Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Document
D-93-06

Beiträge zum Gründungsworkshop der Fachgruppe Verteilte Künstliche Intelligenz Saarbrücken 29.-30. April 1993

Jürgen Müller (Hrsg.)

April 1993

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, SEMA Group, 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
- Computer Linguistics
- Programming Systems
- Deduction and Multiagent Systems
- Document Analysis and Office Automation.

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.

Friedrich J. Wendl
Director

Beiträge zum Gründungsworkshop der Fachgruppe Verteilte Künstliche Intelligenz Saarbrücken 29.-30. April 1993

Jürgen Müller (Hrsg.)

DFKI-D-93-06

Diese Arbeit wurde finanziell unterstützt durch das Bundesministerium für Forschung und Technologie (FKZ ITW-9104 9).

Inhaltsverzeichnis

Modelle interagierender Agenten:

Verteiltes Planen:

Kooperative Systeme in der Fertigung:

Tools und Experimentierumgebungen:

Theoretische Grundlagen:

Mensch-Maschine Arbeiten:

Anwendungen von Multiagentensystemen:

Föderative Systeme:

Eine Hybride Architektur für Mehragenten-Systeme

Afsaneh Haddadi, Birgit Burmeister
Daimler-Benz AG, Forschung Systemtechnik
Alt-Moabit 91 b, W–1000 Berlin 21
afsaneh@DBresearch–berlin.de

Übersicht

Gegenwärtig wird in der Verteilten Künstlichen Intelligenz heftig diskutiert, ob ein Agent eher als reflektiven oder als reaktives System aufgefaßt werden sollte. In der reflektiven Sichtweise muß der Agent explizit über "mentale" Zustände (Wissen, Annahmen, Ziele) verfügen um über Ziele und Pläne räsonieren zu können. In der reaktiven Sichtweise dagegen bildet sich das angemessene Verhalten aus einem einfachen Reiz-Antwort Schema heraus. In jüngster Zeit werden sogenannte hybride Agenten-Architekturen diskutiert, in denen sowohl reaktives als auch reflektives Verhalten modelliert wird. Die meisten der vorgestellten Systeme betreffen aber einen einzelne Agenten und sein Verhalten in dynamischen Umgebungen. In diesem Beitrag wird eine hybride Architektur vorgestellt, die zusätzlich berücksichtigt, daß im Umfeld eines Agenten weitere Agenten mit weitgehend unvorhersehbarem Verhalten existieren. Diese Architektur ist im Rahmen des COSY Projekts entstanden.

1 Einleitung

Diskussionen in der DAI laufen derzeit immer wieder auf die Frage hinaus, ob ein Agent eher eine reflektive oder reaktive Architektur haben sollte. Anhänger der "Reflektiven Schule" vertreten die Auffassung, daß die einen Agenten auszeichnende Autonomie und Rationalität sinnvoll nur dadurch charakterisiert werden können, daß der Agent explizit über mentale Zustände (Wissen, Absichten, Ziele) verfügt. Gemäß der "Reaktiven Schule" bildet sich dagegen sinnvolles Verhalten eines Agenten aus einem einfachen Reiz-Antwort Schema heraus. Die Wahrheit liegt wohl, wie immer, in der Mitte: Gefragt sind hybride Systeme, die über ein ausgewogenes Maß sowohl reflektiver als auch reaktiver Fähigkeiten verfügen.

Als Vorläufer der reflektiven Architekturen können klassische Planungssysteme wie STRIPS [Fikes/Nilsson 71] angesehen werden. Bei diesen Systemen wird ein Plan zur Erreichung eines gewünschten Zielzustands zunächst detailliert ausgearbeitet und dann erst ausgeführt. Klassische Planungssysteme setzen eine statische, sich außer durch die Aktionen des Planers nicht verändernde, Umwelt voraus (*"static world assumption"* [Sanborn/Hendler 88]) und sind daher nicht für komplexe hochdynamische Umgebungen geeignet. In solchen Umgebungen kann durch die Überbetonung der Planungsphase nicht in angemessener Zeit auf Veränderungen der Umgebung reagiert werden. Planungssysteme in dynamischen Umgebungen müssen in der Lage sein, den Planungsprozeß zu unterbrechen, um einen Kompromiß zwischen der benötigten Planungs- und Ausführungszeit zu finden, oder einen Plan zu überarbeiten sowie den Planungsprozeß vollständig zu beenden und einen gänzlich neuen Plan zur Erreichung eines neuen Zielzustandes zu erarbeiten [Georgeff/Lansky 87].

Es wurde eine Reihe von Systemen entwickelt, die die oben genannten Problem angehen, z.B. *Incremental Planning* [Durfee/Lesser 86], in dem Planung und Ausführung miteinander verwoben werden, die Ausführung von Aktionen überwacht und mit der so gewonnenen Information ein Plan modifiziert wird. Obwohl diese Systeme für dynamische Umgebungen besser geeignet sind als die klassischen Planer, stoßen sie an ihre Grenzen in Domänen, in denen ein ständiges Neu-Planen erforderlich ist.

Zu den reaktiven Architekturen gehören u.a. PENGI [Agre/Chapman 87], die *Subsumption Architecture* [Brooks 86] und die *Situated Automata* [Kaelbling 87]. Der Vorteil dieser Systeme liegt in der direkten Kopplung der Sensoren und Effektoren, die eine dezentrale Steuerung und die dynamische Interaktion mit der Umgebung erlaubt [Maes 90]. Durch ihr einfaches Reiz-Antwort-Verhalten sind reaktive Systeme fähig, schnell auf Veränderungen der Umgebung zu reagieren. Allerdings basieren auch diese Systeme auf strengen Annahmen, nämlich daß alle Aktionen eines Agenten situationsbedingt sind, daß alle Ziele oder Absichten des Agenten in eine einfache Rangordnung eingeordnet werden können und daß zur Laufzeit des Systems nicht über Ziele räsoniert werden muß [Ferguson 92]. Es ist daher fraglich, ob die reaktiven Systeme fähig sind, anspruchsvolle Aufgaben auszuführen.

In letzter Zeit wird versucht, diese beiden extremen Ansätze zu verbinden. Sogenannte hybride Architekturen erlauben es dem Agenten, einerseits über die Umgebung und sich selbst zu reflektieren, andererseits auf unvorhergesehene Ereignisse in der Umgebung in angemessener Zeit zu reagieren. Beispiele für solche Systeme sind das *Procedural Reasoning System* (PRS) [Georgeff/Ingrand 88], die "*architecture for a rational agent*", im folgenden kurz ARA genannt [Bratman/Israel/Pollack 88] und *Touring Machines* [Ferguson 92]. PRS und ARA werden als BDI-(*Belief/Desire/Intention-*)Architekturen bezeichnet, da Annahmen, Wünsche und Absichten des Agenten explizit repräsentiert werden.

Diese hybriden Systeme sind in der "*Centralized AI*" angesiedelt. Sie sollen Agenten befähigen, in unvorhergesehenen Situationen, die sich aus der Dynamik des Umfelds des Agenten ergeben, zielgerichtet zu agieren. Im Kontext der "*Distributed AI*" wird die Dynamik durch die Anwesenheit anderer Agenten verstärkt.

