenten modelliert, die durch lokale Strategievorgaben die ihnen gestellten Aufgaben zu lösen. Die Aufgaben die von den Gabelstaplern gelöst werden müssen werden dabei aus der Beschreibung der Zielsituation abgeleitet. Die Zielsituation beschreibt dabei, wo die einzelnen Kisten stehen müssen, damit die Gesamtaufgabe gelöst ist. Sowohl der Lastwagen, als auch das Lager sind in der aktuellen Implementierung als passive Einheiten modelliert.

Damit die Ausführbarkeit der von den Gabelstaplern ausgewählten Aktionen überwacht werden kann, wurde ein Weltagent eingeführt, der die reale Welt simuliert. Wenn sich also ein Gabelstapler entschließt eine bestimmte Aktion auszuführen (z.B. sich von einem Platz auf einen anderen zu bewegen), dann entscheidet die Welt, ob diese Aktion erfolgreich ausgeführt werden kann. In einer Multiagentenwelt kann man nicht davon ausgehen, daß das Wissen, das die einzelnen Agenten haben global konsistent ist. Damit ist es möglich, daß einzelne Agenten auf Grund von Wissen, das mit in der simulierten Welt gültigen Fakten inkonsistent ist, falsche Entscheidungen treffen.

Abbildung 14 zeigt die prinzipielle Struktur der Applikation, wobei davon ausgegangen wird, daß die anstehenden Aufgaben von drei Gabelstaplern gelöst werden. Die Anzahl der Gabelstapler ist nur durch die im Szenario verfügbaren Plätze begrenzt. 'MAGSY_WB' ist die zentrale Instanz, die das globale Wissen verwaltet und verteilt. So lernen alle Ageten, die die Gabelstapler repräsentieren den Weltagenten über die MAGSY_WB kennen. In der MAGSY_WB ist auch globales Wissen über das Szenario abgelegt. So kann hier Geometrieinformation über die Größe des Verladehofes abgefragt werden, oder auch, wieviele Kisten im Szenario vorhanden sind, wo sie beim Systemstart stehen und wo sie in der Zielsituation stehen müssen.

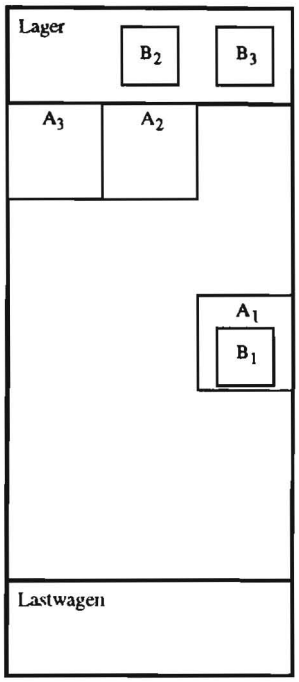


Abbildung 13: Momentaufnahme beim Verladehofbeispiel.

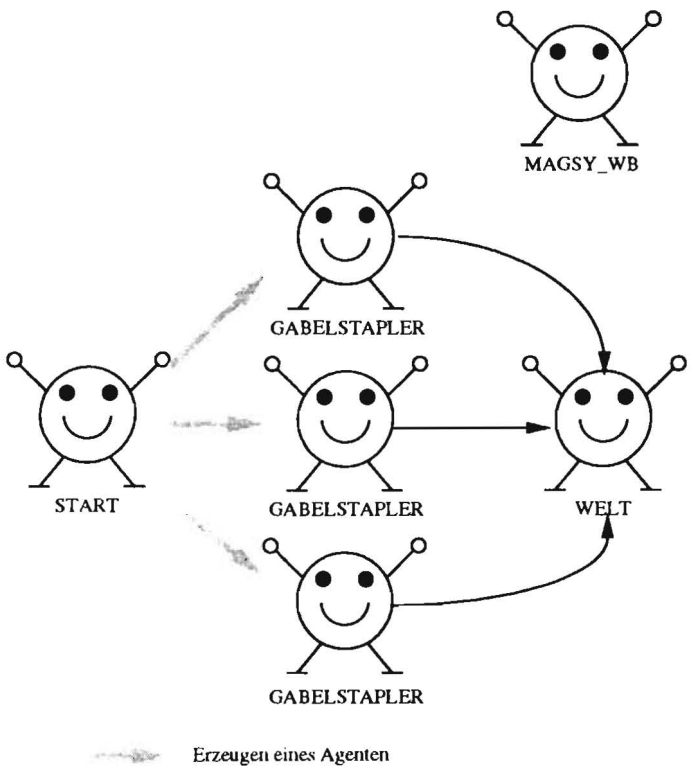


Abbildung 14: Struktur der MAGSY-Anwendung für das Verladehofbeispiel.

Aus der Differenz zwischen aktueller Situation, die sich dynamisch aus der Startsituation entwickelt, und Zielsituation leiten die Agenten, die die Gabelstapler repräsentieren, das Ziel ab, das sie als naechstes zu erreichen versuchen. Da die Einzelziele, eine Kiste vom Lager auf den Lastwagen zu transportieren, nicht interagieren und da die Gabelstapler alle Ziele alleine für sich selbst erreichen können, kann die Problemlösung durch eine vollständig lokale Planung in den Gabelstaplern erfolgen. Eine Interaktion zwischen den Gabelstaplern findet nur statt, wenn von den Gabelstaplern Ressourcen in der Welt (Kisten, Plätze etc.) beansprucht werden.

4.2.2 Spedition

Das Anwendungsbeispiel 'Spedition' zielt auf ein ähnliches Anwendungsgebiet wie das Verladehofbeispiel. Zunächst werden diese beiden Anwendungsbeispiele getrennt untersucht, es ist aber möglich, daß sie in Zukunft zusammenwachsen.

Beim Anwendungsbeispiel 'Spedition' geht es darum, eine Menge von Speditionen zu modellieren, die jeweils eine Menge von Lastwagen (Lkw) zur Verfügung haben und die versuchen müssen nach lokalen Kriterien optimal zu wirtschaften, indem sie sich um Kundenaufträge bewerben und diese, sofern sie den Zuschlag bekommen, möglichst optimal auf ihre Lkw verteilen. Bei der Abarbeitung der Kundenaufträge sollen die Speditionen kooperieren, indem sie für sie selbst ungünstige Situationen dadurch zu lösen versuchen, indem sie Teilaufgaben an andere Speditionen delegieren. Dabei sollen sie sowohl Teilaufträge an andere Speditionen weitergeben können, als auch Leerfahrten von Lkw anderen Speditionen anbieten können.

Die derzeitige Implementierung sieht neben dem Agenten 'magsy_wb', der das globale Faktenwissen der Multiagentenanwendung verwaltet, einen Makleragenten vor, der die vom Benutzer vorgegebenen Aufträge verwaltet. Zusätzlich wird für jede Spedition ein eigener Agent eingeführt, wobei auch die einer Spedition zugeordneten Lkw durch jeweils einen eigenen Agenten modelliert werden (siehe Abbildung 15).

Wird vom Benutzer ein Auftrag in das System eingegeben, so wird dieser an den Makleragenten weitergegeben. Der Makleragent gibt seinerseits alle eintreffenden Aufträge an die Speditionen weiter, mit der Aufforderung, ein Gebot abzugeben, zu welchem Preis und bis zu welchem Zeitpunkt der Auftrag von der jeweiligen Spedition abgearbeitet werden kann. Die Speditionsagenten haben keine Informationen über den aktuellen Zustand ihrer Lkw. Aus diesem Grund wird ein Auftrag, der bei einer Spedition eintrifft, von dieser Spedition an alle dieser Spedition zugeordneten Lkw weitergeleitet, wo bei nun die Lkw aufgefordert werden, ein Gebot abzugeben, zu welchem Preis und bis zu welchem Zeitpunkt sie einen bestimmten Auftrag bearbeiten können.

In den Lkw liegt zu jedem Zeitpunkt ein möglicherweise leerer Plan vor, der vom Lkw abgearbeitet wird. Wird ein Lkw aufgefordert ein Gebot für einen bestimmten Auftrag abzugeben, so versucht er diesen Auftrag an eine für den Lkw nach lokalen Gesichtspunkten optimale Stelle in den bereits vorliegenden Plan einzubauen. Ein Lkw braucht dabei nicht immer für die Gesamtmenge des Auftrags ein Gebot abzugeben, sondern kann in seinem Gebot angeben welche Menge des Auftrags zu den angegebenen Konditionen transportiert werden kann. In die Bewertung, zu welchen Konditionen ein Auftrag von einem Lkw erledigt werden kann, gehen die Länge der Fahrstrecke und die Kosten für den belegten Laderaum ein.

Die Speditionen vergleicht die Angebote der Lkw und wählt das beste Gebot aus. Wird in dem Gebot nicht die im Auftrag geforderte Gesamtmenge abgedeckt, so wird diese Gesamtmenge um die im ausgewählten Gebot angebene Fördermenge verringert und der Auftrag danach erneut an alle Lkw übergeben, mit der Aufforderung ein Gebot für den so entstendenen Auftrag

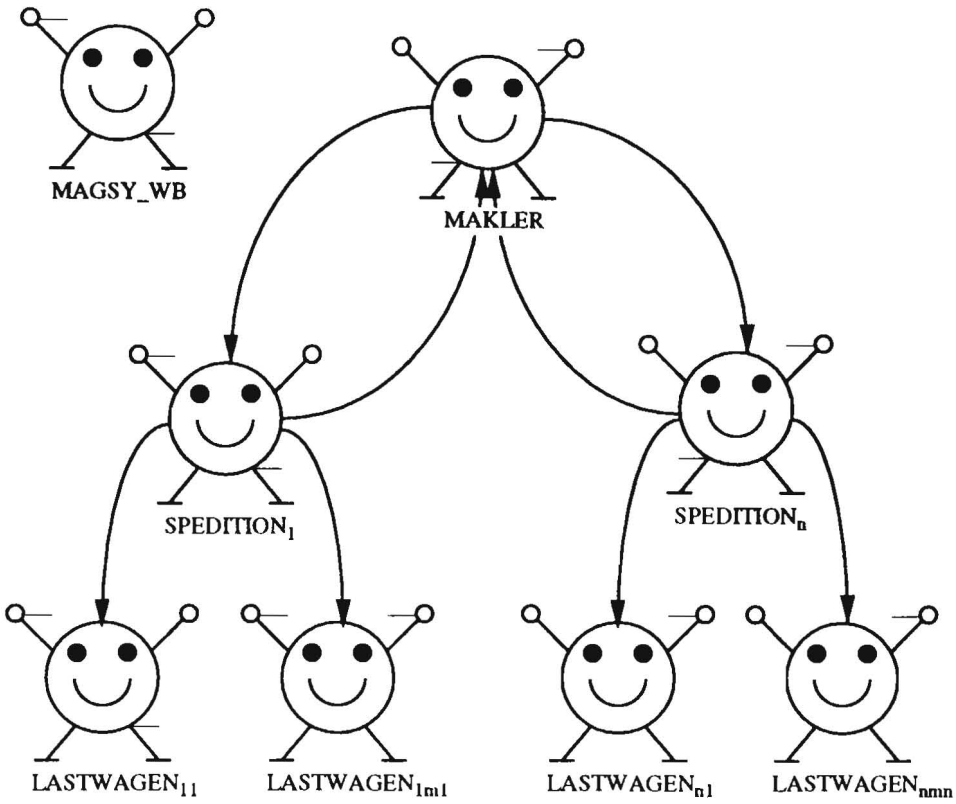


Abbildung 15: Struktur der MAGSY-Anwendung für das Speditionsbeispiel.

abzugeben. Auf diese Weise läßt sich im Speditionsagenten ein Gesamtgebot errechnen, das sich aus einem oder mehreren Teilgeboten von Lkw zusammensetzt. Dieses Gesamtgebot wird von dem Speditionsagenten an den Makleragenten übermittelt. Der Makleragent sammelt die Gebote aller Speditionen und wählt das konstengünstigste Gebot aus. Die Spedition, die das günstigste Gebot abgeben hat erhält also den Zuschlag.

An dieser Stelle Trittt ein wichtiger effekt ein. Bis zu dem Zeitpunkt, bis zu dem der Makleragent einer bestimmten Spedition den Zuschlag für die Durchführung eines Auftrags gegeben hat, sind die von den Lkw gemachten Planungsentscheidungen temporär. Nur in den Lkw der Spedtion, die den Zuschlag für den Auftrag erhält, kann diese Planung beibehalten werden. In allen anderen Lkw der anderen Speditionen müssen diese Planungsentscheidungen rückgängig gemacht werden.

4.2.3 Flexible Fertigung

Das komplexeste Anwendungsbeispiel, das bisher mit MAGSY implementiert wurde, wurde im Rahmen einer Diplomarbeit [Jelinek 91] mit Hilfe des unter VAX/VMS implementierten Prototypen von MAGSY implementiert. Derzeit wird dieses Anwendungsbeispiel auf die UNIX-Implementierung von MAGSY portiert. Bei dem Anwendungsbeispiel handelt es sich um die Implementierung einer Planungshierarchie für die Steuerung einer flexiblen Fertigung. Der Aufbau der flexiblen Fertigung war dabei durch ein 140x120 cm^2 großes Modell einer Fertigungsanlage gegeben, das aus Fischer-Technik-Baugruppen aufgebaut ist (Abb. 16). Die prinzipielle Funktionalität der einzelnen Komponenten in diesem Modell soll im folgenden kurz erläutert werden.

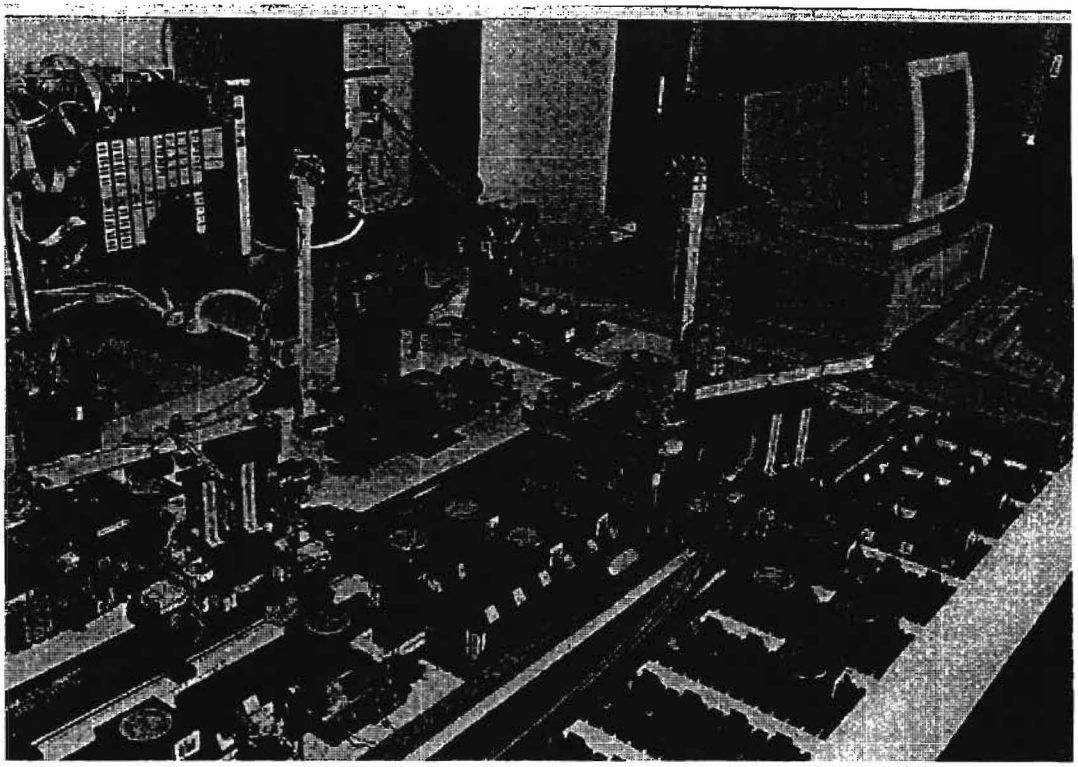


Abbildung 16: Bild des fischertechnik-Fabrikmodells.

Zwei mobile Roboter, die jeweils 4 Achsen besitzen, führen die Handhabungs- und Transportaufgaben aus. Das Hauptlager kann nur von Roboter_1, die drei Maschinen auf der anderen Seite können nur von Roboter_2 bedient werden. Im Bereich des Eingabeförderbandes, des Zwischenlagers, der Heizzelle und des Ausgabeförderbandes überlappt der Arbeitsraum der beiden Roboter. Deshalb ist es notwendig, den Robotern Strategien für eine Kollisionsvermeidung vorzugeben. Bei der Beschickung der Maschinen und beim Beladen der Heizzelle müssen die Bewegungen der Roboter mit den Aktionen der Maschinen koordiniert werden.

Die Werkstücke gelangen über das Eingabeförderband, auf dem ein Strichcodeleser zur Werkstückidentifizierung angebracht ist, in den Bearbeitungsprozeß. Anhand von induktiven Sensoren (Initiatoren), wie sie auch auf den Werkstückplätzen der Maschinen und beim Ausgabeförderband angebracht sind, kann die Anwesenheit eines Werkstücks festgestellt werden. Das Hauptlager dient als Aufnahmeplatz für Teile, die gerade nicht bearbeitet werden können oder auf eine Auslieferung warten. Das Zwischenlager wird als Übergabebereich zwischen den beiden Robotern verwendet.