In diesem Artikel wird eine Architektur vorgestellt, die sowohl reaktives als auch reflektives Verhalten, insbesondere bei der Interaktion mit anderen Agenten erlaubt. Diese Architektur ist eine BDI Architektur, die in einigen ihren Strukturen mit PRS und ARA vergleichbar ist.

Die vorgestellte hybride Architektur beruht auf dem dem COSY Projekt zugrunde liegenden Agenten Modell. Dieses wird, soweit es zum Verständnis der Architektur nötig ist, im nächsten Abschnitt beschrieben. Der darauf folgende Abschnitt enthält eine Darstellung von Struktur und Funktion einer "Reasoning, Deciding and Reacting" Komponente. Der letzte Abschnitt enthält einen Zusammenfassung und einen Ausblick auf nach ungelöste Frage in Zusammenhang mit der hybriden Architektur.

2 COSY Agentenmodell

Das Agentenmodell beschreibt einen Agenten durch seine Absichten, sein Verhalten (Handlungen und Wahrnehmung) sowie durch die Ressourcen, die er zur Erfüllung seiner Absichten und Durchführung seines Verhaltens benötigt [Sundermeyer 90].

Das in COSY entwickelte domänenunabhängige Verfahren zum kooperativen Problemlösen basiert auf Skripten und Kooperationsprotokollen, bei dem ein Agent, abhängig von seiner Wahrnehmung und seinen Absichten, ein der jeweiligen Situation angemessenes Verhaltensschema zu verfolgen versucht [Burmeister/Sundermeyer 92].

Aus dem Agentenmodell wurde eine modulare Agenten-Architektur abgeleitet, s. Abb. 1. Ein COSY Agent besteht aus Modulen ACTUATORS, SENSORS, COMMUNICATION, INTENTION und COGNITION. Die Namen der erstgenannten Module sind weitgehend selbsterklärend. INTENTION steht für langfristige Ziele, Grundeinstellungen, Verantwortlichkeit etc., also das, was im ARA System von Bratman, Israel und Pollack "Desires" genannt wird. Das zentrale Modul COGNITION, welches für die Auswertung der Wahrnehmung und die Vorbereitung von Aktionen zuständig ist, ist wissensbasiert realisiert, vgl. Abb. 1. COGNITION besteht aus den Komponenten

- KB ('Knowledge Base')
- SEC ('Skript Execution Component')
- PEC ('Protocol Execution Component')
- RDRC ('Reasoning, Deciding and Reacting Component').

In der Wissensbasis ist u.a. das Wissen des Agenten über seine Verhaltensschemata abgelegt. Dies entspricht der "plan library" von ARA und den "knowledge areas" in PRS.

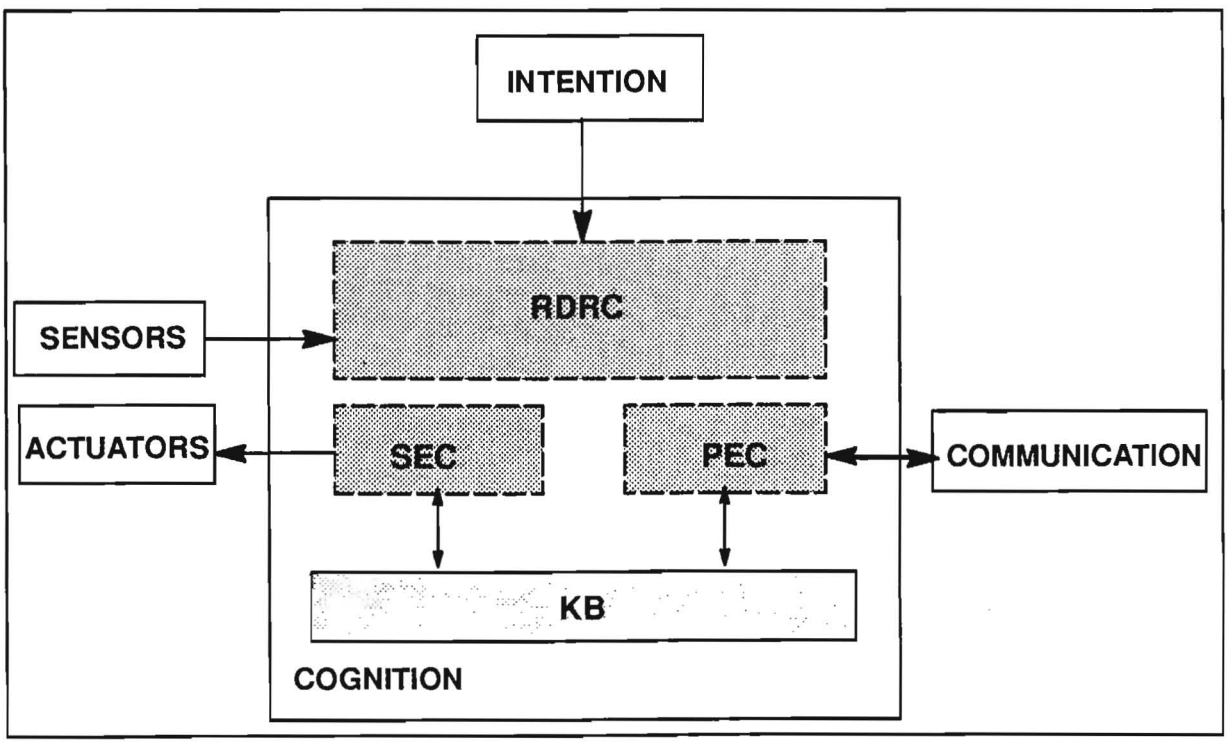


Abb. 1: COSY Agent Architektur

Skripte beschreiben einen prototypischen Verhaltensablauf. Sie enthalten eine Ausführungsvorschrift, die Aufrufe von anderen Skripten, Kooperationsprotokollen oder elementarem Verhalten enthalten kann. Elementares Verhalten wird als "acting" von ACTUATORS oder als "sending" von COMMUNICATION ausgeführt. Die COGNITION Komponente SEC sorgt für die korrekte Abwicklung eine Skriptes. Unter anderem delegiert sie die Ausführung von elementaren Handlungen an ACTUATORS und verweist Kooperationsprotokolle an PEC.

Kooperationsprotokolle beschreiben generische Dialoge [Burmeister/Haddadi/Sundermeyer 93]. Sie sind Skripten vergleichbar insofern sie Planmuster für die Interaktion von Agenten über Kommunikation darstellen. Die COGNITION Komponente PEC überwacht und steuert die Abwicklung der Kooperationsprotokolle; unter anderem bereitet sie Nachrichten zum Senden auf und wertet empfangene Nachrichten aus. Innerhalb eines Protokolls zu treffende Entscheidungen hinsichtlich der Reaktion auf empfangene Nachrichten ist nicht Aufgabe von PEC, sondern erfolgt, wie unten beschrieben, durch RDRC.