Die Heizzelle in der Mitte der Fabrik kann von beiden Robotern beschickt werden. Ihr Innenraum ist durch ein Dach abgeschlossen, das geöffnet sein muß, wenn ein Werkstück eingelegt oder entnommen werden soll. Bei der Bearbeitung wird durch einen Regelungsprozeß mit Hilfe von Heizdrähten und Ventilatoren ein vorgegebenes Temperaturprofil eingehalten. Die Fräsmaschine kann von zwei Seiten her beladen werden. Die Werkstücke werden mit Hilfe von Förderbändern von links und rechts auf den Bearbeitungsplatz geschoben und von dort wieder auf den Eingabeplatz entladen. Bei der Bearbeitung folgt der Fräskopf einer durch ein Programm vorgegebenen Bahn. Auch die Revolverdrehmaschine kann von zwei Seiten her beladen werden. Je nach Stellung des Bearbeitungstisches ist dabei für den Roboter immer nur ein Werkstückplatz auf dem Bearbeitungstisch zugänglich. Bei der Bearbeitung kann eine Sequenz aus verschiedenen Werkzeugen und zugehörigen Bearbeitungszeiten vorgegeben werden. Der Meßplatz kann nur von vorn beladen werden. Für den Roboter ist immer nur der Werkstückplatz zugänglich, der

sich nicht unter dem Meßkopf befindet. Auf zu große Abweichungen vom vorgegebenen Sollwert muß die Fertigungssteuerung entsprechend reagieren, z.B. durch die Wiederholung eines Bearbeitungsvorgangs.

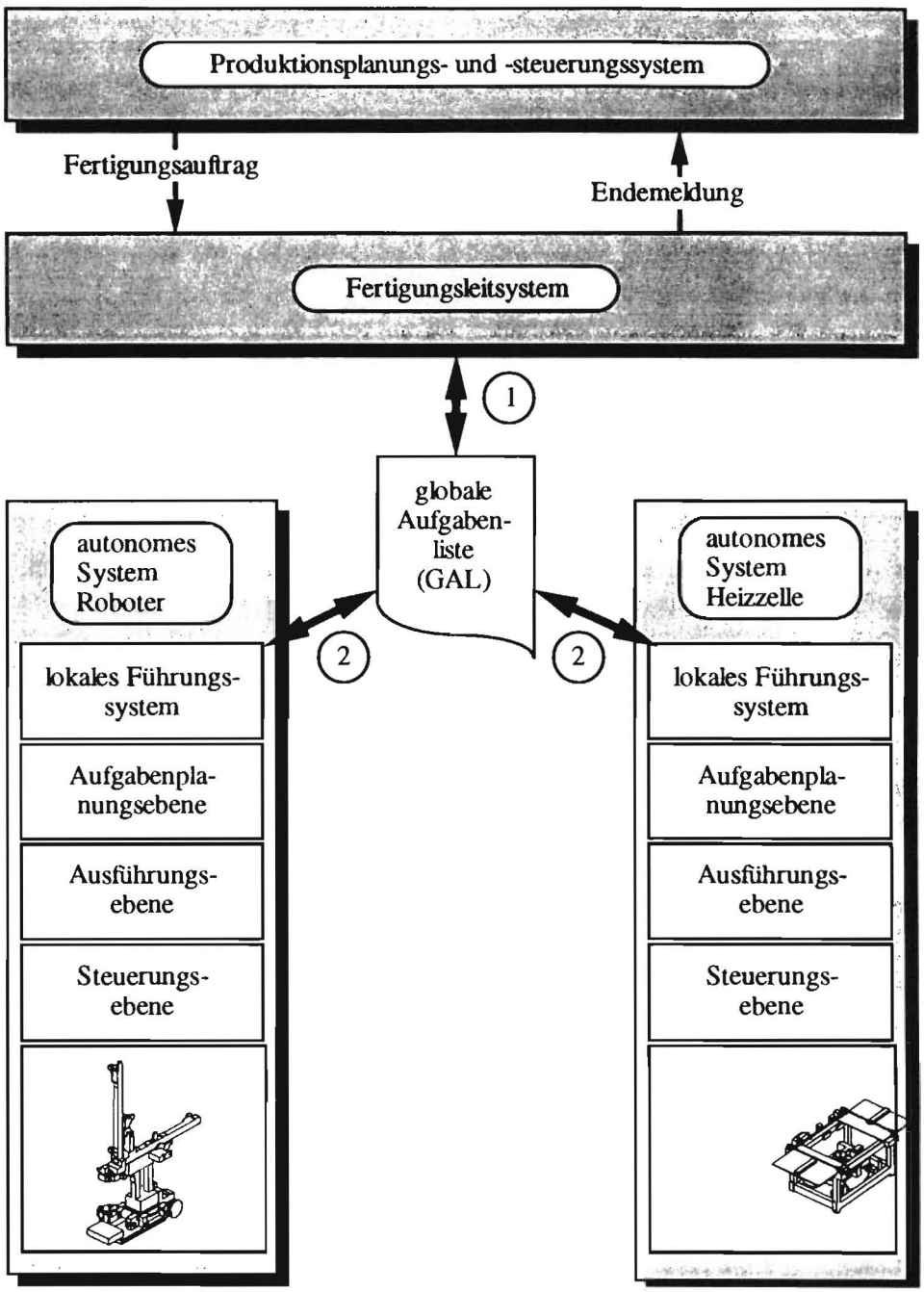
Die Planungshierarchie (siehe Abbildung 17 die für die Steuerung der flexiblen Fertigung für dieses Fabrikmodell entworfen wurde, wurde bereits in [Fischer 91, Fischer 92] veröffentlicht. Sie daher hier nur kurz zusammengefaßt werden.

Die zentrale Planungsinstanz in der Planungshierarchie ist das Produktionsplanungs- und -steuerungssystem. Hier werden Kundenaufträge erfaßt. Auf die Erfassung eines Kundenauftrages erfolgt ein grober Kapazitätsabgleich und eine Zeitrechnung die es erlaubt, eine Prognose bzgl. des voraussichtlichen Fertigstellungstermins zu machen. Falls das notwendige Rohteil nicht im Lager vorhanden ist, wird eines entsprechende Bestellung ausgegeben. Wurde ein Kundenauftrag angenommen, so wird er, sobald es die Auslastung der Fertigungsanlage zuläßt, als Fertigungsauftrag an das Fertigungsleitsystem eingelastet.

Die eigentliche Fertigung eines Produktes wird durch das Fertigungsleitsystem geplant und überwacht. Im Fertigungsleitsystem liegen zu einem Zeitpunkt eine Menge von Fertigungsaufträgen vor. Für jeden Fertigungsauftrag ist durch einen Arbeitsplan vorgegeben, welche Bearbeitungsschritte auf welcher Maschine an dem zugehörigen Werkstück ausgeführt werden müssen. Aus den Arbeitsplänen der Fertigungsaufträgen werden Aufgaben abgeleitet, die von den autonom agierenden Systemen (den Robotern und den Maschinen), gelöst werden müssen, damit die Fertigung der Endprodukte erreicht wird. Beim Ableiten dieser Einzelaufgaben wird das sheduling-Problem gelöst, d.h. die Frage welcher Auftrag als nächstes auf einer bestimmten Maschine bearbeitet werden soll. Bei diesem sheduling-Problem wird aufgrund der Komplexität des Problems (es handelt sich hier um ein NP-schweres Problem) nicht versucht einen optimalen Maschinenbelegungsplan zu berechnen. Vielmehr wurde das Fertigungsleitsystem als ein reaktives System implementiert, daß unter Verwendung der aktuellen Daten durch Heuristiken (dead-line sheduling) entscheidet, welcher Fertigungsauftrag als nächstes auf eine bestimmte Maschine eingelastet wird.

Die vom Fertigungsleitsystem aus den Arbeitsplänen abgeleiteten Aufgaben werden der Ebene der lokalen Führungssysteme in einer Aufgabenliste bekanntgegeben. Diese Aufgaben müssen im allgemeinen von mehreren autonomen Systemen in Kooperation gelöst werden. Dazu ist es notwendig, daß sich für jede Aufgabe eine Gruppe von autonomen Systmen findet, die zusammen in der Lage ist, diese Aufgabe zu lösen. Eine solche Gruppe von autonomen Systemen soll *Team* genannt werden. Die lokalen Führungssysteme der einzelnen autonomen Systeme bestimmen, welchem Team sich ein autonomes System als nächstes anschließt, hier erfolgt also eine Koordination bzlg. der Aufgaben zwischen den autonomen Systemen. Hat sich für eine bestimmte Aufgabe ein Team gefunden, so muß jedes der im Team enthaltenen autonomen Systeme ein entsprechendes Verhalten zeigen, damit die Gesamtaufgabe gelöst werden kann. Dazu wird vom lokalen Führungssystem eines autonomen Systems aus der vom Fertigungsleitsystem bekanntgegebenen Gesamtaufgabe entsprechend der Rolle, die das autonome System bei dieser Gesamtaufgabe übernimmt eine Einzelaufgabe abgeleitet, die das autonome Syste lösen muß, damit die Gesamtaufgabe gelöst wird.

Die vom lokalen Führungssystem eines autonomen Systems abgeleitete Einzelaufgabe wird an die Aufgabenplanungsebene des autonomen Systems weitergeleitet. Zur Lösung dieser Einzelaufgabe muß auf der Aufgabenplanungsebene ein Verhaltensmuster (das ist eine Menge von Regeln, die auf eine bestimmte Art strukturiert sind) gefunden werden, das das Verhalten beschreibt, das von dem autonomen System gezeigt, werden muß, damit es die Einzelaufgabe lösen kann. Ist dieses Verhaltensmuster im lokalen Wissen des Aufgabenplanungsagenten nicht bekannt, so wird es in einer globalen Wissensbasis gesucht. Wird das Verhaltensmuster auch in



(1) Bekanntgabe der Aufträge und Abfrage von Endemeldungen.

(2) Koordinationsprotokoll bzgl. der Aufgaben.

Abbildung 17: Strukturdefinition einer flexiblen Fertigungsteuerung

der globalen Wissensbasis nicht gefunden, so kann das autonome System die Einzelaufgabe nicht lösen. Bei der Ausführung eines Verhaltensmusters werden primitive Aktionen abgeleitet, die an die Ausführungsebene weitergeleitet werden, wo sie tatsächlich zur Ausführung kommen. In dem die Verhaltensmuster in den einzelnen autonomen Systemen Schrittweise ausgeführt werden, wird die Lösung der Gesamtaufgabe vorangetrieben.

Änderungen in der Umwelt werden dem Aufgabenplanungsagenten durch asynchron beim Aufgabenplanungsagenten eingehendes Wissen mitgeteilt. Dieses asynchron eingehende Wissen treibt den Planungsvorgang im Aufgabenplanungsagenten voran. Der Aufgabenplanungsagent soll dabei jederzeit auf externe Ereignisse reagieren können. Aus diesem Grund wird die Ausführung und die Überwachung einzelner primitiver Aktionen in die Ausführungsebne verlagert, wo sie parallel zur Ausführung des Aufgabenplanungsagenten geschieht. Primitive Aktionen unterscheiden sich in dem Sinne von den Einzelaufgaben die vom lokalen Führungssystem an den Aufgabenplanungsagenten weitergegeben werden, als bei der Ausführung einer primitiven Aktion keine Kooperation von mehreren autonomen Systemen erforderlich ist, d.h. die gesamte Synchronisation die zwischen zwei autonomen Systemen erforderlich ist erfolgt auf der Aufgabenplanungsebene.

Literatur

[Fischer 91] Fischer, Klaus: Ein Agentensystem für eine flexible Fertigungssteuerung, Prozeßrechensysteme '91, Springer-Verlag, Berlin, Februar 1991, S. 140–149.

[Fischer 92] Klaus Fischer, Verteiltes und kooperatives Planen in einer flexiblen Fertigungsumgebung, Doktorarbeit, Institut für Informatik, TU München, Juli, 1992.

[Jelinek 91] Elisabeth Jelinek, Implementierung einer flexiblen Fertigungssteuerung am Beispiel eines Fischer-Technik-Modells mit Hilfe von KI-Werkzeugen, Diplomarbeit, TU München, Institut für Informatik, November, 1991.

4.3 COSMA: Ein verteilter Terminplaner als Fallstudie der Verteilten KI

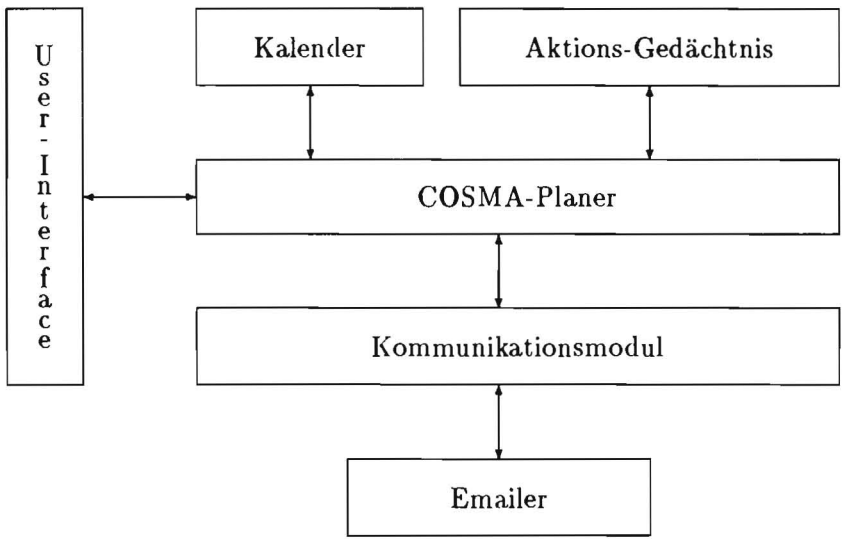
von A. Schupeta

Eine der wichtigsten Fragen in dem noch sehr jungen und inhomogenen Gebiet der verteilten KI betrifft die Koordination der Aktivitäten der verschiedenen Agenten: Wie lassen sich die einzelnen Aktionen der Agenten so koordinieren, daß ein konsistentes Gesamtverhalten des Systems entsteht? In vielen klassischen Gebieten der Informatik (z.B. bei Betriebssystemen) geht es bei der Koordination von Prozessen im wesentlichen um die Verhinderung **negativer Interaktion** wie etwa beim Zugriff auf exklusive Datenbereiche oder bei den Mechanismen zur Deadlockverhinderung. Im Bereich der Multiagentensysteme steht daneben die **positive Interaktion** im Zentrum des Interesses: Wie können Aktionen geplant und ausgeführt werden, die das gemeinsame gleichzeitige oder zeitlich eng verzahnte Handeln von mehreren Agenten inhärent erfordern? Aus dieser Fragestellung lassen sich die meisten anderen zentralen Fragen der verteilten KI ableiten: Welche Art von Kommunikation ist notwendig und angemessen? Welche Mechanismen lösen mögliche Konflikte zwischen den geplanten Aktionen der Agenten? Welches Maß an globalem Wissen benötigen die Agenten zur Koordination? Welche Form von Organisation innerhalb der Agentengesellschaft ist nötig und kann auf welche Weise realisiert werden?

Ein verteilter Terminplaner erscheint uns als ideale Anwendung zur Modellierung einer Agentengesellschaft um einige der Fragestellungen der VKI zu untersuchen, weil:

- Terminplanung ist ein inhärent verteiltes Problem. Jeder kennt es und hat es.
- Es umfaßt alle wichtigen VKI Fragestellungen, denn eine erfolgreiche Terminplanung setzt die erfolgreiche Koordination mit den Terminen anderer Personen voraus. Dabei müssen Konflikte aufgelößt werden, es muß kommuniziert und verhandelt werden.
- Ein verteilter Terminplaner arbeitet mit einem relativ überschaubaren Weltausschnitt. Die Wissensmodellierung wirft also keine unlösbaren Probleme auf.
- Die Agenten sind nicht auf sensorische oder motorische Fähigkeiten angewiesen. Das Agentenmodell kann sehr einfach gehalten werden. Seine Kernbestandteile sind ein Planungsmodul und ein Kommunikationsmodul, die jedoch eng verzahnt miteinander arbeiten sollten.

Das folgende Bild zeigt die Grobstruktur eines unserer COSMA-Agenten (für Cooperative Schedule Management Agent):



Der Kalender dient als Datenbasis für vereinbarte Termine. Zeitintervalle, die sich aktuell in Verhandlung befinden, können abhängig von der gewählten Verhandlungsstrategie als für andere Verhandlungen gesperrt markiert sein. Im Aktions-Gedächtnis wird der aktuelle Zustand, sowie die Historie einer Verhandlung gespeichert. Der COSMA-Planer realisiert unter Zugriff auf die anderen Module die Funktionalität des Agenten. Das Kommunikationsmodul wird im Projekt DISCO, mit dem wir im Rahmen des COSMA-Projektes eng kooperieren, erstellt. Als Basissystem für die Kommunikation wird auf das Emailsystem zurückgegriffen. Zur komfortableren Handhabung für den Benutzer ist zusätlich eine window-orientierte Benutzer-Schnittstelle vorgesehen. Die Funktionalität von COSMA umfaßt folgende Punkte:

- Verwaltung des persöhnlichen Terminkalenders des Benutzers.
- Beantwortung von Anfragen über die Verfügbarkeit von Zeiten durch den Benutzer und andere COSMA-Agenten.
- Vereinbarung von Terminen und Treffen mit mehreren Teilnehmern.
- Bestätigung von Terminen.
- Absage von bereits vereinbarten Terminen.

- Verschieben von vereinbarten Terminen.
- Erinnerung an Termine.