3 Reagieren und Reflektieren

Die Skript- und Protokoll-Verwaltung ist Aufgabe der COGNITION Komponente RDRC. Die Verwaltung beinhaltet die Auswahl von Skripten und Protokollen, Entscheidungen bei deren Verzweigungen, die flexible Behandlung von möglichen Unterbrechungen und Wiederaufsetzen nach Unterbrechungen sowie den eventuellen Abbruch eines Skripts.

3.1 Hintergrund

Zur Motivierung von Struktur und Funktion von RDRC soll zunächst auf eine Arbeit von Covrigaru und Lindsay [Covrigaru/Lindsay 91] eingegangen werden. Diese Autoren bezeichnen ein System als autonom "...if it includes a measure of complexity, interaction, movement, a variety of behaviours, robustness, differential responses to a variety of environmental conditions, selective attention and independent existence without detailed, knowledgeable intervention". Sie betonen darüberhinaus, daß "...a system with all these properties is usually perceived to be autonomous provided that in addition it was also seen to be goal-directed". Und konsequenterweise charakterisieren sie Autonomie als "...having the right kinds of goals and the ability to select goals from an existing set". Dies gibt Anlaß zu zwei Fragen, nämlich (1) Woher kommen die Ziele? und (2) Wie wählt ein Agent Ziele aus?

Die zweite Frage ist letztlich diejenige, die wir in diesem Artikel zu beantworten versuchen. Dies ist allerdings nur möglich, wenn zuvor die Frage nach dem Ursprung der Ziele beantwortet ist. Covrigaru und Lindsay klassifizieren Ziele nach ihrem Ursprung im Kontext eines isolierten Agenten. Wir übertragen ihre Klassifikation auf Multi-Agenten Szenarien. Ziele werden hinsichtlich zweier überlappender Dimensionen, nämlich *acquired/built-in* und *endogenous/exogenous* klassifiziert:

Built-in sind die dem Agenten per Entwurf zugewiesenen Ziele, die letztlich die Funktion und den Zweck des Agenten beschreiben.

Acquired sind diejenige Ziele, die ein Agent im Laufe des Wechselwirkung mit anderen Agenten annimmt (z. Bsp. als Resultat einer Aufforderung).

Endogenous sind *built-in* Ziele zusammen ihren Subzielen. Dies sind typischerweise die Ziele, die von klassischen Planern betrachtet werden.

Exogenous sind *acquired* Ziele zusammen mit Zielen, die ein Agent als Reaktion auf Umweltänderungen annimmt. Diese Art von Zielen sind damit typischerweise mit der Dynamik in Multi-Agenten Szenarien verknüpft.

In dem hier vorgestellten Agentenmodell werden diese Ziele durch perzeptives und kognitives Verhalten ermittelt. Perzeptives Verhalten ist zum einem die direkte Wahrnehmung der Umgebung und zum andern das Empfangen von Nachrichten von anderen Agenten.

Diese Verhaltensweisen sind passiv, d.h. sie laufen nicht gemäß Skripten ob, sondern werden zyklisch ausgeführt. Ihnen sind Skripte zugeordnet, die anwendbar werden, wenn gewisse Bedingungen erfüllt sind.

3.2 Struktur von RDRC

Abbildung 2. gibt das COGNITION Modul RDRC schematisch wieder. RDRC verfügt über Schnittstellen zu INTENTION sowie zu den Komponenten SEC und PEC in COGNITION. Über die Schnittstelle mit INTENTION greift RDRC auf die dort (anwendungspezifisch) modellierten langfristigen Ziele und Grundeinstellungen zu.

RDRC besteht aus einer Agenda, einer Intentions-Struktur und Prozeduren zum "reasoning, deciding and reacting". Die Agenda und die Intentions-Struktur beinhalten diejenige Skripte, die für die Prozeduren als Daten benötigt werden.

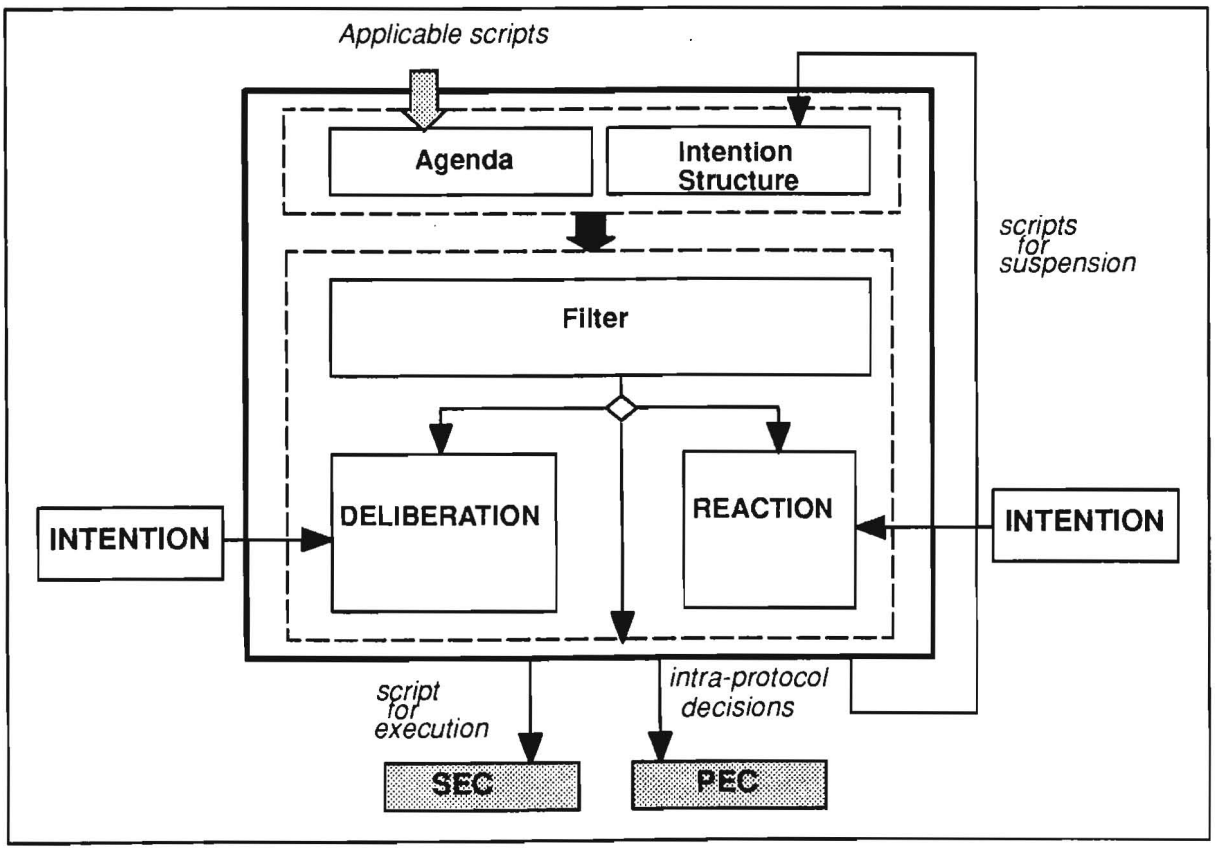


Abb. 2: RDRC

3.2.1 Agenda und Intentions-Struktur

Die Agenda enthält all diejenige Skripte, die in einer gegebenen Situation anwendbar sind. Diese Skripte gelangen auf die Agenda aufgrund von Wahrnehmung des Agenten (d.h. über SENSORS oder COMMUNICATION), oder von Änderungen langfristiger Ziele (d.h. über INTENTION). Skripte auf der Agenda sind danach unterschieden, welchem Ziel sie zugeordnet sind und durch welche wahrgenommenen Ereignisse sie anwendbar werden:

Reactive: heißen diejenigen Skripte, die unmittelbar auszuführen sind und jene, die die Ausführung eines gegenwärtig bearbeiteten Skripts tangieren.