Um einen Termin zu arrangieren beginnt COSMA Verhandlungen mit allen Teilnehmern des Treffens bzw. deren COSMAs über einen für alle passenden Zeitpunkt. Diese Verhandlungen mögen mehrere Verhandlungsrunden von Vorschlägen und Gegenvorschlägen umfassen. Einigen sich die Teilnehmer schließlich auf einen Termin, so wird dieser in die Termindatenbasis eines jeden COSMA-Agenten eingetragen. Auch bereits festgelegte Termine können erneut verschoben oder abgesagt werden. Das zentrale Objekt der Verhandlungen ist jeweils der Termin, den wir mit den folgenden Eigenschaften ausstatten:

- **Teilnehmer** Die Teilnehmer eines Treffens müssen dem COSMA-System wenigstens über ihre Email-adresse bekannt sein. Die Einführung von Gruppennamen ist möglich
- **Initiator** Die Person, die ihrem COSMA-Agenten initial den Wunsch für ein Treffen spezifiziert. Sie muß nicht notwendig in der Menge der Teilnehmer sein.
- **Typ** Treffen können von unterschiedlichem Typ sein. Die Typen führen dabei implizit verschiedene Bedingungen ein. Z.B. kann ein "Vortrag" nur in einem Vortragsraum stattfinden (Projektor, Tafel...), eine unspezifizierte Dauer eines "Kurztreffens" wird mit 15 Minuten angenommen usw.
- **Priorität** Die Priorität des Treffens kann festgelegt werden.
- **Zeit** Die Zeit eines Treffens soll möglichst grob spezifiziert sein. Wenn ich ein Treffen für morgen 15:37 wünsche, ist die Wahrscheinlichkeit groß, daß jemand nicht kommen kann. Spezifiziere ich dagegen morgen Nachmittag, so wird sich eher ein für alle Teilnehmer passender Zeitpunkt finden lassen. Die Zeit ist in unserer ersten Prototypversion der eigentliche Verhandlungsgegenstand.
- **Dauer** Die Dauer des Treffens. Ist keine Dauer spezifiziert, so wird in Abhängigkeit vom Typ des Treffens ein Defaultwert angenommen.
- **Ort** Der Ort des Treffens. Auch hier soll die Behandlung von unterspezifizierten Orten möglich sein.
- **Thema** Das Thema ist nur für den menschlichen Benutzer vorgesehen und wird nicht vom System verwendet. Es kann auch für Bemerkungen und Hinweise verwendet werden.

Die Verhandlungsführung steht im Mittelpunkt unseres Interesses. Sie wird sicherlich einer Art Methode oder Protokoll folgen, Ziel ist aber, eine möglichst offene flexible Verhandlung zu modellieren. Die für unsere Domäne plausibel erscheinenden **Verhandlungsmethoden** unterscheiden sich in den Strategien zur Ankündigung eines Termins durch den Initiator, in den Antworten der angesprochenen Teilnehmer darauf, in der Art und Weise wie noch in Verhandlung befindliche Termine in andere Verhandlungen behandelt werden und in der Rolle, die der Initiator im Verlauf der Verhandlung einnimmt:

- **Ankündigungsstrategie**:
  - Der Initiator wählt den ihm am besten passenden Zeitpunkt.
  - Der Initiator wählt einen möglichst großen Zeitraum.
- **Antwortstrategie**:

- Der Teilnehmer sendet eine ja-oder-nein Antwort
- Der Teilnehmer sendet in seiner Antwort, falls er zu der spezifizierten Zeiten nicht kann, alternative Zeitvorschläge.

- **Strategie für gleichzeitige Verhandlungen:**
  - Alle noch in Verhandlung befindlichen Zeitabschnitte werden in anderen Verhandlungen wie belegte Zeiten behandelt, solange keine endgültige Zeit festgelegt ist.
  - Alle noch in Verhandlung befindlichen Zeitabschnitte werden in anderen Verhandlungen wie freie Zeiten behandelt. Eine endgültige Festlegung in einer der Verhandlungen führt dann zu einer veränderten Situation in der anderen Verhandlung.

- **Strategie des Initiators:**
  - Dem Initiator obliegt während der gesamten Verhandlung die Rolle der Verhandlungsführung. Er ist verantwortlich für die Fixierung des Termin am Ende der Verhandlung.
  - Der Initiator ist gleichberechtigter Verhandlungspartner. Ein Termin gilt als fixiert, wenn alle Teilnehmer einem exakten Zeitpunkt zugestimmt haben.

Bei der letzten Strategie treten natürlich weitere unbeantwortete Fragen auf, wie z.B.: "Woher weiß der einzelne Agent, wann alle anderen Agenten zugestimmt haben?" Eine Bestätigung von jedem einzelnen Agenten an alle Anderen wäre ineffizient. Die implizite Annahme der Zustimmung durch alle, falls man eine "gewisse" Zeit lang keinen Gegenvorschlag erhält wäre eine andere Alternative. Dies erscheint auch natürlich, denn wenn mich jemand fragt, ob mir ein Gruppentreffen morgen um 3 passt, und ich sage "ja" und höre nichts weiter über das Treffen, so gehe ich implizit davon aus, daß es stattfindet.

Genau diese Fragen der Interaktion wollen wir im COSMA-Projekt untersuchen. Die oben beschriebenen Verhandlungsstrategien sollen miteinander kombiniert und getestet werden. Die Nachbildung von spezifisch menschlichen Interaktionsmechanismen ist dabei eines unserer Hauptanliegen. Ein wichtiger Schritt darür ist die Verwendung von natürlicher Sprache zur Kommunikation. Das **Kommunikationsmodul** des COSMA-Agenten wird von der Projektgruppe DISCO erstellt. Es soll in der Lage sein, auch von Menschen lesbare Nachrichten als Emails zu versenden, und umgekehrt von menschlichen Benutzern geschriebene Nachrichten, sofern sie in den Kontext einer Terminvereinbarung passen, zu verstehen. Gelingt die Integration von COSMA-Benutzern und Benutzern, die ohne COSMA nur mit ihrem Email-System arbeiten, so ist das der beste Beweis für eine natürliche Modellierung der Verhandlungsführung. Im übrigen ist die Anbindung einer natürlich sprachlichen Schnittstelle an die Agenten eines VKI-Systems ein bisher noch in keiner Weise angegangenes Problem. Wir erhoffen uns daraus Impulse sowohl für die nat. Sprachverarbeitung als auch für die Entwicklung von adäquaten Verhandlungsmechanismen [5].

Abschließend soll kurz unsere erste Prototypversion des COSMA-Planers skizziert werden: Die Schnittstelle zwischen COSMA-Planer und Kommunikationsmodul bzw. User-Interface wird durch Nachrichten eines bestimmten Formates festgelegt. Diese Nachrichten enthalten neben Sender und Empfänger der Nachricht, eine Nummer zur Identifikation, eine weitere Nummer als Referenz, falls es sich um eine Antwort auf eine andere Nachricht handelt und eine Aktionenliste. Obwohl der Planer immer nur eine Aktion ausführt, kann es sein, daß das Kommunikationsmodul

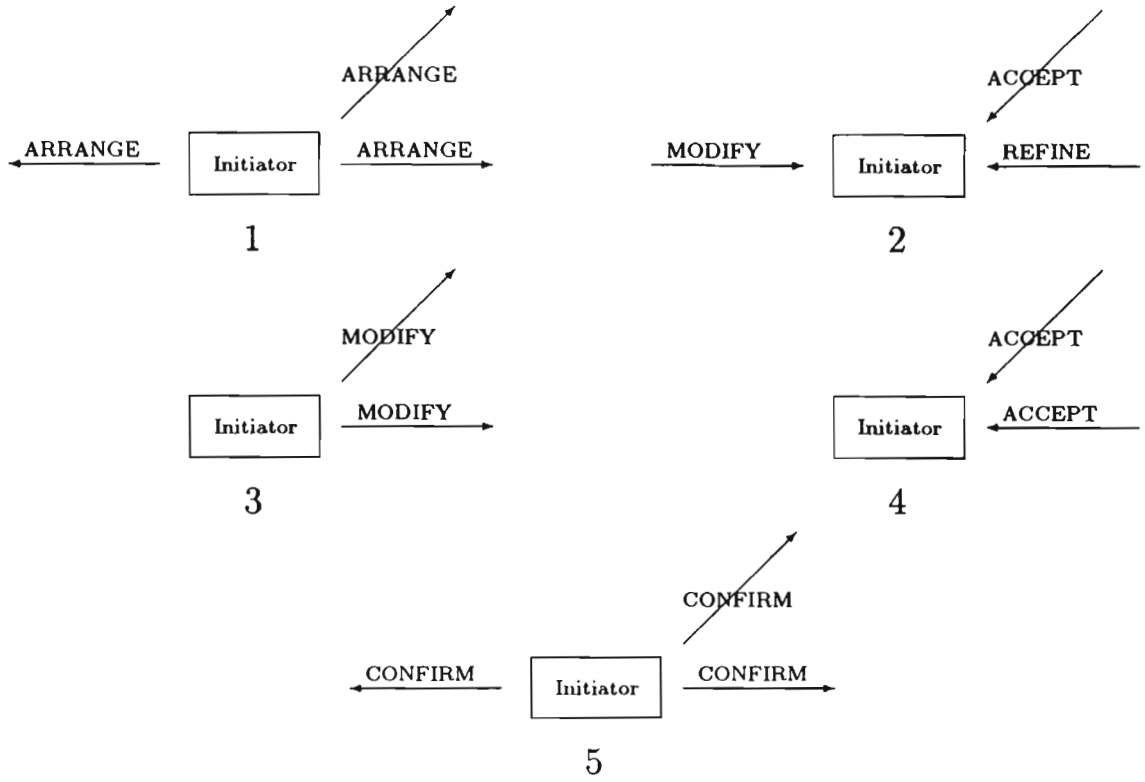
[5] Etwa wie sie in diesem Band von Steiner in Kap. 3.1 vorgeschlagen wurden und bereits so auch tendenziell in die Aktionsplanung eingegangen sind.

die intendierte Aktion nicht eindeutig erkennen kann. Der Planer muß dann mit Hilfe der Verhandlungshistorie die sinnvollste Aktion aus der Liste ausführen. Folgende Aktionen können auftreten:

- **ARRANGE**: Initierung der Verhandlung zur Terminvereinbarung.
- **REFINE**: Ein vorgeschlagener Terminzeitraum wird weiter eingeschränkt.
- **MODIFY**: Ein alternativer Zeitraum wird vorgeschlagen.
- **CHANGE**: Ein bereits bestehender Termin soll verschoben werden. Eine abgeschlossene Verhandlung muß erneut eröffnet werden.
- **ACCEPT**: Zustimmung zu einem Termin.
- **CONFIRM**: Fixierung des Termins. (Bei der momentanen Verhandlungsstrategie dient diese Aktion dem Verhandlungsführer als Abschluß der Verhandlung wenn alle Teilnehmer den Termin akzeptiert haben.)
- **REJECT**: Endgültige Ablehnung einer Anfrage nach Terminvereinbarung.
- **CANCEL**: Absage eines bereits bestehenden Termins.
- **QUERY**: Anfrage nach Daten über bestehende Termine aus dem Kalender.

Assoziiert mit den Aktionen ist jeweils eine Terminbeschreibung, welche im wesentlichen die weiter oben beschriebenen Eigenschaften des Termins enthält. Darin sind Zeiträume durch Begriffe wie z.B.: "morgen", "Mittwoch in 2 Wochen", oder "in der Woche nach der ECAI" spezifiziert. Der Planer transformiert dies unter Verwendung von Defaultweren (z.B. Nachmittag = von 12:00 bis 18:00) in konkrete Zeitintervalle. Auch Disjunktionen von Zeiten (etwa "morgen oder übermorgen") sind zugelassen. Ein einfaches Szenario stellt sich mit der aktuellen Version etwa so dar:

Einer der Benutzer beauftragt sein COSMA ein Zusammentreffen mit einer Anzahl anderer Personen zu arrangieren. Dies geschieht im Dialog mit Hilfe der graphischen Benutzeroberfläche. COSMA fragt alle Teilnehmer, ob sie zu dieser Zeit können (ARRANGE). Die Antworten werden auf die vier möglichen Fälle "ACCEPT", "REFINE", "MODIFY" und "REJECT" abgebildet. Im Falle eines "REJECT" ist der Termin nicht zu vereinbaren, und der Initiator sendet ein "REJECT" an alle Teilnehmer und das User-Interface, d.h. an den Benutzer. "MODIFY" bedeutet, das der Teilnehmer zu der vorgeschlagenen Zeit nicht kann und eine andere Zeit vorschlägt. Der Initiator versucht solche Antworten miteinander in Einklang zu bringen und die modifizierte Zeit in einer zweiten Verhandlungsrunde erneut an alle Teilnehmer zu senden, oder falls sich die Alternativvorschläge gegenseitig ausschließen, generiert er selber einen neuen Zeitvorschlag. Die "REFINE" Antworten schränken den unscharfen Zeitvorschlag auf ein Teilintervall ein. Der Initiator versucht solche Antworten auf ein allen genehmes Intervall abzubilden. Falls dies mißlingt steigt man wiederum ein in eine weitere Verhandlungsrunde mit einem neuen Termin, den der Initiator vorschlägt. Bleiben am Ende nur noch "ACCEPT" Antworten übrig, so legt der Initiator einen präzisen Zeitraum für das Treffen fest und informiert mit "CONFIRM" alle Teilnehmer, diesen in ihrem Terminkalender zu fixieren. Die folgende Zeichnung zeigt den Ablauf eines kleinen Beispiels:



4.4 Entwicklung kooperativer Anwendungen unter MEKKA

von A. Lux

4.4.1 Einführung

Der nachfolgende Beitrag skizziert alltägliche Kooperationsszenarien aus den Bereichen Terminvereinbarung und Verkehrswesen. Die ausgewählten Szenarien dienen dazu, die im Vortrag "MEKKA: Eine Umgebung zur Konstruktion kooperativer Anwendungen"vorgestellten Ansätze zu verdeutlichen, insbesondere unter den Gesichtspunkten Agenten- und Kooperationsmodell. Dabei soll vor allem der Grundgedanke, verschiedenartige kooperative Applikationen schnell entwerfen und prototypisch implementieren zu können, veranschaulicht werden. Zum besseren Verständnis dieses Artikels sei deshalb noch einmal auf [Ste91] in diesem Band verwiesen.

Betrachtet man Applikationen, bei denen mehrere Agenten in einem kooperativen Verhältnis zueinander stehen, so lassen sich die Anwendungen in Kategorien unterschiedlicher Komplexität untergliedern. Einfache Szenarien wie z.B. Hotelreservierung, bei denen nicht mehr als zwei Personen involviert sind, lassen sich leicht zu komplexeren Anwendungen wie Reiseplanung im Team bis hin zu hochkomplizierten kooperativen Szenarien wie (Luft-) Verkehrswesen oder Netzwerkmanagement mit einer Vielzahl von menschlichen und maschinellen Agenten erweitern.

Allen Anwendungen gemeinsam ist allerdings, daß sie auf einem Agentenmodell basieren müssen, das sowohl menschliche als auch maschinelle Agenten abdeckt, und das die Kooperationsbeziehungen, die zwischen einzelnen Agenten bestehen, durch ein formales, generisches Kooperationsmodell abgedeckt werden müssen. Insofern besteht das Ziel dieses Artikels darin, die Verwendbarkeit der in MEKKA gewählten Ansätze an konkreten Beispielen zu zeigen. Bei den

beiden ausgewählten Beispielen Terminvereinbarung und Verkehrswesen stehen zum gegenwärtigen Zeitpunkt aber nicht die Modellierung umfassender und vollständig arbeitender Systeme im Vordergrund, sondern die prototypische, explorative Modellierung ausgewählter Aspekte.

4.4.2 Anwendung 1: Terminvereinbarung

Bei der Vereinbarung von Terminen handelt es sich um ein inhärent verteiltes und kooperatives Vorgehen erforderndes Problem. Insofern haben sich verschiedene Forschungsrichtungen im letzten Jahrzehnt verstärkt mit dieser Thematik auseinandergesetzt, insbesondere unter dem Aspekt der Computerunterstützung. Es entstanden personenbezogene Kalendersysteme, mit denen der Benutzer computerunterstüzt seine eigenen Termine verwalten kann. Darüberhinaus entstanden erste, allerdings zentralisierte Systeme, zur Terminvereinbarung. Dank der neuesten Fortschritte in den Bereichen *Verteilte Systeme* und *Kommunikationstechnologie* sind nun die technischen Voraussetzungen gegeben, verteilte Terminvereinbarungssystme zu entwickeln. Bei dem von uns gewählten Multiagenten-Ansatz betrachten wir nicht nur die Teilnehmer eines zu vereinbarenden Treffens als Agenten[6], sondern auch Ressourcen wie Besprechungsräume, Ausstattungsgegenstände wie Overheadprojektor, Videorekorder etc.[7]

Ausgangspunkt unseres Szenarios ist eine Anzahl von geografisch verteilten Personen, die ein Treffen vereinbaren wollen. Dabei tritt eine Person als Initiator[8] eines Treffens auf, die restlichen Personen als Teilnehmer. Den einzelnen Personen sind jeweils lokal elektronische Terminkalender zugeordnet. Um nun ein Treffen zu vereinbaren, müssen eine Reihe von Parametern berücksichtigt werden. Bei unserer ersten prototypischen Implementation konzentrieren wir uns auf die folgenden wichtigen Parameter:

- Teilnehmer: hierbei kann unterschieden werden, ob ein Teilnehmer am Treffen teilnehmen muß oder aber ob auf ihn evtl. verzichtet werden kann; ebenso können Teilnehmer mit Prioritäten versehen werden
- Zeitpunkt bzw. Zeitraum: neben genau spezifizierten Terminen sollten auch vage Zeitangaben wie z.B. 'nächste Woche' verarbeitet werden können
- Zeitdauer
- Hauptthema des Treffens
- Wichtigkeit des Treffens

Mit Hilfe der obigen Parameter lassen sich jetzt folgende Strategien für eine Terminvereinbarung realisieren:

- eine *optimistische Strategie*
- eine *realistische Strategie*, und zwar in einer zentralisierten Version und in einer Umlaufversion

Im folgenden werden diese Strategien kurz beschrieben und als Kooperationsmethoden, zusammengesetzt aus Kooperationsprimitiven, in Form von höherwertigen Protokollen grafisch

[6] Beachte: Mensch $\neq$ Agent, sondern (Mensch + zugeordneter Benutzeragent) = Agent!