Deliberative: heißen diejenige Skripte, über deren Ausführung der Agent entscheiden kann, und die er gegenüber dem gegenwärtig bearbeiteten Skript abwägen muß.

Goal-directed: heißen diejenige Skripte, die langfristigen Zielen (*built-in* oder *acquired*) zugeordnet sind.

Die Agenda ist gemäß dieser Klassifikation unterteilt, und Skripte werden, sofern sie anwendbar werden, in das entsprechende "Fach" abgelegt.

Während die Agenda diejenigen Skripte enthält, die in der gegenwärtiger Situation anwendbar sind, enthält die Intentions-Struktur (der Name wurde der "intention structure" des PRS entlehnt) Skripte, die der Agent zu einem gegebenen Zeitpunkt betrachtet. Dementsprechend werden in der Intentions-Struktur zwei Arten von Listen geführt:

- das Skript, welches gegenwärtig ausgeführt wird, einschließlich aller seiner Subskripte.
- diejenigen Skripte, deren Ausführung unterbrochen wurde. Diese Unterbrechung kann dadurch bedingt sein, daß (i) eine unmittelbare Antwort auf eine unvorhergesehene Situation durch ein *reactive* Skript nötig wurde, (ii) ein anderes Ziel aufgegriffen wird, oder (iii) die Resourcen zur Ausführung des Skripte gegenwärtig nicht verfügbar sind.

3.2.2 Filter, Reaktions und Deliberations Prozeduren

Filter, Reaktion und Deliberation sind die Prozeduren, die die Fähigkeit des Agenten zum angemessenen Reagieren und Räsonieren ausmachen. Die Idee ist die folgende: Zu jedem Zeitpunkt ist der Agent bestrebt ein bestimmtest Ziel zu erreichen. Dieses Ziel kann entweder ein *built-in* Ziel oder ein *acquired* Ziel sein. Einige Situationen sind nicht vorhersagbar und stehen nicht in Zusammenhang mit dem Plan zur Erreichung des angenommenen Zieles. Der Agent muß jedoch auf diese Situation reagieren ohne dabei sein ursprüngliches Zeit aus dem Auge zu verlieren. Anderseits kann es geschehen, daß der Agent während der Verfolgung eines Zieles auf andere Ziele aufmerksam wird. In diesem Fall muß darüber räsoniert werden, ob vorübergehend oder endgültig ein anderes Ziel verfolgt werden soll. Falls der Agent sich für ein neues Ziel entscheidet, muß er auch entscheiden, was mit dem ursprünglichen Ziel zu geschehen ist.

Diese Überwachung und Steuerung geschieht durch die Prozeduren Filter, Reaktion und Deliberation in RDRC. In jedem Interpretations-Zyklus ruft RDRC zunächst die Filter Prozedur auf. Diese filtert all diejenigen Skripte aus, die als nächstes ausgeführt werden sollten und trennt sie von denjenigen, die weiteres räsonieren zulassen.

Dementsprechend teilt sich der Kontrollfluß zum einen in Richtung der Reaktions Prozedur (im Falle des Vorliegens eines *reactive* oder *goal-directed* Skripts) und zum anderen in Richtung Deliberations Prozedur (im Falle des Vorliegens von Skripten, über deren Ausführung entschieden werden muß). Falls keine derartige Skripte vorliegen, ist SEC erlaubt, das gegenwärtig bearbeitete Skript weiter auszuführen. In Abhängigkeit von den Entscheidungen der Deliberations Prozedur aktualisiert RDRC die Intentions-Struktur durch neu ausgewählte Skripte.

Die Filter Prozedur

Zentrale Aufgabe des Filters (die der gleichnamigen Prozedur in ARA entspricht) ist die Überprüfung, ob auf der Agenda irgendwelche *reactive* oder *goal-directed* Skripte vorliegen und gegebenenfalls sofort die Reaktions Prozedur mit der Bearbeitung dieser Skripte zu beauftragen. Diese Überprüfung geschieht zunächst hinsichtlich *reactive* Skripte bevor überhaupt *goal-directed* Skripte in Erwägung gezogen werden. Falls weder *reactive* noch *goal-directed* Skripte auf der Agenda liegen, wird SEC erlaubt, mit der Ausführung eines geöffneten Skripts fortzufahren. Eine weitere Aufgabe der Filter Prozedur ist es, die in der Intentions-Struktur anstehenden Skripte zusammen mit gegenwärtig vorhandenen *deliberation* Skripten auf der Agenda zur nächsten Entscheidungsrunde der Deliberations Prozedur zur Verführung zu stellen.

Die Reaktions Prozedur

Die Reaktions Prozedur bestimmt aus den von der Filter Prozedur ermittelten *reactive* und *goal-directed* Skripten die dringendsten oder angemessensten zur Ausführung aus. In einer gegebenen Situation kann durchaus mehr als nur ein *reactive* oder *goal-directed* Skript auf der Agenda stehen. Dies kann daran liegen, daß entweder mehr als ein Skript geeignet ist, ein Ziel zu erreichen, oder daß auf unterschiedliche Ereignisse in der Umgebung reagiert werden muß. In diesen Fällen greift die Reaktions Prozedur auf das INTENTION Modul zu in dem die Prioritäten solcher Ziele bzw. Ereignisse festgelegt ist. Die Reaktions Prozedur wählt dementsprechend ein Skript aus und leitet es zur Ausführung an SEC weiter.

Die Deliberation Prozedur

Skripte über deren Auswahl weiter zu entscheiden ist, werden an die Deliberations Prozedur übergeben. Diese Prozedur ist somit für Meta-Entscheidungen zuständig. Diese Entscheidungen beruhen auf Kosten-Nutzen ("utility") Bewertungen und sind damit stark applikationsspezifisch. Die Aufgabe von Deliberation ist vergleichbar mit den nicht näher spezifizieren "meta-level knowledge areas" des PRS.

Die Deliberations Prozedur enthält applikationsspezifische Teile die auf deklarativ dargestellte langfristige Ziele, Grundeinstellungen und Rollen im Modul INTENTION zugreifen. Input für Deliberation sind *deliberative* Skripte zusammen mit den Ereignissen, die diese Skripte anwendbar machen sowie die in der Intentions-Struktur verwalteten Skripte. Output von Deliberation sind Skripte, die die auf der Intentions-Struktur verwalteten Listen aktualisieren sowie die Entscheidung darüber, ob das gegenwärtig verfolgte Skript fortgesetzt, vorübergehend unterbrochen oder ganz und gar beendet werden soll.