[7] Dies ist möglich, weil MEKKA ein sehr heterogenes Multiagenten-Modell zugrunde liegt.

[8] Der Initiator kann selbst Teilnehmer des Treffens sein, muß aber nicht.

Initiator	Teilnehmer
APP_PREPARE_PROPOSAL	
send_app_proposal	
— **PROPOSE (appointment, exact_time, ...)** ⟶	
	receive_app_proposal
	APP_EVALUATE_PROPOSAL
	send_app_reply
⟵ **ACCEPT** —	
⟵ **REJECT** —	
⟵ **MODIFY** —	
receive_app_reply	
APP_EVALUATE_REPLIES	
— **ORDER (appointment, ...)** ⟶	
goto(send_app_proposal)	goto(send_app_reply)
goto(app_prepare_proposal)	goto(send_app_info)

Tabelle 1: Optimistische Strategie

veranschaulicht. Die Tabellen verdeutlichen einerseits den Kontrollfluß innerhalb der Kooperation (Senden und Empfangen der Kooperationsprimitive), andererseits beinhalten sie aber auch die Funktionen[9], die sich im intelligenten Teil des Agenten, d.h. im Kopf, befinden und bei Abarbeitung der Kooperationsmethode aufgerufen werden.

Bei der *optimistischen Strategie* (s. Tabelle 1) spezifiziert der Initiator eines Treffens einen genauen Zeitpunkt mit einer genauen Zeitdauer. Er sendet diesen Vorschlag an die Benutzeragenten aller Teilnehmer. in der Hoffnung, daß diese ihn akzeptieren. Falls alle Teilnehmer zu dem spezifizierten Zeitpunkt frei sind und nach entsprechender Rückfrage den Terminvorschlag akzeptieren, wird der Termin automatisch in die persönlichen Kalender der Teilnehmer eingetragen; sollte einer der Teilnehmer zum vorgeschlagenen Termin verhindert sein bzw. den Termin bei der Rückfrage ablehnen, so teilt sein Benutzeragent bzw. er selbst dies dem Initiator mit, woraufhin dieser einen neuen Vorschlag unterbreiten muß. Da man aus Erfahrung weiß, daß es bereits bei einer kleinen Gruppe von drei bis fünf Personen schwierig ist, auf diese Art und Weise einen für alle geeigneten Termin zu finden, muß zu einer 'realistischeren Methode' übergegangen werden.

Die zentralisierte Version der *realistischen Kooperationsmethode* (s. Tabelle 2) ist eine Erweiterung der optimistischen Methode. Bei der Spezifikation des Zeitpunktes für ein Treffen können nun eine Menge von disjunkten Zeitintervallen angegeben werden. Einzige Bedingung für die Zeitintervalle ist, daß ihre Länge größer gleich der gewählten Zeitdauer des Treffens ist. Der grob spezifizierte Vorschlag wird an alle Teilnehmer gesendet, wo automatisch (ohne Benutzerinteraktion) ein Abgleich der vorgeschlagenen Zeiten mit den lokalen Terminkalendern vorgenommen wird. Ergebnis dieses Abgleichs sind jeweils die Zeiträume, zu denen der Teilnehmer am Treffen teilnehmen könnte, d.h. die Teilnehmer senden automatisch eine Rückmeldung an den Initiator mit ihren freien Zeitintervallen. Beim Initiator wird nun eine Auswertung dieser Rückmeldungen vorgenommen. Finden sich ein oder mehrere Zeitintervalle, an denen alle Teilnehmer noch nicht belegt sind, so werden diese Zeitintervalle - analog zur optimistischen Methode - den Benutzern vorgeschlagen, und zwar solange, bis ein Vorschlag von allen akzeptiert wird oder aber kein freies Intervall mehr vorhanden ist.[10] Findet sich nach Auswertung der freien Zeitintervalle der

[9] Diese Funktionen sind durch Verwendung des Fonts SMALL CAPS gekennzeichnet.

[10] Es kann ja vorkommen, daß ein ursprünglich freies Zeitintervall eines Teilnehmers mittlerweile durch einen anderen Terminvereinbarungsprozeß belegt worden ist.

Initiator	Teilnehmer
APP_PREPARE_PROPOSAL	
send_app_proposal	
— **PROPOSE (appointment, Rough_time, ...)** ⟶	
	receive_app_proposal
	APP_EVAL_FREE_TIME
	send_app_info
⟵ **REFINE (free_time)** —	
receive_app_info	
APP_EVAL_INFO	
send_app_proposal	
— **PROPOSE (appointment, exact time, ...)** ⟶	
	receive_app_proposal
	APP_EVALUATE_PROPOSAL
	send_app_reply
⟵ **ACCEPT** —	
⟵ **REJECT** —	
⟵ **MODIFY** —	
receive_app_reply	
APP_EVALUATE_REPLIES	
— **ORDER (appointment, ...)** ⟶	
goto(send_app_proposal)	goto(send_app_reply)
goto(app_prepare_proposal)	goto(send_app_info)

Tabelle 2: Realistische Strategie, zentralisierte Version

Teilnehmer kein gemeinsames Zeitintervall, so schlägt die Kooperationsmethode fehl.

An dieser Stelle bieten sich jetzt aber Erweiterungsmöglichkeiten in der Kooperationsmethode an, von denen zwei kurz angerissen werden:

- Verschieben von Terminen: Teilnehmer werden angehalten, bereits vereinbarte Termine, die aber eine geringere Priorität als der gegenwärtig zu verhandelnde Termin haben, zu verschieben.
- Die Initiatorrolle wird gegebenenfalls an jemanden übertragen, dessen Status gegenüber den anderen Teilnehmern herausragend ist und der deshalb die Teilnehmer zu einem Verschieben ihrer Termine 'zwingen' kann.

Hier kommen also die Parameter *Teilnehmerpriorität*, der die Rolle/den Status einer Person widerspiegelt, und *Terminpriorität* ins Spiel.

Die Umlaufversion einer Terminvereinbarung (s. Tabelle 3) läuft ähnlich zur zentralisierten Version ab: auch hier startet der Initator den Kooperationsprozeß mit einer Menge von zu verhandelnden Zeitintervallen. Allerdings senden die Teilnehmer, die jetzt nach aufsteigender Priorität geordnet sind, nicht ihre freien Zeitintervalle an den Initiator zurück, sondern sie verfeinern den ursprünglichen Vorschlag. Sie streichen die Zeitintervalle, an denen sie belegt sind und senden die modifizierte Zeitintervalliste an den nächsten Teilnehmer. Auf diese Weise werden die möglichen Zeitintervalle von Teilnehmer zu Teilnehmer immer weiter eingeschränkt. Die Beendigung der Kooperation läuft wiederum analog zur zentralisierten Version ab. Entweder ist innerhalb des Umlaufs die Liste der möglichen Zeitintervalle bereits leer; in diesem Fall sendet der gerade betroffene Teilnehmer eine entsprechende Nachricht an den Initiator; oder aber der letzte Teilnehmer sendet die Liste der Zeitintervalle, an denen alle Teilnehmer frei sind, an den

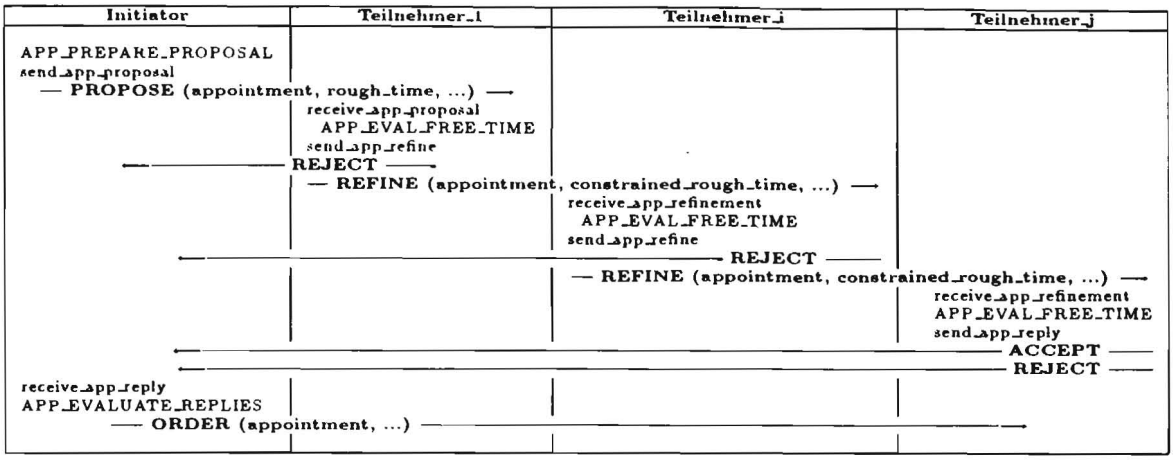


Tabelle 3: Realistische Strategie, Umlaufversion

Initiator zurück. Der Initiator startet dann nach der optimistischen Methode einen konkreten Terminvorschlag mit einem Termin aus dieser Liste.

Aus den bisherigen Ausführungen wie auch aus einem Vergleich der Tabellen 1 und 2 wird deutlich, daß die realistische Kooperationsmethode eine Erweiterung der optimistischen Kooperationsmethode darstellt.

4.4.3 Anwendung 2: Verkehrswesen

Auch der Bereich Verkehrswesen kann, im Gegensatz zum Großteil bisher existierender Forschung auf diesem Gebiet, als Multiagentensystem modelliert werden. Betrachtet man alle Elemente in einem solchen Szenario, also Menschen, Autos, Straßen, Ampeln, Parkhäuser usw. als Agenten eines komplexen Gesamtsystems, so ergibt sich eine Fülle von kooperativen Einzelszenarien. Aus dieser Fülle betrachten wir im folgenden stellvertretend die Kooperation zwischen Autos, die einen Parkplatz suchen, und den in Frage kommenden Parkhäusern.

Die Autos sind in einem ersten, einfachen Ansatz charakterisiert durch Ortskoordinaten und ein Budget, das ihnen zum Bezahlen der Parkgebühren zur Verfügung steht. Die Parkhäuser sind spezifiziert durch Ortskoordinaten, eine maximale Aufnahmekapazität, die momentane Auslastung und die Parkgebühren, die sie verlangen. Will nun ein Auto parken, so kann es verschiedene Strategien anwenden, um dieses Ziel zu erreichen.

Auto	Parkhäuser
SIMPLE_REQUESTING	
send_request	
— REQUEST →	
	receive_request
	SIMPLE_EVALUATE
	send_result
← (ACCEPT | REJECT) —	
goto(SIMPLE_REQ)	goto(SIMPLE_EVAL)

Tabelle 4: Einfaches Anfragen

Die einfachste Methode besteht für ein Auto (resp. seinen Fahrer) darin, sukzessive bei einem ihm bekannten Parkhaus nachzufragen, ob es dort parken kann. Hierbei sollen weder

Entfernungsüberlegungen noch anfallende Kosten eine Rolle spielen. Das *einfache Anfragen* (s. Tabelle 4) wird vom Parkhaus entweder akzeptiert oder abgelehnt.

Auto	Parkhäuser
CN_MAKE_REQUEST	
send_request_for_bids	
— REQUEST →	
	receive_request_for_bids
	CN_MAKE_BID
	send_bid
← PROPOSE —	
receive_bids	
CN_EVALUATE_BIDS	
send_assignment	
— (ACCEPT | REJECT) →	
	receive_assignment
	CN_EXECUTE_TASK
	send_result
← TELL —	
receive_result	
CN_EVALUATE_RESULT	
— ACCEPT →	
— REJECT →	
goto(CN_MAKE_REQUEST)	

Tabelle 5: contract net-Methode

Intelligenter ist dann schon ein Vorgehen nach der bekannten *contract net-Methode* (s. Tabelle 5). Das Auto sendet ein Aufforderung, ihm ein Angebot zu unterbreiten, an alle ihm bekannten Parkhäuser aus und erwartet deren Gebot. Nach Ablauf der Wartezeit wertet das Auto alle erhaltenen Angebote nach einem bestimmten Kriterium aus (geringste Kosten, geringste Entfernung, gewichtetes Mittel aus Kosten und Entfernung). Dann sendet es dem am besten bewerteten Parkhaus die Annahmebestätigung des Angebots zu, alle anderen Parkhäuser erhalten eine Ablehnung. Die Kooperation kommt zu einem guten Ende, wenn das ausgewählte Parkhaus beim Erhalt des Zuschlags noch freie Kapazitäten hat und zu seinem Gebot stehen kann. Sollte sich die Situation aber durch irgendwelche Gegebenheiten geändert haben, so schlägt diese vereinfachte Version des contract net fehl. Der gesamte Kooperationsprozeß müßte von neuem gestartet werden.

Deshalb wäre es geschickter, wenn das Auto nach der Auswertung der Gebote nicht direkt eine Ablehnung an die schlechter liegenden bzw. teureren Parkhäuser schicken würde, um bei einem evtl. Fehlschlagen des Parkvorhabens beim bevorzugten Parkhaus auf das zweit-, drittbeste, ... Parkhaus zurückgreifen zu können. Hierbei handelt es sich um eine *modifizierte contract net-Version* (s. Tabelle 6).

Eine weitere Möglichkeit zur Kooperation würde ein Verhandeln der Parkhäuser untereinander zum Zwecke des Lastausgleichs darstellen. Auch hier sind verschiedene Formen denkbar, etwa das Verhandeln unter Gleichberechtigten oder aber auch mit Parkhäusern, die aufgrund gewisser Attribute eine herausragende Stellung besitzen.

Auto	Parkhäuser
CN_MAKE_REQUEST send_request_for_bids — REQUEST →	
	receive_request_for_bids CN_MAKE_BID send_bid
← PROPOSE —	
receive_bids CN_EVALUATE_BIDS send_assignment	
— ACCEPT →	
	receive_assignment CN_EXECUTE_TASK send_result
← TELL —	
receive_result CN_EVALUATE_RESULT	
— ACCEPT → — REJECT →	
goto(send_assignment)	

Tabelle 6: Modifizierte contract net-Methode

4.4.4 Zusammenfassung

Zusammenfassend läßt sich festhalten, daß sowohl das Agentenmodell als auch das Kooperationsmodell von MEKKA aufgrund ihrer Applikationsunabhängigkeit für eine Vielzahl verschiedenartigster Anwendungen geeignet sind. Zur Zeit werden diese Konzepte unter Berücksichtigung unterschiedlicher Komplexität bzgl. der Agenten und Kooperationsstrukturen prototypisch implementiert und getestet. Die Implementation der Prototypen erfolgt in einer verteilten Umgebung von Unix-Workstations unter PARLOG/Prolog [Bur92]. Hierbei implementierte applikationsunabhängige Module fließen wiederum in MEKKA ein.

Literatur

[Bur92] Alastair Burt, Parlog+Prolog: Eine praktische Sprache für Multi-Agenten Systeme. In diesem Band. 1992.

[Ste91] Donald Steiner, MEKKA: Eine Entwicklungsumgebung zur Konstruktion kooperativer Anwendungen. In diesem Band. 1992.

4.5 TEAMCOM Applikation: Kooperatives Bestellwesen

von D. Scheidhauer

Die Applikation *Kooperatives Bestellwesen* nimmt gegenüber den anderen Applikationen in TEAMKOM eine Sonderstellung ein, insofern sie kein EG-Förderprojekt, sondern eine rein projektinterne Applikation darstellt, die dazu dient, die in TEAMKOM spezifizierten Konzepte anhand eines spezifischen Anwendungsszenarios prototypisch zu implementieren und auszutesten. Im Besonderen hilft sie mit, die Anforderungen an die KMC-Shell zu spezifizieren.

Dabei wird das kooperative Zusammenarbeiten der an einer Bestellung beteiligten Agenten in einer verteilten Umgebung behandelt.

Zusammen mit der Applikation "Multi-Mediales verteiltes Lernen in einer vernetzten Umgebung", die im Projekt SETT/MALIBU durchgeführt wird, zielen beide Applikationen auf die Realisierung einer anwendungsunabhängigen generischen Kommunikationsshell mit deren Hilfe schließlich dann einmal verteilte kooperative Anwendungen wie z.B. das kooperative Bestellwesen, relativ einfach und komfortabel konstruiert werden können.

Diese Applikationen werden beispielhaft zeigen, wie das kooperative Zusammenarbeiten geographisch voneinander getrennter Personen durch die Integration von multimedialen Kommunikationswerkzeugen optimal unterstützt werden kann. Verteilte Teamarbeit soll nicht nur technisch ermöglicht werden, sondern für die Teilnehmer vor allem auch stark vereinfacht werden.

Hierzu werden intelligente maschinelle Hilfesysteme miteinbezogen, so daß ein verteilte Mensch-Maschine Team entsteht. Die Applikation kooperatives Bestellwesen beinhaltet z.B die folgenden maschinellen Teammitglieder (maschinelle Agenten):

Angebotsagent

- sammelt Angebote bei den Firmen für die Kunden über gewünschte Funktionalitäten Produkte oder Produktkombinationen, die der Kunde zuvor mit dem Marktberater ausgewählt hat.

Formularagent

- unterstützt den Kunden bei der Auswahl und beim korrekten Ausfüllen des Bestellformulars.

Marktberater

- ermöglicht dem Kunden einen komfortablen Zugriff auf einen großen Produktkatalog, der Firmenkataloge mehrerer, verschiedener Angebotsfirmen vereint. Diese firmenspezifische Teilbereiche des Gesamtproduktkatalogs können von den entsprechenden Firmen ständig aktualisiert werden.
- gibt dem Kunden einen Überblick bzgl. der Produktpalette, die es auf dem Markt gibt (funktionaler und optischer Eindruck, allgemeine technische Beratung).
- erspart dem Kunden das Sammeln und Durchforsten von Katalogen, Broschüren bzw. das Besuchen der entsprechenden Firmen.
- stellt selbstständig Anfragen an die Firmen, falls der Kunde Wünsche hat, die in der Form im Katalog nicht vorhanden sind.
- hilft dem Kunden, die von ihm gewünschten Funktionalitäten und Eigenschaften zu konkretisieren und eine Produktschnittmenge zu bilden, die alle geforderten Funktionalitäten und weitere Constraints (wie z.B. Abmessungen, Preis) erfüllen.

Der Benutzer braucht kein großes technisches Wissen über die Handhabung des Systems zu besitzen:

- Bedienung seiner MultiMedia Workstation

- Vernetzung seiner Workstation mit denen der anderen Teammitgliedern, d.h. Aufbau bzw. Abbau der verschiedenen möglichen Verbindungen zu diesen Workstations, der Verbindungsarten, z.B. Daten (ftp, telnet, E-Mail), Audio, Video, und der Übertragungsmedien, z.B. ISDN, Ethernet, oder Telephonleitung.

Bei der Spezifikation der Applikation kooperatives Bestellwesen wurden die Beispielszenarios "Bestellung von Einbauküchen" und "Bestellung von Hard- und Software" durchgespielt, und für die Bestellung von Einbauküchen ein konkretes Drehbuch erstellt, das in detaillierter Form die verschiedenen benötigten Benutzeroberflächen und einen möglichen Ablauf eines Bestellvorgangs beinhaltet.

Für eine erste Demoversion wird ein abgemagertes Szenario mit vier eventuell fünf Agenten betrachtet und modelliert.

Menschliche Agenten:

Kunde oder Besteller

firmenneutraler Berater unterstützt durch Softwaretools, betreut den Kunden durch technische und logistische/finanzielle Beratung

ein Sachbearbeiter einer Firma X wobei als reines Auskunftssystem auch ein maschineller Agent denkbar ist.

ein Sachbearbeiter einer Firma Y eventuell, falls dies den Rahmen der Demoversion nicht sprengt

Maschinelle Agenten: (wie bereits beschrieben)

Marktberater

Angebotsagent gegebenenfalls mit Formularagent

Es wurden auch bereits einige Überlegungen zu der erforderlichen multimedialen Infrastruktur angestellt, die zur Realisierung der kooperativen Vorgänge zwischen den Agenten benötigt wird.

Der Marktberater wird einen auf mehrere Firmen verteilten "Multimediakatalog" verwalten, der neben Hypertext, Graphiken und Bildern auch "Bewegtbildprospekte" enthalten könnte (CD-ROM) auf denen neben der Optik auch die Funktionen der Objekte vorgeführt werden können.

5 Techniken

5.1 Wissen und Glauben in Multiagenten-Systemen: ein Überblick

von J. P. Müller

Verteilte Künstliche Intelligenz und dabei speziell der Bereich der Multiagentensysteme ist in den letzten Jahren zu einem der meistbeachteten neuen Teilgebiete der KI herangereift. Die Vorstellung, daß autonome wissensbasierte Computersysteme selbständig Entscheidungen treffen, miteinander kooperieren und verhandeln können, übt eine Faszination aus, derer sich weder Laien noch Fachleute leicht entziehen können.

In diesem Artikel möchten wir einen Überblick darüber geben, über welche Arten von Wissen Agenten in Multiagenten-Systemen verfügen müssen, um ihre Hauptaufgaben - kooperatives Lösen von Problemen - erfüllen zu können. Insbesondere möchten wir die Rolle von *Glauben* in diesem Kontext umreißen. Abschließend möchten wir einen Ausblick auf unsere weitere Tätigkeit auf diesem Gebiet geben.

5.1.1 Einleitung

Verteilte Künstliche Intelligenz (VKI) befaßt sich mit der Modellierung von Gesellschaften autonomer Systeme, die in kooperativer Weise Probleme lösen. Die einzelnen Systeme heißen *Agenten*. Die VKI läßt sich grob in zwei große Teilbereiche gliedern, nämlich *Verteiltes Problemlösen* und *Multiagentensysteme*. Beim Verteilten Problemlösen befaßt man sich mit der Frage, wie ein Problem von einer Gruppe von Agenten gelöst werden kann (vgl. [CL87]). Im Bereich der Multiagentensysteme untersucht man hingegen Gesellschaften autonomer Agenten, die in einer gemeinsamen Umwelt[Hew85, Hew86] und unter Benutzung gemeinsamer Ressourcen ihre eigenen lokalen Ziele verfolgen. Hier ist man insbesondere an Aspekten der Koordination, Konkurrenz und Kooperation interessiert, d.h., wie können die Agenten möglichst gut ihre Ziele erreichen (vgl. [BG88, GH89, DM90, DM91]).

Anwendungsgebiete, für die der Einsatz von Methoden der VKI besonders erfolgversprechend erscheint, lassen sich wie folgt charakterisieren;

- Anwendungen, in denen Probleme modelliert werden sollen, die inhärent verteilt sind, sich also durch die Verteilung von Ressourcen, Wissen und Problemlösefähigkeiten einer zentralen Lösung entziehen.
- Anwendungen, die sich in natürlicher Weise mit Hilfe der *Human Society Metaphor* beschreiben lassen, d.h. Systeme, die in ähnlicher Weise funktionieren wie menschliche Gesellschaften mit ihren diversen Formen der Organisation und Rollenbildungen.
- Anwendungen, die Probleme beinhalten, die sich von ihrer Komplexität her mit herkömmlichen Problemlösetechniken nicht lösen lassen, wie z.B. eine Reihe von Scheduling-Problemen. Für diese Probleme haben sich bereits Methoden der herkömmlichen KI bewährt. Wo die Abbildung dieser Probleme auf ein verteiltes System in natürlicher Weise möglich ist, bietet sich durch Parallelverarbeitung und durch Synergieeffekte die Möglichkeit eines weiteren qualitativen Sprungs.

Bekannte Anwendungen der VKI sind u.a. Büroverwaltung [Hew86], Verkehrssteuerung und -überwachung [CHRS+88, CDL87] und die Kopplung von Expertensystemen [Bon89]. Im Projekt AKA-Mod am DFKI untersuchen wir ein verteiltes Speditionsszenario (vgl. [BKMP92]),

in dem eine Gruppe von Speditionsgesellschaften Transportaufträge zu erledigen hat. Da die Aufträge die Kapazitäten einzelner Speditionen übersteigen können, müssen die Speditionen kooperieren, um gute Lösungen zu finden. Das Speditionsszenario verkörpert in natürlicher Weise alle oben genannten Aspekte: die natürliche Verteilung des Problems ist durch die örtliche Verteilung der Speditionen gegeben. Speditionen sind vom Menschen geschaffene Organisationen, die demzufolge ihre Probleme mit menschlichen Problemlösemethoden angehen, die miteinander konkurrieren, aber auch (wenn auch in der Realität noch zu selten) miteinander kooperieren. Weiterhin sind in dem Szenario sehr komplexe mehrdimensionale Scheduling-Probleme zu lösen, die die Verwendung von KI-Methoden geeignet erscheinen lassen.

Allen aktuellen Ansätzen zur Modellierung von Multiagentensystemen liegt ein wissensbasierter Ansatz zugrunde. Das Wissen, die Fähigkeiten und die Intentionen von Agenten werden explizit repräsentiert, ein Agent kann dabei als Wissensbasis zusammen mit einer Kontrollstruktur charakterisiert werden [BM91]. Dieses Wissen dient dem Agenten als Grundlage für seine Entscheidungen und sein Verhalten. Bezeichnend für Multiagentensysteme ist dabei der Umgang mit unsicherem, unvollständigem und sogar inkonsistentem Wissen. Da Multiagentensysteme dynamische *offene Systeme* sind, kann sich die Umwelt eines Agenten ohne dessen eigenes Einwirken ändern. Dies kann durch Interaktionen anderer Agenten verursacht werden. Zudem ist es möglich, daß Agenten fehlerhafte Informationen über ihre Umwelt erhalten, sei es durch die Unzuverlässigkeit ihrer eigenen Wahrnehmungskanäle, oder sei es durch fehlerhafte Information, die durch andere Agenten übermittelt worden ist. Agenten können sich irren, oder können unter Umständen sogar mit Absicht falsche Informationen verbreiten. Diese Einsicht hat zu einer Relativierung des Wissensbegriffs im Multiagenten-Kontext geführt. Der Begriff *Glauben* erfährt dabei in der aktuellen Literatur steigende Beachtung. *Eigentlich kann ein Agent nichts sicher wissen, er kann es nur glauben.*

In diesem Papier möchten wir einen Überblick über die Rolle und die Repräsentation von Wissen und Glauben in Multiagentensystemen geben. Im nächsten Abschnitt werden wir nach einigen grundsätzlichen Bemerkungen zu Wissen und Glauben näher auf die Bedeutung dieser Begriffe im Multiagenten-Kontext eingehen. Wir versuchen, eine Idee davon zu vermitteln, warum die explizite Repräsentation von dem, was ein Agent glaubt, für Multiagentensysteme so wesentlich ist. Abschließend wollen wir einen Ausblick auf unsere weitere Tätigkeit in diesem Bereich geben. Dazu geben wir einen kurzen Überblick über bisherige Ansätze zur Repräsentation von Wissen und Glauben.

5.1.2 Wissen und Glauben

Wissen und Glauben - Was heißt das?

Um diese Frage, und insbesondere darum, was den Unterschied zwischen beiden Begriffen ausmache, rankt sich eine Diskussion, die so alt ist wie die Philosophie selbst[11][Fre92]. So scheint es wenig sinnvoll, so etwas wie eine Definition dieser Begriffe angeben zu wollen. Stattdessen wollen wir uns darauf beschränken, einige wesentliche Eigenschaften aufzuführen, die der Wissensbegriff beinhalten soll. Dabei wollen wir uns auf das Verhältnis zwischen Wissen und Information konzentrieren.

Wissen entsteht im wesentlichen aus Information, die ein Agent wahrnimmt. Nicht alle wahrgenommene Information wird jedoch Wissen. In jedem Falle spielen die Zuverlässigkeit

[11]Schon Sokrates fühlte sich zu der halb verzweifelten, halb resignierenden Frage "Was ist Wissen? Kann irgendjemand diese Frage beantworten?" genötigt. Immerhin kam er wenigstens zu dem Schluß "Ich weiß, daß ich nichts weiß!"

der Informationsquelle, die Zuverlässigkeit der Informationskanäle selbst, das Verhältnis neuer Information zu bereits vorhandenem Wissen (Konsistenz) sowie die Situation, in der eine Information aufgenommen wird, eine entscheidende Rolle. Etwas, was ein Agent weiß, sollte zumindest wahr sein, sonst macht es wenig Sinn, von Wissen zu reden. Weiterhin hat Wissen etwas mit Bewußtsein zu tun. Kann man sagen, man wisse etwas, das einem nicht bewußt ist? In diesem Kontext ist das interessante Phänomen zu erwähnen, daß man sich an eine Sache nicht erinnert, aber dennoch weiß, daß man sie "eigentlich weiß" bzw. wußte.

Weiterhin hat Wissen etwas mit Relevanz zu tun, also der Beziehung zwischen der Information selbst und dem Empfänger der Information. Es scheint sinnvoll, Wissen und Glauben als Relationen zwischen einer Information (z.B. einer Aussage) und dem Agenten, der die Information wahrnimmt und bewerten soll, zu sehen (vgl. [MP92]). Die Frage nach den zusätzlichen Argumenten dieser Relation scheidet jedoch nach wie vor die Geister.

Was die Beziehung zwischen Glauben und Wissen anbelangt, gibt es im wesentlichen zwei unterschiedliche Sichten [Dav90]. Die eine besagt, daß Wissen eine stärkere Relation als Glauben sei. Die andere, die immer mehr Anhänger zu finden scheint, besagt, daß Glauben das allgemeinere Konzept als Wissen sei, d. h. , daß sich Wissen als Spezialisierung von Glauben ausdrücken lassen könne. Jedoch konnte auch hier noch keine allseits akzeptierte Lösung gefunden werden. Wissen ist sicherlich mehr als "wahrer Glaube", und, wie spätestens seit Gettier [Get63] bekannt ist, auch mehr als "wahrer, berechtigter Glaube".

Von entscheidender Bedeutung dafür, wie man die Beziehung zwischen Wissen und Glauben definiert, ist, welche Art von Wissen und Glauben man modellieren möchte. Diese Entscheidung bestimmt die Eigenschaften, die man den beiden Begriffen geben möchte. Diese Frage soll im folgenden Abschnitt näher untersucht werden.

Wissen und Glauben in Multiagentensystemen

Multiagentensysteme sind offene Systeme. Das heißt, daß die Welt sich durch die Einwirkung von Agenten dynamisch ändert, daß ständig neue Agenten hinzukommen und andere aus der Welt verschwinden können. Dies impliziert, daß nicht jeder Agent alle anderen Agenten kennen kann, er demzufolge deren Aktionen und ihre Auswirkungen auf die Welt gar nicht berücksichtigen kann.

Nehmen wir an, unsere Welt enthält zwei Agenten, Agent A und Agent B. Agent A möchte eine schwere Kiste durch eine Tür tragen. Er öffnet dafür die Tür, dreht sich um und hebt die Kiste auf. Kann man jetzt sagen, A wisse, daß die Tür offen ist? Die Antwort könnte wie folgt lauten:

> In dem Moment, in dem er sie aufmacht, oder sieht, daß sie auf ist, weiß er es. Dreht er sich um und hebt die Kiste auf, weiß er es nicht mehr, denn Agent B könnte inzwischen den Zustand der Welt verändert, sprich, die Tür geschlossen haben.

Selbst den ersten Teil der Aussage muß man unter Umständen revidieren, wenn man annimmt, daß Agent A schlecht sieht. Dann kann er zwar glauben, gesehen zu haben, daß die Tür auf ist, man würde aber hier nicht von Wissen sprechen. Man kann in jedem Fall beobachten, daß das Vertrauen in eine Information mit der Zeit nach Wahrnehmen der Information abnimmt. Dies gilt vor allem für Faktenwissen, d. h. Wissen über die Welt. Methodenwissen ist von diesem Phänomen weniger berührt. Weiß ein Agent, daß er eine Kiste von einem bestimmten Gewicht tragen kann, so wird er dies wohl eine Stunde oder einen Tag später auch noch wissen. Dies gilt dagegen nicht für die Aussage *"Kiste K steht in Raum R"*!.

Wenn man über Multiagentensysteme spricht, scheint es angesichts dieser Beobachtungen sinnvoll, den Begriff Wissen durch "Sehr starkes Glauben" zu ersetzen. Dadurch ist man, selbst wenn man bezüglich einer gegebenen Information die stärkstmögliche Form des Glaubens zugrundelegt, nicht verpflichtet, anzunehmen, daß das, woran man glaubt, auch wirklich gilt. Der oben beschriebene Dynamik der Relation zwischen einem Agent und der Information, die er hat, kann nach unserer Meinung durch eine Konzept von *Glaubensabstufungen* (engl. *degrees of belief*) am besten entsprochen werden. Diese Abstufungen können entweder quantitativ (z.B. 1 für *sehr stark glauben*, 0 für *keine Meinung dazu haben/nichts darüber wissen*, -1 für *sehr stark bezweifeln*) sein, oder aus einer qualitativen Abstufung wie zum Beispiel $\{$*stark glauben*, *glauben*, *schwach glauben*, *indifferent sein*, *schwach zweifeln*, *zweifeln*, *stark zweifeln*$\}$ bestehen.

Zusammenfassend kann man sagen, daß wir, wenn wir in Multiagenten-Systemen über Glauben reden, vorrangig daran interessiert sind, die Unsicherheit des Wissens von Agenten zu modellieren. Demzufolge scheinen sich für die Darstellung von Glauben in unserem Kontext insbesondere die Mechanismen anzubieten, die man in der KI bereits zur Modellierung unsicheren Wissens kennt.

Die Bedeutung von Glauben in Multiagentensystemen

In diesem Abschnitt werden wir uns mit der Frage befassen, wozu eine explizite Repräsentation dessen, was Agenten glauben, überhaupt nötig ist. Wir wollen dabei mit einer kleinen Topologie des Glaubens beginnen.