3.3 Reagieren und Räsonieren in Multi-Agenten Systeme

Wie bereits zuvor erläutert, wird die Wechselwirkung von Agenten in Form von Kooperationsprotokollen beschrieben. Basierend auf [Haddadi/Sundermeyer 93] wurden in [Burmeister/Haddadi/Sundermeyer 93] elementare Protokolle "Informing", "Querying", "Commanding" und "Offering" definiert, aus denen komplexere Protokolle wie "Demanding", "Requesting" und "Proposing" konstruiert wurden.

Protokolle sind verzweigte Strukturen, bei denen jede Verzweigung einer möglichen Entscheidung im Rahmen eines Kooperationsmusters entspricht. Zum Beispiel entsprechen die Zweige in "Requesting" dem Annehmen, dem Ablehnen und dem Vorschlag einer Alternative zu einer Aufforderung. Die PEC veranlaßt, falls sie bei der Abarbeitung eines Protokolls an eine solche Verzweigung gelangt, eine Entscheidung durch RDRC. Dazu überträgt sie die entsprechenden Protokolle in die Agenda und wartet auf die Entscheidung bevor sie mit der Abarbeitung des Protokolls fortfährt. Im folgenden wird beschrieben, wie diese Übertragung für die unterschiedlichen Protokolle geschieht.

Protokolle vom Typ "Offering" und "Proposing" drücken aus, daß der sendende Agent gewillt ist, zum *built-in* Ziel des Empfänger beizutragen. Andererseits werden Protokolle vom Typ "Requesting", "Demanding" und "Commanding" vom Sender verwendet um vom Empfänger Unterstützung für *built-in* Ziele zu erhalten. Dies ist dann andrerseits der Anlaß für *acquired* Ziele des empfangenen Agenten.

Die Protokolle "Offering" und "Proposing" unterscheiden sich dadurch, daß "Proposing" den Vorschlag von Alternativen durch den Empfänger zuläßt. Ein ähnlicher Unterschied besteht zwischen "Demanding" und "Requesting". Das "Commanding" Protokoll wird in Situationen verwendet, in denen die Ausführung eines Skripts durch den Empfänger der Nachricht dringend geboten ist. Somit gehört das zum "Commanding" gehörende Skript in den *reactive* Anteil der Agenda, während die zu "Offering", "Proposing", "Demanding" und "Requesting" gehörenden Skripte vom Typ *deliberative* sind.

Die Steuerung der innerhalb von Protokollen notwendigen Entscheidungen geschieht in der Deliberations Prozedur. Die getroffene Entscheidung wird von RDRC an PEC weitergegeben. In ähnlicher Weise werden gegebenenfalls Resultate des in der Reaktions Prozedur bearbeiteten "Commanding" Protokolls an PEC gegeben, wie z. Bsp. Berichte über den Erfolg eines Agenten bei der Ausführung einer Anordnung.

4 Zusammenfassung und Ausblick

Die in diesem Beitrag vorgestellte Architektur erfaßt sowohl reaktives als auch reflektives Verhalten, welches einem Agenten ermöglicht, nicht nur zielgerichtet vorzugehen, sondern dabei auch auf unvorhergesehene Ereignisse zu reagieren. Die Architektur ist Teilen den BDI Architektur von Bratman, Israel und Pollack sowie Georgeff und Mitarbeitern vergleichbar. Verglichen mit diesen Systemen berücksichtigt sie zusätzlich die dynamischen Aspekte in Multi-Agenten Szenarien.

In diesen Szenarien kann ein Agent veranlaßt werden, andere als seine ursprünglich verfolgten Ziele anzunehmen. Diese Ziele entstehen durch Wechselwirkung der Agenten über Kooperationsprotokolle. Bislang wurden von nur einfache Protokolle analysiert die sich entweder auf *built-in* Ziele ("Offering" und "Proposing"), auf *acquired* Ziele ("Demanding" und "Requesting") oder darauf beziehen, daß ein Agent unmittelbar zu handeln hat ("Commanding"). Zukünftige Arbeiten werden die Einordnung komplexerer Protokolle in das reflektiv/reaktiv Schema betreffen.

Die Behandlung von *built-in* und *acquired* Zielen durch die Prozedur Deliberation beziehen in der gegenwärtigen Form applikationsspezifisch. Wünschenswert wäre es, hier applikationsunabhängige Aspekte abzutrennen und sie an bekannte Verfahren der Entscheidungs- und Spieltheorie anzubinden. Dazu ist es nötig, ein besseres Verständnis für die Feinstruktur von langfristigen Zielen, Rollen, Grundeinstellungen und dergleichen zu gewinnen.

References

[Bratman/Israel/Pollack 88]
M. E. Bratman, D. J. Israel and M. E. Pollack,"Plans and Resource-bounded Practical Reasoning", Computational Intelligence, 4(4): 349–355, 1988.

[Brooks 86]
R .A. Brooks, "A robust Layered control system for a mobile robot", in: IEEE Journal of Robotic and Automation 2(1), 1986.

[Burmeister/Haddadi/Sundermeyer 93]
B. Burmeister, A. Haddadi, and K. Sundermeyer, "Generic Configurable Cooperation Protocols for Multi-Agent Systems", to be published.

[Burmeister/Sundermeyer 92]
B. Burmeister and K. Sundermeyer: "Cooperative Problem-Solving Guided by Intentions and Perception", in: E. Werner and Demazeau eds., Decentralized A.I. 3, North-Holland, 1992.

[Covrigaru/Lindsay 91]
A. A. Covrigaru and R. K. Lindsay, "Deterministic Autonomous Systems", in: AI Magazine 1991.

[Durfee/Lesser 86]
E. H. Durfee, V. R. Lesser: "Incremental planning to control a blackboard-based problem solver", in: Proc. 5th National Conf. on Artif. Intell., pp. 58–64, 1986.

[Georgeff/Ingrand 88]
M. P. Georgeff and F. F. Ingrand "Research on procedural reasoning systems", Final Report, Phase 1, AI Center, SRI International, Menlo park, Cal. 1988.

[Georgeff/Lansky 87]
M. P. Georgeff, A. L. Lansky:"Reactive Reasoning and Planning", in: Proc. AAAI 87, pp. 677–682.

[Ferguson 92]
I. A. Ferguson, "Touring Machines: An Architecture for Dynamic, Rational, Mobile Agents", PhD Thesis, Technical Report No. 273, University of Cambridge, Computer laboratory, 1992.

[Fikes/Nilsson 71]
R. E. Fikes, N.J. Nilsson, "STRIPS: a new approach to the application of theorem proving to problem solving to problem solving", in: Atrif. Intell. vol 2, pp. 189–208, 1971.

[Haddadi/Sundermeyer 93]
A. Haddadi, K. Sundermeyer, "Knowledge about other agents in heterogeneous dynamic domains", will be published in Proceedings of ICICIS '93.

[Kaelbling 87]
L. P. Kaelbling, "An architecture for intelligent reactive systems", in: M. P. Georgeff and A. L. Lansky, eds., Reasoning about Actions and Plans, Proc. 1986 Workshop, pp.395–410, Morgan Kaufmann, 1987.

[Maes 90]
P. Maes, "Situated agents can have goals", in: Designing Autonomous Agents: P. Maes eds., Theory and Practice from Biology to Engineering and Back, MIT Elsvier Publ., 1991.