Topologie des Glaubens Es gibt viele verschiedene Arten, Wissen und Glauben zu klassifizieren. In [KR87] wird insbesondere zwischen problemorientierten und systemorientierten Klassifikationen unterschieden. Da unsere Agenten in erster Linie Problemlöser sind, möchten wir hier eine problem-orientierte Klassifizierung angeben, die uns für die Verwendung in Multiagentensystemen sinnvoll scheint. Dabei klassifizieren wir nach drei Dimensionen

- Dem Bezug des Glaubens. In Multiagentensystemen kann man hier zwischen Glaube über sich selbst, Glaube über andere Agenten und Glaube über die Welt unterscheiden.
- Der Gegenstand, d.h. das, was geglaubt wird. Hier kann man zwischen dem Glauben von Fakten und dem Glauben von Methoden unterscheiden.
- Der Ebene des Glaubens. Hier unterscheidet man die Objektebene und die Meta-Ebene. Obwohl die Meta-Ebene typischerweise ein Teilbereich des "Methoden-Glaubens" ist, sind die Gegenstandsdimension und die Ebenendimension doch nicht äquivalent, denn es können sowohl "Objekt-Fakten" (z.B. "Block A ist rot") als auch Fakten auf Meta-Ebene (z.B. die Bewertung von Methoden - "Ich kann schwimmen!") geglaubt werden.

Bild 18 zeigt den sich daraus ergebenden Klassifikationsbaum. Kombinationen, die praktisch nicht auftraten, sind schraffiert. Dabei wird ersichtlich, wie sich Begriffe wie epistemisches und autoepistemisches Wissen in der Klassifizierung wiederfinden lassen.

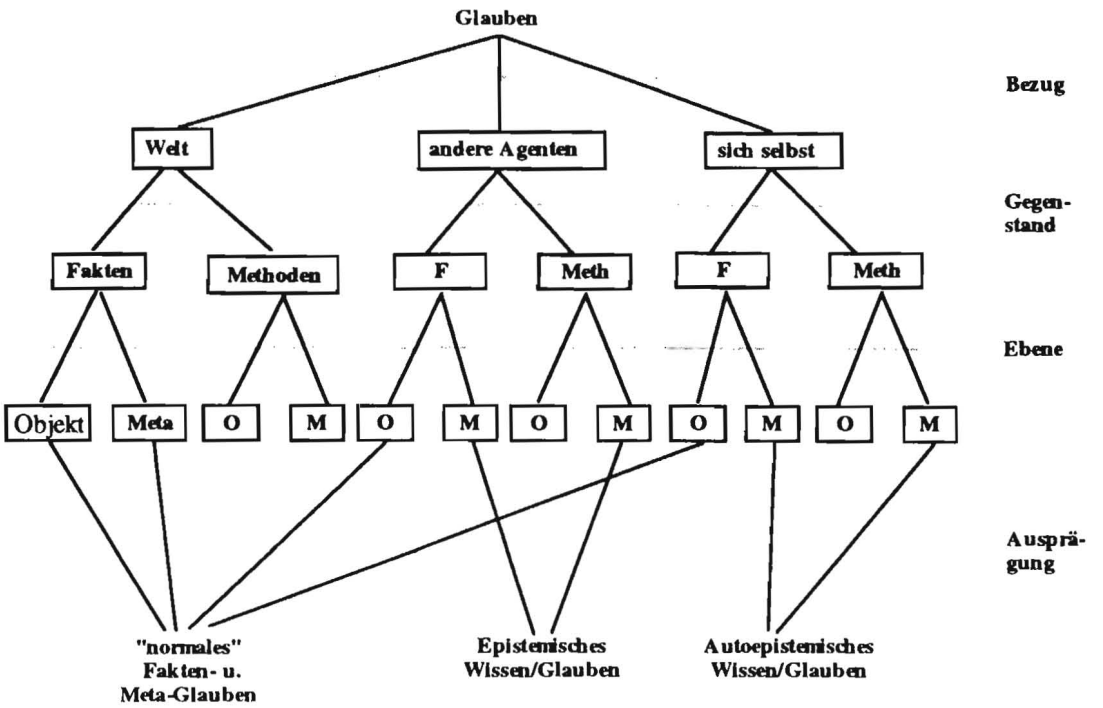


Abbildung 18: Eine Topologie des Glaubens in Multiagentensystemen

Glauben als Grundlage für Entscheidungen

Kooperierende Agenten in Multiagentensystemen planen ihre Aktionen und treffen ihre Entscheidungen auf der Basis ihres Wissens [Moo80, Moo85] bzw., in unserer Terminologie, auf der Basis dessen, was sie glauben. Dabei sind folgende Teilbereiche von besonderer Bedeutung.

- **Glaubensbasierte Kompetenzbewertung:**
  Agenten müssen beurteilen können, ob sie eine Aufgabe, die ihnen aufgetragen wird, lösen können. Dies erfordert sowohl Faktenwissen über sich selbst, d.h. über lokale Ressourcen, als auch Metawissen/-glauben (autoepistemisches Wissen / Glauben) über ihre eigenen Fähigkeiten.

- **Glaubensbasierte Aufgabenverteilung:**
  Soll ein Multiagentensystem eine gestellte Aufgabe sinnvoll lösen können, so muß die Aufgabe in der Regel in Teilaufgaben gegliedert werden. Die einzelnen Teilaufgaben müssen auf einzelne Agenten zur Lösung verteilt werden [KMM92]. Die Zerlegung und Verteilung der Aufgaben hat in geeigneter Weise, d.h. unter Berücksichtigung der Fähigkeiten, Ziele, Präferenzen und der sozialen Rolle der einzelnen Agenten zu geschehen. Somit erfordern Entscheidungen der Aufgabenzerlegung und -verteilung, daß Agenten Modelle über die Fähigkeiten, Ziele etc. andere Agenten haben.

- **Glaubensbasierte Kooperationsentscheidungen:**
  Stellt ein Agent beim Erledigen einer Aufgabe fest, daß er diese Aufgabe entweder nicht allein ausführen kann, oder daß er die Aufgabe *besser* ausführen kann, indem er mit einem anderen Agenten kooperiert, so ist auch die Entscheidung, wie, wann und mit wem er kooperieren soll, wesentlich von dem abhängig, was er von anderen Agenten und von der Welt glaubt. Muß z.B. für eine gewisse Kooperationsform ein geeigneter Kooperationspartner

einer Spedition auf den Nahverkehr spezialisiert sein, so werden die Entscheidungen der Spedition betreffend der Auswahl eines geeigneten Partners dadurch gelenkt, wen sie für spezialisiert dafür hält, und nicht dadurch, welche Speditionen tatsächlich Nahverkehrsspezialisten sind.

- **Glaubensbasierte Verhandlungsführung:**
  Verhandlungen sind ein wesentlicher, wahrscheinlich sogar der wesentlichste Aspekt der Modellierung autonomer intelligenter Systeme. Nur über Verhandlungen können sich Agenten auf ein gemeinsames Verhalten, auf die Erreichung gewisser Ziele einigen, nur über Verhandlungen können Konflikte gelöst werden. Es ist offensichtlich [KMM92, MP92], daß es für die Entscheidungen darüber, wann, mit wem und worüber zu verhandeln ist, sowie für das Verhalten der Agenten im Verlauf einer Verhandlung wesentlich ist, Wissen/Glauben über die Ziele, Intentionen, Ressourcen, Pläne etc. des Verhandlungspartners bzw. -gegners zu haben. Dies läßt sich sehr gut am Beispiel des Feilschens um einen Preis zwischen dem Anbieter einer Dienstleistung und dem Interessenten an dieser Dienstleistung (Anbieter-Kunde Verhältnis) beobachten. Das aktuelle Gebot eines Agenten hängt zum einen von einer Reihe von Kriterien ab, die sich aus der eigenen Situation des Agenten ergeben, zum anderen aber auch von seiner Einschätzung des Verhandlungspartners, von dessen Kriterien, seinen Ressourcen und der Bedeutung, die das Erreichen des Verhandlungsziels für ihn hat. Diese Einschätzung, um es noch einmal zu betonen, beruht nicht auf sichere rInformation, sondern auf dem, was der Agent glaubt.

- **Glaubensbasierte Planung von Aktionen:**
  Agenten haben Ziele und planen, um diese Ziele zu erreichen. Bei der Planung stützen sie sich auf die ihnen zugängliche Information, also auf das, was sie glauben. Müssen Agenten die voraussichtlichen Aktionen anderer Agenten bei ihrer Planung mit einbeziehen, so erfordert auch dies die Verwaltung eines Partnermodells.

5.1.3 Ausblick

In diesem Papier haben wir einen motivierenden Überblick über die Bedeutung von Wissen und Glauben von Agenten in Multiagentensystemen gegeben. Unsere weitere Arbeit auf diesem Gebiet wird sich im wesentlichen Fragen der Repräsentation von Wissen und Glauben widmen. Die entscheidenden Fragen hierbei lauten: *Welche Aspekte von Glauben sollen repräsentiert werden?* und *Wie sollen sie repräsentiert werden?*.

Was die erste Frage anbelangt, so ergeben sich die grundlegenden Aspekte aus den in Abschnitt 2 genannten Anwendungsgebieten. Dabei wollen wir in erster Linie den Aspekt der Unsicherheit behandeln: Glauben unterscheidet sich für uns von Wissen primär dadurch, daß das, was man glaubt, nicht wahr zu sein braucht. Auch die Frage der zu verwendenden Repräsentationsmechanismen läßt sich nur im Zusammenhang mit den Gesichtspunkten beantworten, die dargestellt werden sollen. Im folgenden möchten wir einen kurzen Überblick auf die wichtigsten bestehenden Ansätze zur Repräsentation von Wissen und Glauben geben.

Die bisherige Forschung zur Repräsentation von Wissen und Glauben geht fast immer von einem logikbasierten Ansatz aus. Dabei genießt der modallogische Ansatz mit seiner auf sogenannten *möglichen Welten* basierenden Semantik [Hin62, Kri63] große Bedeutung. Dabei werden verschiedene Zustände der Welt, die ein Agent für möglich hält, durch verschiedene Welten dargestellt. Der Tatbestand des *Für-Möglich-Haltens* wird durch eine übergangsrelation zwischen Welten dargestellt. Man definiert Wissen bzw. Glauben dabei als Gültigkeit in allen möglichen (d.h. von der aktuellen, tatsächlichen Welt erreichbaren) Welten. Diesem Ansatz liegt die Idee zugrunde, daß das Nicht-Wissen einer Aussage gleichbedeutend damit ist, daß man bezüglich

dieser Aussage mehrere Welten für möglich hält (nämlich mindestens eine, in der die Aussage gilt, und eine in der sie nicht gilt). Die verschiedenen Eigenschaften von Wissen und Glauben werden dabei durch eine Reihe von modallogischen Axiomen ausgedrückt, denen verschiedene Eigenschaften der Erreichbarkeitsrelation entsprechen (siehe [Moo80, Dav90, HM92] als Überblick). Ein für die Unterscheidung zwischen Wissen und Glauben zentrales Axiom ist dabei das *Wissensaxiom*

$$K_i\Phi \Rightarrow \Phi,$$

wobei K ein modallogischer Operator für Wissen bzw. Glauben, i die Bezeichnung für einen bestimmten Agenten, und Φ eine (aussagenlogische, $\mathcal{ALC}_{\mathcal{K}}$ [LM92] oder prädikatenlogische) Formel ist. Dies entspricht der Eigenschaft von Wissen, stets wahr zu sein, und wird durch die Reflexivität der Erreichbarkeitsrelation ausgedrückt. Will man Glauben und die darin enthaltene Unsicherheit ausdrücken, so ist dieses Axiom sicher nicht angemessen.

Der modallogische Ansatz birgt zum einen einige grundsätzliche Probleme in sich, wie das der *logischen Allwissenheit* (ein Agent weiß bzw. glaubt alle Folgerungen aus seinem tatsächlichen Wissen). Zum anderen ist die Frage einer angemessenen Axiomatisierung des Wissens-, und in stärkerem Maße noch des Glaubensbegriffs sehr schwer zu beantworten, was sich durch die Vielzahl der vorgeschlagenen unterschiedlichen Axiomensysteme belegen läßt. Sicherlich läßt sich der qualitative Unterschiede zwischen Wissen und Glauben und die damit verbundene Charakterisierung des Glaubensbegriffs nicht einfach dadurch erhalten, daß man das Wissensaxiom wegläßt. Desweiteren ist es keineswegs klar, ob sich der modallogische Weltengraph implizit aus den epistemischen Formeln in der Wissensbasis ergeben soll, oder ob die zumindest partielle Konstruktion des Weltengraphen, d.h. also, die Frage, was denn überhaupt mögliche Welten seien. am Anfang der Entwicklung einer Wissenskomponente für einen Agenten stehen soll. Zudem sind ursprünglich modallogische Ansätze zur Modellierung von Wissen und Glauben primär für den Ein-Agenten-Fall konzipiert und müssen erweitert werden, um den Erfordernissen der Modellierung verteilter Systeme, wie z.B. der Unsicherheit von Kommunikationskanälen, Rechnung zu tragen.

Als Reaktion auf diese Schwächen des klassischen Kripkemodells sind eine Reihe alternativer, *nicht-klassischer* Konzepte zur Modellierung von Wissen und Glauben entwickelt worden. Diese lassen sich grob in drei Richtungen gliedern:

1. Weiterentwicklung des modallogischen Ansatzes.
2. Der sententielle oder syntaktische Ansatz.
3. Nicht-klassische und nicht-monotone Logiken zur Modellierung unsicheren Wissens.

Die erste Richtung umfaßt eine Erweiterung von Modallogiken für den Mehragentenfall, sowie die Modellierung verschiedener Arten von Gruppenwissen wie z.B. den Begriff des *Common Knowledge* [HM90]. Zum anderen wurden Ansätze zur Integration von Agenten mit begrenzten Berechnungsfähigkeiten [Mos88], sowie von probabilistischen und evidenztheoretischen Elementen [FH91, FH92] in die Mögliche-Welten-Semantik vorgeschlagen. Ein weiterer Ansatz, dem Problem der logischen Allwissenheit Herr zu werden, liegt in der Einführung sogenannter *unmöglicher Welten*, sowie in der Unterscheidung zwischen *explizitem* und *implizitem Glauben*[Lev84, Lak86, Lev90]. Levesque's Arbeit adressiert insbesondere den Zusammenhang zwischen Glauben und Bewußtsein, indem explizites Glauben als implizites bewußtes Glauben interpretiert wird. [Bou92] hat den erweiterten modallogischen Ansatz dazu verwendet, um verschiedene Grade des Glaubens zu modellieren, indem er zwischen mehr oder weniger unmöglichen Welten unterscheidet.

Beim syntaktischen Ansatz wird Wissen bzw. Glauben durch eine Menge von Formeln repräsentiert. Syntaktische Theorien sind Theorien erster Ordnung, die durch eine Menge von Sätzen (Strings) erweitert sind. Während diese Repräsentationsmethode das Problem der logischen Allwissenheit löst (die Formelmenge, die das repräsentiert, was ein Agent glaubt, braucht nicht mehr abgeschlossen bezüglich der logischen Folgerung zu sein), sind syntaktische Theorien sehr schwer analysierbar und axiomatisierbar [Hal86]. Zudem ist es sehr leicht möglich, Widersprüche in diesen Theorien zu konstruieren[Dav90]. Dieses Problem tritt durch die Möglichkeit der Selbstreferenz auf, da sich Sätze auf Strings beziehen können, in denen sie selbst enthalten sind. Konolige [Kon85] hat den syntaktischen Mechanismus dahingehend erweitert, daß er das, was ein Agent glaubt, als den Abschluß einer Menge von Grundfakten bezüglich eines Systems von Ableitungsregeln definiert. Diese Menge von Ableitungsregeln kann dabei unvollständig sein[12].

Die dritte Richtung sind die nicht-klassischen Logiken zur Darstellung von unsicherem Wissen. Diese umfassen das große Gebiet des *Nicht-monotonen Schließens*, als dessen Hauptansätze die *Default Logik* [Rei80], Autoepistemische Logik [Moo83], die eigentliche nicht-monotone Logik [DM80] und Circumscription [McC80] betrachtet werden können. Desweiteren sind probabilistische und induktive Ansätze [Lou87] zu nennen.

Zusammenfassend ist eine weite Kluft auf der einen Seite zwischen den formal-theoretischen, logik-basierten Ansätzen und auf der anderen Seite der Repräsentation der Ansätze zur Modellierung von Wissen und Glauben in aktuellen Anwendungen der VKI festzustellen, die für diese Probleme immer noch größtenteils *ad-hoc* Lösungen vorgibt. Diese Kluft ist zum einen darauf zurückzuführen, daß das Gebiet der Verteilten Künstlichen Intelligenz noch recht jung ist, und die Problemlöse- und Wissensrepräsentationsmechanismen doch noch in den Kinderschuhen stecken. Zum anderen berücksichtigen aber viele, und insbesondere die logischen Ansätze, zu wenig das Argument, daß VKI-Systeme soziale Systeme sind, die eine adäquate Darstellung von Wissen z.B. für Kooperation und Verhandlungen erfordern. Vielleicht ist es das große Dilemma der KI, daß man versucht, soziales (menschliches) Verhalten, welches nun einmal nicht-logische und irrationale Elemente enthält, mit Methoden der Logik nachzubilden. Andererseits bietet die Logik eine wohlfundierte mathematische Ausgangsbasis, mit Hilfe derer man weite Bereiche menschlichen Handelns, nämlich eben die logisch erfaßbaren, darstellen kann. Zudem ermöglicht Logik, Aussagen über Eigenschaften und Verhalten dieser Systeme zu treffen. Möglicherweise besteht die momentan einzige Möglichkeit, dieses Dilemma zu bewältigen, darin, zu versuchen, diese bestehende Kluft von beiden Ebenen, der logisch-formalen ebenso wie der außerlogisch-informellen aus, Schritt für Schritt zu verkleinern, und damit eine immer größere Bandbreite von Problemen einer Lösung im Rahmen der VKI zugänglich zu machen. Hierin sehen wir unsere Aufgabe.