[Sanborn/Hendler 88]
J. C. Sanborn and J. A. Hendler: "A model of reaction for planning in dynamic environments", in: Int. Journal of Artif. Intell in Engineering, 3(2), pp. 95–102, 1988.

[Sundermeyer 90]
K. Sundermeyer: "Modellierung von Szenarien kooperierender Akteure", in H.Marburger(ed.): GWAI-90, Springer, Informatik Fachberichte, pp.11-18.

INTERRAP:
eine Architektur zur Modellierung Flexibler Agenten

Jörg P. Müller*, Markus Pischel†
German Research Center for Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3, D-6600 Saarbrücken 11

Zusammenfassung

In diesem Artikel wird INTERRAP, eine Agentenarchitektur für Multiagentensysteme, vorgestellt. Die grundlegende Idee dabei liegt in der Kombination von sogenannten *Verhaltensmustern* mit plan-basierten Mechanismen. Dies ermöglicht es, die Vorteile reaktiver, verhaltensbasierter Architekturen einerseits und rationaler, plan-basierter Architekturen andererseits nutzbar zu machen. Verhaltensmuster ermöglichen Agenten, schnell und flexibel auf Änderungen in ihrer Umwelt zu reagieren. Die Fähigkeit zu planen erweist sich dagegen als notwendig, sollen komplexere Situationen gemeistert werden. Um die konzeptuellen Vorteile der Architektur effizient nutzbar zu machen, wird ein flexibler Kontrollmechanismus definiert, durch welchen die verhaltensbasierte und die planbasierte Komponente miteinander verzahnt sind.

Der INTERRAP Ansatz erweist sich aufgrund seiner Flexibilität als ein geeignetes Werkzeug zur Beschreibung dynamischer Agentengesellschaften. Diese Ansicht wird durch erste Erfahrungen untermauert, die wir anhand der Modellierung des Verladehofszenarios, einem Anwendungsbeispiel aus dem Bereich der Mobilen Roboter, gewonnen haben.

1 Einleitung

Das Forschungsgebiet der Verteilten Künstlichen Intelligenz (VKI) [BG88] beschäftigt sich mit der Frage nach Mechanismen und Gesetzmäßigkeiten, die es Gesellschaften intelligenter und autonomer Systeme (sog. Agenten) ermöglichen, ihr Wissen, ihre Ziele, ihre Fähigkeiten und ihre Pläne zu koordinieren (Multiagentensysteme); der Teilbereich des Verteilten Problemlösens erforscht, wie die Lösung eines bestimmten Problems innerhalb einer Menge von Problemlöseknoten verteilt werden kann. Unsere Forschung ist dabei im Bereich der Multiagentensysteme angesiedelt. Das heißt, daß die von uns betrachteten Agenten autonome entscheidungstragende Instanzen mit lokalen, möglicherweise unvollständigem Wissen sind. Sie sind an die Erreichung ihrer

*e-mail:jpm@dfki.uni-sb.de
†e-mail:mfp@dfki.uni-sb.de

lokalen Ziele gebunden, müssen aber aufgrund der Umwelt, in der sie existieren, mit anderen Agenten in Interaktion treten.

Das Forschungsgebiet *interagierender Roboter* hat sich in den letzten Jahren in zunehmendem Maße als fruchtbares Testbett für die Entwicklung und Auswertung von Konzepten in der VKI erwiesen. Dafür sind verschiedene Gründe maßgeblich:

- Die inhärente Verteilung von Kontrolle und Information in diesen Domänen.
- Die hohe Dynamik und Komplexität, die flexibles und intelligentes Verhalten der Agenten erfordern, und die Robotikdomänen zu anspruchsvollen Prüfsteinen für Architekturen und Methoden sowohl aus dem verhaltens- als auch aus dem planbasierten Lager innerhalb der VKI-Gemeinde gemacht haben.
- Die große Zahl an Interaktionsmöglichkeiten zwischen den Agenten einer Gesellschaft von Robotern.

Herkömmliche Ansätze im Bereich der Robotik waren vorwiegend mit den Problemen der mechanischen Kontrolle und der Kollisionsvermeidung bei der Wegeplanung beschäftigt (siehe auch [Bro86, FD90, ST92, Lat92]). Unterstützt durch die wachsende Rolle der Verteilten KI ist jedoch in den letzten Jahren die Vision von Robotern entstanden, die in einer dynamischen Umgebung aktiv kooperieren, um gewisse Ziele zu erreichen.

In diesem Artikel stellen wir die INTERRAP Agentenarchitektur vor und demonstrieren, wie sie zur Modellierung eines Multiagentensystems zur Simulation eines Verladehofszenarios verwendet werden kann: in diesem Szenario hat eine Gruppe von automatisierten Gabelstapler-Robotern die Aufgabe, einen LKW zu be- und entladen. Obwohl wir das Szenario im wesentlichen aus der Perspektive der Modellierung als Multiagentensystem betrachten, sind wesentliche Aspekte herkömmlicher Robotik zu beachten, wie beispielsweise Probleme der Perzeption und der Wegeplanung. Dabei wurden bei der Entwicklung des INTERRAP Modells wurde die Kriterien der Operationalisierbarkeit und der effizienten Implementierbarkeit in den Vordergrund gestellt.

Der Aufbau des Artikels ist wie folgt: in Abschnitt 2 geben wir eine kurze Einführung in die verwendete Terminologie. Abschnitt 3 beschreibt das konzeptionelle, Abschnitt 4 das funktionale INTERRAP Agentenmodell. In Abschnitt 5 wird beschrieben, wie die Verladehofdomäne unter Verwendung von INTERRAP als Multiagentenszenario modelliert wird.

2 Bemerkungen zur Terminologie

Begriffe wie "rational", "planbasiert", "verhaltensbasiert" oder "reaktiv", die wir zur Beschreibung von Agenten und Systemen verwenden, besitzen - bezüglich ihrer Verwendung innerhalb der VKI - zahlreiche Konnotationen. Daher möchten wir in diesem Abschnitt die Bedeutungen klären, die wir diesen Begriffen beimessen.

Zum einen sehen wir die Unterscheidung zwischen planbasierten und verhaltensbasierten Agenten als eine Unterscheidung auf der Ebene der Repräsentation an: planbasierte Agenten verfügen über explizite symbolische Repräsentationen ihrer Umwelt und ihrer eigenen Pläne und sind dadurch in der Lage, über die Struktur dieser Pläne Schlüsse zu ziehen [FHN71]. In diesem Sinne verwenden wir die Begriffe *rational* und *plan-basiert* als Synonyme. Verhaltensbasierte Agenten sind über sogenannte *Verhaltensmuster* beschrieben, die im Gegensatz zu Plänen keine sichtbare Struktur besitzen, sondern von außen gesehen *Black-Box*-Charakter haben [Suc87, Ste90]. Verhaltensmuster sind eng mit ihrer Ausführung verbunden, ja sie existieren sozusagen nur in Verbindung mit ihrer Ausführung. In dieser Hinsicht ähnelt die Einteilung in plan- und verhaltensbasierte Agenten dem Verhältnis zwischen dem deklarativen und dem prozeduralen Paradigma [Win75]: die Aktivitäten verhaltensbasierter Agenten werden prozedural repräsentiert, wohingegen plan-basierte Agenten nach dem deklarativen Paradigma modelliert sind.