Unsere weitere konkrete Arbeit wird darin bestehen, die vorstehend genannten Ansätze auf ihre Eignung zur Repräsentation von Wissen und Glauben in Multiagentensystemen, und darin speziell in Verhandlungen zwischen Agenten (siehe z.B. [DL89, ZR89, ZR92, KvM91, Bus92]) zu überprüfen. Möglichkeiten und Grenzen dieser Ansätze für diesen Anwendungsbereich sollen herausgearbeitet werden. Ziel ist es, einen *trade-off* zwischen formaler Beschreibbarkeit einerseits sowie Ausdruckskraft und Anwendbarkeit in praktischen Multiagenten-Szenarien zu finden, wie z.B. dem Speditionsszenario[BKMP92] oder dem Loading-Dock Szenario, die im Projekt AKA-MOD entwickelt werden.

[12] Konolige weist darauf hin, daß man sogar von der Korrektheitsanforderung für die Ableitungsregeln abgehen könne.

Literatur

[BG88] A. Bond and L. Gasser. *Readings in Distributed Artificial Intelligence.* Morgan Kaufmann, Los Angeles, CA, 1988.

[BKMP92] M. Buchheit, N. Kuhn, J. P. Müller, and M. Pischel. Mars: Modelling a multiagent scenario for shipping companies. In *Proceedings of ESS92, Dresden.* Forthcoming, 1992.

[BM91] H. J. Bürckert and J. Müller. Ratman: Rational agents testbed for multi agent networks. In *Decentralized A.I. 2.* North Holland, 1991.

[Bon89] A. H. Bond. The cooperation of experts in engineering design. In *Distributed Artificial Intelligence, Volume II,* pages 463–486. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1989.

[Bou92] C. Boutilier. A logic for revision and subjunctive queries. In *Proceedings of AAAI-92, San Jose, CA,* pages 609–615. AAAI Press, Menlo Park, CA, 1992.

[Bus92] S. Bussmann. Simulation Environment for Multi-Agent Worlds. Technical Report DFKI Document D-92-01, DFKI, Saarbrücken, January 1992.

[CDL87] D. D. Corkill, E. H. Durfee, and V. R. Lesser. Coherent cooperation among communicating problem solvers. In *IEEE Transactions on Computers,* C-36, pages 1275–1291, 1987.

[CHRS+88] S. Cammarata, F. Hayes-Roth, R. Steeb, P. Thorndyke, and R. Wesson. Architectures for distributed air-traffic control. In A. H. Bond and L. Gasser, editors, *Readings in Distributed Artificial Intelligence,* pages 102–105. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1988.

[CL87] D. D. Corkill and V. R. Lesser. Distributed problem solving. In S. C. Shapiro, editor, *Encyclopedia of Artificial Intelligence,* pages 245–251. John Wiley and Sons, New York, 1987.

[Dav90] E. Davis. *Representations of Commonsense Knowledge.* Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990.

[DL89] E. H. Durfee and V. R. Lesser. Negotiating task decomposition and allocation using partial global planning. In *Distributed Artificial Intelligence, Volume II,* pages 229–244, San Mateo, CA, 1989. Morgan Kaufmann Publishers, Inc.

[DM80] J. Doyle and J. McDermott. Non-monotonic logic i. *Artificial Intelligence,* 13:27–40, 1980.

[DM90] Y. Demazeau and J.-P. Müller, editors. *Decentralized A.I.* North-Holland, 1990.

[DM91] Y. Demazeau and J.-P. Müller. *Decentralized A.I. 2.* North Holland, 1991.

[FH91] R. Fagin and J. Y. Halpern. Uncertainty, belief, and probability. *Computational Intelligence,* 7:160–173, 1991.

[FH92] R. Fagin and J. Y. Halpern. Two views of belief. *Artificial Intelligence,* 54(3):275–317, April 1992.

[Fre92] G. Frege. Über Sinn und Bedeutung. *Zeitschrift für Philosophie und Philosophische Kritik,* 100:25–50, 1892.

[Get63] P. Gettier. Is justified true belief knowledge? Analysis, 1963.

[GH89] L. Gasser and M.N. Huhns. *Distributed Artificial Intelligence, Volume II.* Research Notes in Artificial Intelligence. Morgan Kaufmann, San Mateo, CA, 1989.

[Hal86] J. Y. Halpern. Reasoning about knowledge: an overview. In *Proceedings of TARK'86,* pages 1–15, 1986.

[Hew85] C. E. Hewitt. The challenge of open systems. *Byte,* 4(10), 1985.

[Hew86] C. Hewitt. Offices are open systems. *ACM Trans. on Office Information Systems,* 4:271–286, 1986.

[Hin62] J. Hintikka, editor. *Knowledge and Belief.* Cornell University Press, 1962.

[HM90] J. Y. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, 37(3):549–587, 1990.

[HM92] J. Y. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54:319–379, 1992.

[KMM92] N. Kuhn, H. J. Müller, and J. P. Müller. Task decomposition in dynamic agent societies, 1992.

[Kon85] K. Konolige. Belief and incompleteness. In J. R. Hobbs and R. C. Moore, editors, *Formal Theories of the Commonsense World*, pages 359–403. Ablex Publishing Corporation, 1985.

[KR87] R. Krickhahn and B. Radig. *Die Wissensrepräsentationssprache OPS-5*. Vieweg und Sohn, Braunschweig/Wiesbaden, 1987.

[Kri63] S. Kripke. Semantic analysis of modal logics, 1963.

[KvM91] T. Kreifelts and F. v. Martial. A negotiation framework for autonomous agents. In Y. Demazeau and J.-P. Müller, editors, *Decentralized A.I. 2*. North-Holland, 1991.

[Lak86] G. Lakemeyer. Steps towards a first-order logic of explicit and implicit belief. In *Proceedings of TARK'86*, pages 325–339, 1986.

[Lev84] H. J. Levesque. A logic of implicit and explicit belief. In *Proceedings of AAAI-84*. Austin, TX, 1984.

[Lev90] H. J. Levesque. All i know: A study in autoepistemic logic. *Artificial Intelligence*, 42:263–309, 1990.

[LM92] A. Laux and J. P. Müller. On the representation of knowledge and belief in multiagent systems. In *Proc. of 3rd. Workshop on Belief Representation and Agent Architectures*, Durham, UK, 1992. University of Durham, Dep. of Computer Science.

[Lou87] R. P. Loui. *Theory and Computation of Uncertain Inference and Decision*. PhD thesis, Department of Computer Science, University of Rochester, Rochester, NY, 1987.

[McC80] J. McCarthy. Circumscription–a form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–40, 1980.

[Moo80] R. C. Moore. Reasoning about knowledge and action. Technical Report Technical Note 191, SRI International, 1980.

[Moo83] R. C. Moore. Semantical considerations on non-monotonic logic. In *Proceedings of IJCAI-83*, pages 272–279, 1983.

[Moo85] R. C. Moore. A formal theory of knowledge and action. In J. R. Hobbs and R. C. Moore, editors, *Formal Theories of the Commonsense World*, pages 319–358. Ablex Publishing Corporation, 1985.

[Mos88] Y. Moses. Resource-bounded knowledge and belief. In M. Y. Vardi, editor, *Proc. of the Second TARK*. Morgan-Kaufmann, Inc., 1988.

[MP92] J. P. Müller and M. Pischel. Knowledge and Belief of Cooperating Agents In K. Sundermeyer, editor, *Proc. of the DAI Workshop at GWAI*. Bonn, 1992.

[MPS92] J. P. Müller, M. Pischel, and A. Schroth. MARS: Eine Multiagenten-Testumgebung mit Graphischer Benutzerschnittstelle. Internal Working Paper, 1992.

[Rei80] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[ZR89] G. Zlotkin and J. S. Rosenschein. Negotiation and task sharing among autonomous agents in cooperative domains. In *Proc. of the Eleventh IJCAI*, pages 912–917. Detroit, Michigan, August 1989.

[ZR92] G. Zlotkin and J. S. Rosenschein. A domain theory for task oriented negotiation. In A. Cesta, R. Conte, and M. Micheli, editors, *Pre-Proceedings of MAAMAW-92*, July 1992.

5.2 Parlog+Prolog: Eine praktische Sprache für Multi-Agenten-Systeme

by Alastair Burt

Da sie logische Programmiersprachen sind, besitzen Parlog und Prolog die elegante Eigenschaft, eine deklarative Bedeutung in der Logik als Hornklausen zu haben. Die deklarative Herkunft bringt die praktischen Vorteile mit sich, daß beide Sprachen mächtige abstrakte Kontrollkonstrukte und automatisches Memory-Management anbieten.

5.2.1 Vorteile von Prolog

Zusätzlich haben Prolog und Parlog ihre eigenen Stärken. In Prolog sind diese u.a.: *Unifikation*, *automatisches Suchverfahren*, und *einfaches Metaprogrammieren*. Unifikation ersetzt zwei Operationen in anderen Sprachen: Testen und Zuweisen. Dadurch wird Parameterübergabe in Prolog auf eine sehr allgemeine Art realisiert. Eine Prädikatdefinition, die einer Prozedurdefintion in anderen Sprachen gleicht, kann mehrere Arten von Ein- Ausgabeverhalten haben. Man kann zum Beispiel dasselbe Prädikat verwenden, um zwei Listen zusammenzufügen oder aber eine Liste in zwei Unterlisten zu zerteilen.

Backtracking ist das automatische Suchverfahren, wodurch ein Prologinterpreter mit Indeterminismus umgeht. Das oben erwähnte Prädikat, welches eine Liste in zwei Unterlisten zerteilt, wird normalerweise nichtdeterministisch geschrieben. Für eine nicht-leere Liste, (`[1, 2]`, zum Beispiel), gibt es mehr als ein mögliches Paar als Antwort (`[1, 2]-[]`, `[1]-[2]` oder `[]-[1, 2]`). Mit Prolog kann man alle Antworten in einer bestimmten Reihenfolge bekommen. Ein Prologprädikat kann aus mehreren Klausen bestehen. Der Prologinterpreter versucht mit der ersten Klause eine Antwort zu erzeugen. Wenn diese scheitert, „backtrackt" er und versucht, eine Antwort mit der nächsten zu finden. In anderen Sprachen muss der Programmierer ein Suchverfahren explizit codieren.

Weil die Kontrollkonstrukte sehr abstrakt sind, ist ein Interpreter für Prolog sehr leicht in Prolog selbst zu codieren. Es wird zusätzlich dadurch erleichtert, daß Prolog Primitive anbietet, welche gestatten, dynamisch neue Prädikate zu definieren, sie zu evaluieren uvm. Mit einem in Prolog geschriebenen Metainterpreter kann man über die Eigenschaften eines Programmes schließen und die Funktionalität eines Prologprogrammes erweitern. Das Erzeugen von Beweisbäumen und eine objektorientierte Erweiterung von Prolog sind typische Beispiele.

5.2.2 Vorteile von Parlog

Die Vorteile von Parlog gegenüber Prolog sind: *Parallelismus*, *bedingter Indeterminismus* und *Datenfluß-Kontrollkonstrukte*. Man kann Prädikate in Parlog so aufrufen, daß sie gleichzeitig laufen. Das bringt zum einen eine Effizienzsteigerung, da mehrere Prozesse an einem Problem arbeiten können, und zum anderen lassen sich viele Applikationen einfacher als eine Sammlung von nebenläufigen, interaktiven Objekten beschreiben.

Im Gegensatz zu Prolog löst Parlog Indeterminismus nicht durch Backtracking auf, sondern durch das Guard-Konzept. Jede Prädikatdefinition P besteht aus einer Menge von Guard-Body Paaren: $(G_1, B_1), (G_2, B_2), \ldots, (G_n, B_n)$. Die Guards, $G_1, G_2, \ldots, G_n$ selbst bestehen wiederum aus Aufrufen beliebiger Parlogprädikate und werden parallel evaluiert. Wenn ein Guard G_i erfolgreich ausgewertet ist, wird der Body B_i aufgerufen. Die Evaluierung aller anderen Guards wird gestoppt und die entsprechenden Bodies werden nie aufgerufen. Die Guards selbst dürfen keine Ausgabe für P erzeugen. Man kann daher in Parlog ein Prädikat definieren, welches eine

Liste in alle möglichen Paare von Unterlisten zerteilt und eines dieser Paare bei einem einzelnen Aufruf zurückliefert. Da die Auswertung der Guards nicht festgelegt ist, kann das Ergebnis bei wiederholten Aufrufen unterschiedlich ausfallen, muß aber nicht.

In Parlog kommunizieren nebenläufige Prozesse durch gemeinsame Datenstrukturen. Man kann zum Beispiel zwei Prädikate aufrufen, die beide eine Variable `X` als Parameter haben. Ein Prädikat muß als Erzeuger von Werten für `X` gekennzeichnet werden, das andere als Verbraucher. Wenn der Verbraucher Werte für `X` verwenden muß, wartet er, bis der Erzeuger die Werte zur Verfügung stellt. Welche Rolle jeder der beiden spielt, steht in der Definition der Prädikate. Parlogprädikate lassen sich im Gegensatz zu Prolog bezüglich des Ein- Ausgabeverhaltens nicht variabel einsetzen. Allerdings erhält man dadurch eine elegante Steuerung der Nebenläufigkeit.

5.2.3 Vergleich von Prolog und Parlog

Parlog eignet sich für Anwendungen, in denen nebenläufige Prozesse ihr Verhalten durch Daten und Ereignisse von aussen steuern lassen und sich die Suche, wie ein Prozess auf die Daten und Ereignisse reagieren soll, durch die beschränkten Möglichkeiten von Parlog-Guards beschreiben lässt. Prolog anderseits ist besonders geeignet für wissensbasierte Anwendungen, in denen das multimodale Verhalten und die Möglichkeit, Metainterpreter zu schreiben, deklaratives Wissen vielfältig einzusetzen, und Suche durch Backtracking nötig sind.

5.2.4 Parlog+Prolog für Multi-Agenten-Systeme

Parlog+Prolog ist eine Sprache, die Parlog und Prolog koppelt und beider Vorteile vereint[13]. Die zusammengefügten Sprachen haben die gleichen Datenstrukturen und eine Prädikatdefinition in einer Sprache kann eine Prädikatdefinition in der anderen aufrufen. Die Koppelung ist aber verhältnismäßig lose. Insbesondere wenn Prolog backtrackt, und neue Werte für eine Variable liefert, wird dies nicht auf die Parlog-Seite propagiert.

Die Zusammensetzung von Parlog und Prolog ist für die Implementierung von Multi-Agenten-Systemen äußerst nützlich. Die lose Koppelung ist kein Hindernis. Man kann Parlog verwenden, um ein System zu beschreiben, in dem mehrere Agenten verteilt und parallel laufen. Die Kommunikation zwischen Agenten läßt sich dann durch gemeinsame Variablen beschreiben. Das Backtracking von Prolog wird im Verhalten zwischen Agenten nicht gebraucht; wenn ein Agent einem anderen Werte gibt, die später nicht mehr gelten sollen, muß das explizit zwischen den Agenten verhandelt werden. Um das prinzipielle Vorgehen des eigentlichen Agenten zu beschreiben, ist Prolog jedoch sehr nützlich. Automatisches Suchverfahren, leichte Metainterpretation und ein allgemeiner deklarativer Ansatz für wissenbasiertes Programmieren ist das, was man dazu braucht. Dies ist zumindest die Erfahrung, die im KIK-Projekt gemacht wurde. Dort wurde Parlog+Prolog verwendet, um ein System von kooperierenden Agenten zu codieren. Zu Demonstrationszwecken hat es sich auf den Anwendungsgebieten Vekehrswesen und automatische Terminvereinbarung bewährt, [1].

Literatur

[1] A. Lux, Entwicklung kooperierender Anwendungen unter MEKKA. In diesem Band, 1992

[13] Weitere Auskünfte sind von K.L. Clarke oder F.G. M^{c}Cabe, Department of Computing, Imperial College, London, zu erhalten.

Literatur

[BG88] A. Bond and L. Gasser. *Readings in Distributed Artificial Intelligence.* Morgan Kaufmann, Los Angeles, CA, 1988.

[BKMP92] M. Buchheit, N. Kuhn, J. P. Müller, and M. Pischel. Mars: Modelling a multiagent scenario for shipping companies. In *Proceedings of ESS92, Dresden.* Forthcoming, 1992.

[BM91] H. J. Bürckert and J. Müller. Ratman: Rational agents testbed for multi agent networks. In *Decentralized A.I. 2.* North Holland, 1991.

[Boc87] Siegfried Bocionek. Dynamic flavors. Technischer Bericht TUM-I8708, TU München, Institut für Informatik, Juni 1987.

[Boc89] Siegfried Bocionek. *Modularisierung als Grundkonzept zur Entwicklung systemunterstützer Programmierumgebungen für parallele Regelprogramme.* PhD thesis, TU München, Institut für Informatik, Mai 1989.

[Bon89] A. H. Bond. The cooperation of experts in engineering design. In *Distributed Artificial Intelligence, Volume II*, pages 463–486. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1989.

[Bou92] C. Boutilier. A logic for revision and subjunctive queries. In *Proceedings of AAAI-92, San Jose, CA*, pages 609–615. AAAI Press, Menlo Park, CA, 1992.

[Bus92] S. Bussmann. Simulation Environment for Multi-Agent Worlds. Technical Report DFKI Document D-92-01, DFKI, Saarbrücken, January 1992.

[CDL87] D. D. Corkill, E. H. Durfee, and V. R. Lesser. Coherent cooperation among communicating problem solvers. In *IEEE Transactions on Computers*, C-36, pages 1275–1291, 1987.

[CHRS+88] S. Cammarata, F. Hayes-Roth, R. Steeb, P. Thorndyke, and R. Wesson. Architectures for distributed air-traffic control. In A. H. Bond and L. Gasser, editors, *Readings in Distributed Artificial Intelligence*, pages 102–105. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1988.