In der Literatur findet man "verhaltensbasiert" oft als synonymen Begriff zu "reaktiv". Verhaltensbasierte Agenten werden als Stimulus-Response-Systeme definiert, die durch externe Triggerbedingungen aktiviert werden [Fer89, Ste90]. Unsere Definition von "Verhalten" subsumiert diese Beziehung: da ein Verhaltensmuster stets mit der Ausführung von Aktionen in der Welt verbunden ist, müssen verhaltensbasierte Agenten reaktiv sein in dem Sinne, daß sie Änderungen in ihrer Umgebung wahrnehmen und darauf reagieren können.

Nach unserer Ansicht hat das verhaltensbasierte Paradigma jedoch weit mehr zu bieten als die bloße Beschreibung von reaktiven Agenten: Verhaltensmuster sind geeignet zur Beschreibung jeglicher Aktivitäten, die ein Agent ausführen kann, ohne eine explizite Planstruktur dafür aufzubauen, d.h. ohne "viel darüber nachzudenken". Das Entlanggehen eines Ganges oder das Starten eines Autos sind Beispiele für solche Routinetätigkeiten. Verhaltensmuster liefern eine Möglichkeit, solche Routineaktivitäten mehr oder weniger unbewußt auszuführen. Kleinere Zwischenfälle können dabei von dem jeweiligen Verhaltensmuster behandelt werden. Wenn ernsthafte unvorhergesehene Situationen auftauchen, muß entweder eine andere Ebene von Verhaltensmustern aktiviert werden (vgl. Subsumptionsarchitekturen [Bro86]), oder die Kontrolle geht zu einem planbasierten Mechanismus über. In dieser Hinsicht sind Verhaltensmuster *kompilierte Multipläne*, d.h. Mengen möglicher Pläne, die einem intelligenten Ausführungsmechanismus unterworfen sind[1]. Betrachtet man physikalische Systeme, dann kann zumindest ein Teil dieser Verhaltensmuster durch "feste Verdrahtung" in die Hardware kodiert werden.

3 INTERRAP - das Konzeptionelle Modell

Bild 1 zeigt das konzeptionelle INTERRAP Aktions- und Interaktionsmodell. Dabei wird davon ausgegangen, daß ein Agent über eine Menge skriptähnlicher *Aktionsmu-*

[1] Dies wirft die interessante Frage auf, wie Verhaltensmuster aus Plänen gelernt werden können (vgl. dazu auch Newell's SOAR system [Wal91]).

ster P_A und *Interaktionsmuster* P_I verfügt, die seine lokalen und interaktiven Problemlösefähigkeiten beschreiben. Bei der Definition dieser Muster vollziehen wir eine klare strukturelle Trennung zwischen dem deskriptiven Teil, der die Repräsentation, Erkennung und Bewertung von Situationen und mögliche Ergebnisse der Ausführung der Aktions-/Interaktionsmuster beschreibt, und dem Ausführungsteil selbst. Wir

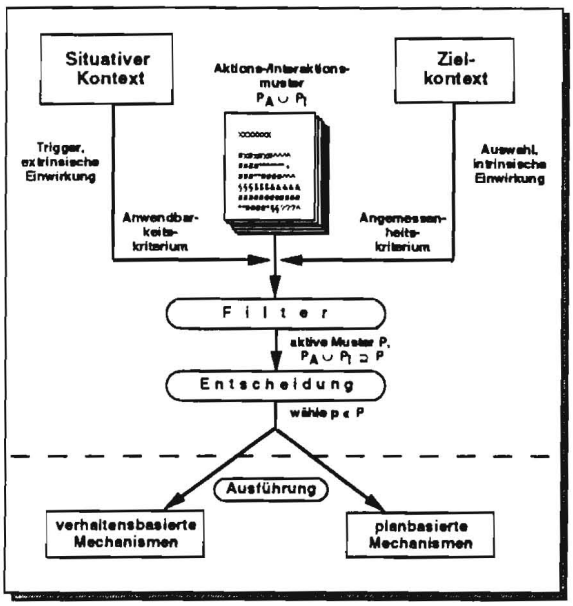


Abbildung 1: Struktur des Konzeptionellen Modells

verwenden zwei Kontexte zur Bestimmung des Verhaltens eines Agenten: der *situative Kontext* (extrinsische Einwirkung auf den Agenten) beschreibt die Muster, die in einer gegebenen (externen) Situation anwendbar sind. Der *Zielkontext* (intrinsische Einwirkung auf den Agenten) beschreibt, welche Muster für den Agenten im Hinblick auf die Erreichung seiner aktuellen Ziele geeignet sind. Zusammen betrachtet bewirken diese beiden Kontexte ein Filter, welches die Menge der Muster $P_A \cup P_I$ auf eine Teilmenge $P \subseteq P_A \cup P_I$ aktiver Aktions- bzw. Interaktionsmuster reduziert. Diese Festlegung spiegelt die Aussage eines der meistzitierten Lehrsätze aus der Soziologie [Lew35] wieder, nach der das Verhalten eines Agenten abhängig ist von Persönlichkeitseigenschaften (die wir hier vereinfacht durch die Ziele und die sich daraus ableitenden Entscheidungen darstellen) und von situativen Bedingungen.

In einem weiteren Entscheidungsschritt wird ein Muster $p \in P$ ausgewählt und kommt zur Ausführung[2]. Diese Entscheidung wird durch Bewertungsinformation unterstützt, die in der Definition der Aktions-bzw. Interaktionsmuster enthalten ist. Zur Ausführung von p können entweder verhaltensbasierte oder planbasierte Mechanismen

[2]Gehen wir von einem nebenläufigen Agentenmodell aus, so kann in diesem Entscheidungsschritt eine Menge $P' \subseteq P$ von Mustern ausgewählt und parallel ausgeführt werden.

angewendet werden. Diese Auswahl erfolgt in unserem Modell nach heuristischen Kriterien, die vom Designer des Systems vorgegeben werden. Für eine detailliertere Beschreibung und Diskussion der Interaktionsmuster verweisen wir auf [Mü93].

Diese Architektur ermöglicht es uns, reaktive, völlig verhaltensbasierte Agenten zu bauen, indem wir Aktions-/Interaktionsmuster definieren, die sehr "situationslastig" sind, d.h., die größtenteils vom situativen Kontext abhängen, und, wenn überhaupt, nur zu einem sehr geringen Maße von den aktuellen Zielen der Agenten beeinflußt werden. Für diese Agenten werden verhaltensbasierte Ausführungsmechanismen spezifiziert. Andererseits können rationale, planbasierte Agenten durch eine stärkere Betonung des Zielkontextes und der Verwendung von Ausführungsmechanismen wie Planung und - im Falle von Interaktion - Verhandlungsführung modelliert werden.