[CL87] D. D. Corkill and V. R. Lesser. Distributed problem solving. In S. C. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 245–251. John Wiley and Sons, New York, 1987.

[Dav90] E. Davis. *Representations of Commonsense Knowledge.* Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990.

[DL89] E. H. Durfee and V. R. Lesser. Negotiating task decomposition and allocation using partial global planning. In *Distributed Artificial Intelligence, Volume II*, pages 229–244, San Mateo, CA, 1989. Morgan Kaufmann Publishers, Inc.

[DM80] J. Doyle and J. McDermott. Non-monotonic logic i. *Artificial Intelligence*, 13:27–40, 1980.

[DM90] Y. Demazeau and J.-P. Müller, editors. *Decentralized A.I.* North-Holland, 1990.

[DM91] Y. Demazeau and J.-P. Müller. *Decentralized A.I. 2.* North Holland, 1991.

[DS83] R. Davis and R.G. Smith. Negotiation as a metaphor for distributed problem solving. In *Artificial Intelligence, 20(1)*, pages 63–109, 1983.

[FH91] R. Fagin and J. Y. Halpern. Uncertainty, belief, and probability. *Computational Intelligence*, 7:160–173, 1991.

[FH92] R. Fagin and J. Y. Halpern. Two views of belief. *Artificial Intelligence*, 54(3):275–317, April 1992.

[FKMM92] K. Fischer, N. Kuhn, H. J. Müller, and J. P. Müller. Sophisticated and distributed: The transportation domain. Submitted for CAIA-93, 1992.

[Fre92] G. Frege. Über Sinn und Bedeutung. *Zeitschrift für Philosophie und Philosophische Kritik*, 100:25–50, 1892.

[FW92] K. Fischer and H. M. Windisch. Magsy- ein regelbasiertes multi-agentensystem, 1992.

[Get63] P. Gettier. Is justified true belief knowledge? Analysis, 1963.

[GH89] L. Gasser and M.N. Huhns. *Distributed Artificial Intelligence, Volume II.* Research Notes in Artificial Intelligence. Morgan Kaufmann, San Mateo, CA, 1989.

[Hal86] J. Y. Halpern. Reasoning about knowledge: an overview. In *Proceedings of TARK'86*, pages 1–15, 1986.

[Hew85] C. E. Hewitt. The challenge of open systems. *Byte*, 4(10), 1985.

[Hew86] C. Hewitt. Offices are open systems. *ACM Trans. on Office Information Systems*, 4:271–286, 1986.

[Hin62] J. Hintikka, editor. *Knowledge and Belief.* Cornell University Press, 1962.

[HM90] J. Y. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, 37(3):549–587, 1990.

[HM92] J. Y. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54:319–379, 1992.

[KMM92] N. Kuhn, H. J. Müller, and J. P. Müller. Task decomposition in dynamic agent societies, 1992.

[Kon85] K. Konolige. Belief and incompleteness. In J. R. Hobbs and R. C. Moore, editors, *Formal Theories of the Commonsense World*, pages 359–403. Ablex Publishing Corporation, 1985.

[KR87] R. Krickhahn and B. Radig. *Die Wissensrepräsentationssprache OPS-5.* Vieweg und Sohn, Braunschweig/Wiesbaden, 1987.

[Kri63] S. Kripke. Semantic analysis of modal logics, 1963.

[KvM91] T. Kreifelts and F. v. Martial. A negotiation framework for autonomous agents. In Y. Demazeau and J.-P. Müller, editors, *Decentralized A.I. 2.* North-Holland, 1991.

[Lak86] G. Lakemeyer. Steps towards a first-order logic of explicit and implicit belief. In *Proceedings of TARK'86*, pages 325–339, 1986.

[Lev84] H. J. Levesque. A logic of implicit and explicit belief. In *Proceedings of AAAI-84.* Austin, TX, 1984.

[Lev90] H. J. Levesque. All i know: A study in autoepistemic logic. *Artificial Intelligence*, 42:263–309, 1990.

[LM92] A. Laux and J. P. Müller. On the representation of knowledge and belief in multiagent systems. In *Proc. of 3rd. Workshop on Belief Representation and Agent Architectures*, Durham, UK, 1992. University of Durham, Dep. of Computer Science.

[Lou87] R. P. Loui. *Theory and Computation of Uncertain Inference and Decision.* PhD thesis, Department of Computer Science, University of Rochester, Rochester, NY, 1987.

[McC80] J. McCarthy. Circumscription–a form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–40, 1980.

[Moo80] R. C. Moore. Reasoning about knowledge and action. Technical Report Technical Note 191, SRI International, 1980.

[Moo83] R. C. Moore. Semantical considerations on non-monotonic logic. In *Proceedings of IJCAI-83*, pages 272–279, 1983.

[Moo85] R. C. Moore. A formal theory of knowledge and action. In J. R. Hobbs and R. C. Moore, editors, *Formal Theories of the Commonsense World*, pages 319–358. Ablex Publishing Corporation, 1985.

[Mos88] Y. Moses. Resource-bounded knowledge and belief. In M. Y. Vardi, editor, *Proc. of the Second TARK*. Morgan-Kaufmann, Inc., 1988.

[MP92] J. P. Müller and M. Pischel. Knowledge and belief of cooperating agents, September 1992.

[MPS92] J. P. Müller, M. Pischel, and A. Schroth. MARS: Eine Multiagenten-Testumgebung mit Graphischer Benutzerschnittstelle. Internal Working Paper, 1992.

[Rei80] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[S. 92] S. Bussmann and H.J. Müller. A Negotiation Framework for Cooperating Agents. Proc. of the 2nd Workshop on Cooperating Knowledge Based Systems, September 1992.

[Smi80] R.G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. In *IEEE Transaction on Computers*, volume C-29, pages 1104–1113, 1980.

[ZR89] G. Zlotkin and J. S. Rosenschein. Negotiation and task sharing among autonomous agents in cooperative domains. In *Proc. of the Eleventh IJCAI*, pages 912–917. Detroit, Michigan, August 1989.

[ZR92] G. Zlotkin and J. S. Rosenschein. A domain theory for task oriented negotiation. In A. Cesta, R. Conte, and M. Micheli, editors, *Pre-Proceedings of MAAMAW-92*, July 1992.

Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

DFKI
-Bibliothek-
PF 2080
D-6750 Kaiserslautern
FRG

DFKI Publikationen

Die folgenden DFKI Veröffentlichungen sowie die aktuelle Liste von allen bisher erschienenen Publikationen können von der oben angegebenen Adresse bezogen werden.
Die Berichte werden, wenn nicht anders gekennzeichnet, kostenlos abgegeben.

DFKI Publications

The following DFKI publications or the list of all publisched papers so far can be ordered from the above address.
The reports are distributed free of charge except if otherwise indicated.

DFKI Research Reports

RR-91-32
Rolf Backofen, Lutz Euler, Günther Görz:
Towards the Integration of Functions, Relations and Types in an AI Programming Language
14 pages

RR-91-33
Franz Baader, Klaus Schulz:
Unification in the Union of Disjoint Equational Theories: Combining Decision Procedures
33 pages

RR-91-34
Bernhard Nebel, Christer Bäckström:
On the Computational Complexity of Temporal Projection and some related Problems
35 pages

RR-91-35
Winfried Graf, Wolfgang Maaß: Constraint-basierte Verarbeitung graphischen Wissens
14 Seiten

RR-92-01
Werner Nutt: Unification in Monoidal Theories is Solving Linear Equations over Semirings
57 pages

RR-92-02
Andreas Dengel, Rainer Bleisinger, Rainer Hoch, Frank Hönes, Frank Fein, Michael Malburg:
Π_{ODA}: The Paper Interface to ODA
53 pages

RR-92-03
Harold Boley:
Extended Logic-plus-Functional Programming
28 pages

RR-92-04
John Nerbonne: Feature-Based Lexicons: An Example and a Comparison to DATR
15 pages

RR-92-05
Ansgar Bernardi, Christoph Klauck, Ralf Legleitner, Michael Schulte, Rainer Stark:
Feature based Integration of CAD and CAPP
19 pages

RR-92-06
Achim Schupetea: Main Topics of DAI: A Review
38 pages

RR-92-07
Michael Beetz:
Decision-theoretic Transformational Planning
22 pages

RR-92-08
Gabriele Merziger: Approaches to Abductive Reasoning - An Overview -
46 pages

RR-92-09
Winfried Graf, Markus A. Thies:
Perspektiven zur Kombination von automatischem Animationsdesign und planbasierter Hilfe
15 Seiten

RR-92-10
M. Bauer: An Interval-based Temporal Logic in a Multivalued Setting
17 pages

RR-92-11
Susane Biundo, Dietmar Dengler, Jana Koehler:
Deductive Planning and Plan Reuse in a Command Language Environment
13 pages

RR-92-13
Markus A. Thies, Frank Berger:
Planbasierte graphische Hilfe in objektorientierten Benutzungsoberflächen
13 Seiten

RR-92-14
Intelligent User Support in Graphical User Interfaces:
1. InCome: A System to Navigate through Interactions and Plans
   Thomas Fehrle, Markus A. Thies
2. Plan-Based Graphical Help in Object-Oriented User Interfaces
   Markus A. Thies, Frank Berger

22 pages

RR-92-15
Winfried Graf: Constraint-Based Graphical Layout of Multimodal Presentations
23 pages

RR-92-16
Jochen Heinsohn, Daniel Kudenko, Berhard Nebel, Hans-Jürgen Profitlich: An Empirical Analysis of Terminological Representation Systems
38 pages

RR-92-17
Hassan Aït-Kaci, Andreas Podelski, Gert Smolka: A Feature-based Constraint System for Logic Programming with Entailment
23 pages

RR-92-18
John Nerbonne: Constraint-Based Semantics
21 pages

RR-92-19
Ralf Legleitner, Ansgar Bernardi, Christoph Klauck: PIM: Planning In Manufacturing using Skeletal Plans and Features
17 pages

RR-92-20
John Nerbonne: Representing Grammar, Meaning and Knowledge
18 pages

RR-92-21
Jörg-Peter Mohren, Jürgen Müller
Representing Spatial Relations (Part II) -The Geometrical Approach
25 pages

RR-92-22
Jörg Würtz: Unifying Cycles
24 pages

RR-92-23
Gert Smolka, Ralf Treinen:
Records for Logic Programming
38 pages

RR-92-24
Gabriele Schmidt: Knowledge Acquisition from Text in a Complex Domain
20 pages

RR-92-25
Franz Schmalhofer, Ralf Bergmann, Otto Kühn, Gabriele Schmidt: Using integrated knowledge acquisition to prepare sophisticated expert plans for their re-use in novel situations
12 pages

RR-92-26
Franz Schmalhofer, Thomas Reinartz, Bidjan Tschaitschian: Intelligent documentation as a catalyst for developing cooperative knowledge-based systems
16 pages

RR-92-27
Franz Schmalhofer, Jörg Thoben: The model-based construction of a case-oriented expert system
18 pages

RR-92-29
Zhaohur Wu, Ansgar Bernardi, Christoph Klauck: Skeletel Plans Reuse: A Restricted Conceptual Graph Classification Approach
13 pages

RR-92-30
Rolf Backofen, Gert Smolka
A Complete and Recursive Feature Theory
32 pages

RR-92-31
Wolfgang Wahlster
Automatic Design of Multimodal Presentations
17 pages

RR-92-33
Franz Baader:
Unification Theory
22 pages

RR-92-34
Philipp Hanschke:
Terminological Reasoning and Partial Inductive Definitions
23 pages

RR-92-35
Manfred Meyer:
Using Hierarchical Constraint Satisfaction for Lathe-Tool Selection in a CIM Environment
18 pages

RR-92-36
Franz Baader, Philipp Hanschke:
Extensions of Concept Languages for a Mechanical Engineering Application
15 pages

RR-92-37
Philipp Hanschke:
Specifying Role Interaction in Concept Languages
26 pages

RR-92-38
Philipp Hanschke, Manfred Meyer:
An Alternative to H-Subsumption Based on Terminological Reasoning
9 pages

RR-92-41
Andreas Lux: A Multi-Agent Approach towards Group Scheduling
32 pages

RR-92-42
John Nerbonne:
A Feature-Based Syntax/Semantics Interface
19 pages

RR-92-43
Christoph Klauck, Jakob Mauss: A Heuristic driven Parser for Attributed Node Labeled Graph Grammars and its Application to Feature Recognition in CIM
17 pages

RR-92-44
Thomas Rist, Elisabeth André: Incorporating Graphics Design and Realization into the Multimodal Presentation System WIP
15 pages

RR-92-45
Elisabeth André, Thomas Rist: The Design of Illustrated Documents as a Planning Task
21 pages

RR-92-46
Elisabeth André, Wolfgang Finkler, Winfried Graf, Thomas Rist, Anne Schauder, Wolfgang Wahlster: WIP: The Automatic Synthesis of Multimodal Presentations
19 pages

RR-92-51
Hans-Jürgen Bürckert, Werner Nutt:
On Abduction and Answer Generation through Constrained Resolution
20 pages

DFKI Technical Memos

TM-91-13
Knut Hinkelmann:
Forward Logic Evaluation: Developing a Compiler from a Partially Evaluated Meta Interpreter
16 pages

TM-91-14
Rainer Bleisinger, Rainer Hoch, Andreas Dengel:
ODA-based modeling for document analysis
14 pages

TM-91-15
Stefan Bussmann: Prototypical Concept Formation An Alternative Approach to Knowledge Representation
28 pages

TM-92-01
Lijuan Zhang:
Entwurf und Implementierung eines Compilers zur Transformation von Werkstückrepräsentationen
34 Seiten

TM-92-02
Achim Schupeta: Organizing Communication and Introspection in a Multi-Agent Blocksworld
32 pages

TM-92-03
Mona Singh
A Cognitiv Analysis of Event Structure
21 pages

TM-92-04
Jürgen Müller, Jörg Müller, Markus Pischel, Ralf Scheidhauer:
On the Representation of Temporal Knowledge
61 pages

TM-92-05
Franz Schmalhofer, Christoph Globig, Jörg Thoben
The refitting of plans by a human expert
10 pages

TM-92-06
Otto Kühn, Franz Schmalhofer: Hierarchical skeletal plan refinement: Task- and inference structures
14 pages

TM-92-08
Anne Kilger: Realization of Tree Adjoining Grammars with Unification
27pages

DFKI Documents

D-92-04
Judith Klein, Ludwig Dickmann: DiTo-Datenbank - Datendokumentation zu Verbrektion und Koordination
55 Seiten

D-92-06
Hans Werner Höper: Systematik zur Beschreibung von Werkstücken in der Terminologie der Featuresprache
392 Seiten

D-92-07
Susanne Biundo, Franz Schmalhofer (Eds.): Proceedings of the DFKI Workshop on Planning
65 pages

D-92-08
Jochen Heinsohn, Bernhard Hollunder (Eds.): DFKI Workshop on Taxonomic Reasoning Proceedings
56 pages

D-92-09
Gernod P. Laufkötter: Implementierungsmöglichkeiten der integrativen Wissensakquisitionsmethode des ARC-TEC-Projektes
86 Seiten

D-92-10
Jakob Mauss: Ein heuristisch gesteuerter Chart-Parser für attributierte Graph-Grammatiken
87 Seiten

D-92-11
Kerstin Becker: Möglichkeiten der Wissensmodel-lierung für technische Diagnose-Expertensysteme
92 Seiten

D-92-12
Otto Kühn, Franz Schmalhofer, Gabriele Schmidt: Integrated Knowledge Acquisition for Lathe Production Planning: a Picture Gallery (Integrierte Wissensakquisition zur Fertigungsplanung für Drehteile: eine Bildergalerie)
27 pages

D-92-13
Holger Peine: An Investigation of the Applicability of Terminological Reasoning to Application-Independent Software-Analysis
55 pages

D-92-14
Johannes Schwagereit: Integration von Graph-Grammatiken und Taxonomien zur Repräsentation von Features in CIM
98 Seiten

D-92-15
DFKI Wissenschaftlich-Technischer Jahresbericht 1991
130 Seiten

D-92-16
Judith Engelkamp (Hrsg.): Verzeichnis von Softwarekomponenten für natürlichsprachliche Systeme
189 Seiten

D-92-17
Elisabeth André, Robin Cohen, Winfried Graf, Bob Kass, Cécile Paris, Wolfgang Wahlster (Eds.): UM92: Third International Workshop on User Modeling, Proceedings
254 pages
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

D-92-18
Klaus Becker: Verfahren der automatisierten Diagnose technischer Systeme
109 Seiten

D-92-19
Stefan Dittrich, Rainer Hoch: Automatische, Deskriptor-basierte Unterstützung der Dokumentanalyse zur Fokussierung und Klassifizierung von Geschäftsbriefen
107 Seiten

D-92-21
Anne Schauder: Incremental Syntactic Generation of Natural Language with Tree Adjoining Grammars
57 pages

D-92-23
Michael Herfert: Parsen und Generieren der Prolog-artigen Syntax von RELFUN
51 Seiten

D-92-24
Jürgen Müller, Donald Steiner (Hrsg.): Kooperierende Agenten
78 Seiten

D-92-25
Martin Buchheit: Klassische Kommunikations- und Koordinationsmodelle
31 Seiten

D-92-26
Enno Tolzmann:
Realisierung eines Werkzeugauswahlmoduls mit Hilfe des Constraint-Systems CONTAX
28 Seiten

D-92-27
Martin Harm, Knut Hinkelmann, Thomas Labisch: Integrating Top-down and Bottom-up Reasoning in COLAB
40 pages