4 INTERRAP - das Funktionale Modell

Bild 2 zeigt die Funktionalität des INTERRAP Agentenmodelles, d.h. die Art und Weise seiner Operationalisierung. Es besteht aus vier grundlegenden Komponenten, nämlich

- Der Schnittstelle zur Welt,
- der verhaltensbasierten Komponente (BBC, engl. *behaviour-based component*),
- der planbasierten Komponente (PBC, engl. *plan-based component*), und
- der Wissensbasis.

Die Weltschnittstelle realisiert die Perzeptions-, Aktions-, und Kommunikationsfähigkeiten des Agenten. Die BBC beschreibt das Basisverhalten des Agenten und die Ausführungskomponente. Sie steht in enger Kopplung zur Weltschnittstelle, und somit zu den Geschehnissen in der Umwelt des Agenten. Die PBC enthält einen Planungsmechanismus, der sowohl lokale *Ein-Agenten-Pläne* als auch Pläne für mehrere Agenten (sogenannte *joint plans*) erzeugen kann.Die Pläne sind als hierarchische Skelettpläne [BKL92] modelliert, deren Knoten entweder wiederum Teilpläne oder ausführbare Verhaltensmuster sein können. Die Wissensbasis enthält Wissen über die Welt, über andere Agenten, und über den Agenten selbst. BBC und PBC sind über einen Kontrollmechanismus verbunden, der eine dynamische Schachtelung und eine flexible Übertragung der Kontrolle zwischen beiden Modulen erlaubt. Diese geschachtelte Kontrollstruktur stellt eine echte Erweiterung bestehender Ansätze zur Kombination von reaktiven und rationalen Verhalten dar. Die letzteren sind auf die Iterationstiefe *eins* beschränkt, d.h. entweder es existiert eine verhaltensbasierte Komponente, die in gewissen Situationen einen Planer aktivieren kann (z.B. Newell's SOAR system [Wal91]), oder aber die Systeme beruhen auf einem Planer mit einem Mechanismus für die Behandlung von Unterbrechungen und für Neuplanung (z.B. [PR90]). Im folgenden Abschnitt wird die Arbeitsweise des Kontrollmechanismus anhand eines Anwendungsbeispiels deutlich gemacht.

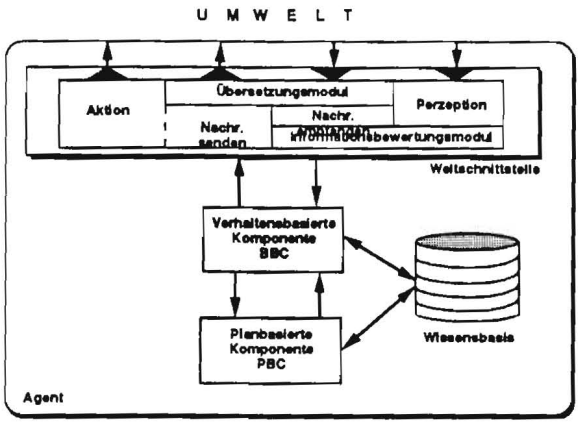


Abbildung 2: Aufbau des Funktionalen Modells

5 Eine Anwendung: das Verladehof-Szenario

Die in diesem Abschnitt beschriebene Anwendung modelliert die Simulation eines automatisierten Verladehofes. Bild 3 zeigt den Aufbau des Szenarios. Die Agenten sind automatisierte Gabelstapler, deren Aufgabe es ist, LKWs, die in den Verladehof einfahren, zu be- oder zu entladen. In dem Verladehof befinden sich Regale, in denen verschiedene Güterarten gelagert sind. Hauptmerkmal des Szenarios ist seine immense physikalische Dynamik. Die Agenten, die sich im Verladehof bewegen, müssen ihre Aktionen synchronisieren und koordinieren. Dabei sind Kollisionen zu vermeiden und Konfliktsituationen, in denen Agenten einander blockieren, aufzulösen. Darüberhinaus sollten die Agenten in der Lage sein, zu kooperieren. Bei der Beschreibung

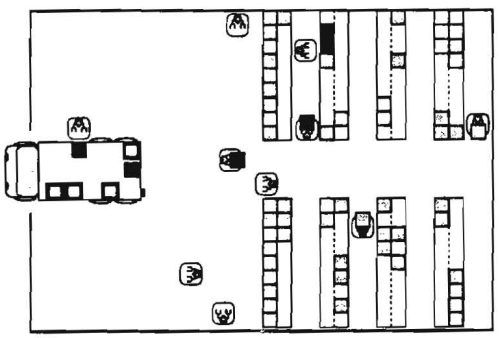


Abbildung 3: Das Verladehofszenario

der Anwendung werden wir uns in dieser Arbeit auf das Zusammenspiel der verhaltensbasierten und der planbasierten Komponente konzentrieren. Für eine detaillierte Abhandlung der relevanten Interaktionsvorgänge in dem Szenario verweisen wir auf

[Mü93].

Ausgehend von einer Zielhierarchie, die die beiden möglichen übergeordneten Ziele `load_truck` (Beladen des LKWs mit einer Kiste) und `unload_truck` (Abladen einer Kiste vom LKW und Transport der Kiste in ein passendes Regal) wurde eine Bibliothek mit Plänen und Verhaltensmustern zum Erreichen der einzelnen Teilziele implementiert. In Bild 4 ist ein Teil dieser Plan- und Verhaltensbibliothek abgebildet. Die Syntax eines Bibliothekseintrages ist

`lpb_entry( Ziel, Typ, Ausführung ).`

Dabei können Einträge in der Bibliothek entweder Skelettpläne (bezeichnet durch `Typ` "s") oder ausführbare Verhaltensmuster (bezeichnet durch `Typ` "b") sein. Der Eintrag `Ausführung` beschreibt, wie das jeweilige `Ziel` erreicht werden kann, also beispielsweise durch das Erfüllen einer Liste von Teilzielen. Beispielsweise muß, um eine Kiste auf einen LKW zu laden (Ziel `load_truck`), zunächst die Kiste gefunden und aufgenommen werden. Dann muß der LKW erreicht und die Kiste dort auf einen freien Platz abgestellt werden. Um eine Kiste zu finden, muß der Agent zunächst das passende Regal (hier ausgedrückt durch eine *Region* um das Regal herum) finden, und innerhalb dieser Region die Kiste suchen. Zum Finden des passenden Regales (Ziel `goto_region`) existieren in dem Beispiel zwei Möglichkeiten: der Agent kann zwischen der Generierung eines Planes (in Form einer Liste von Landmarken, die nacheinander anzufahren sind, und die beispielsweise durch den Voronoi-Algorithmus [OY82] berechnet werden können) und der Ausführung eines Verhaltensmusters (beispielsweise unter Verwendung einer gewichteten Zufallsfunktion oder einer auf Potentialfeldern basierenden Methode [BLL91]) wählen. Bei der Expansion des Skelettplans wird jedes Vorkommen des Prädikates `generate` durch das Ergebnis des Aufrufs seines ersten Arguments ersetzt, wodurch eine Liste neuer Ziele dynamisch erzeugt werden kann. Beispielsweise wird das Ziel

$$\texttt{generate}(\texttt{find\_list\_of\_landmarks}(\texttt{LL}, \texttt{box}_{23}, \texttt{region}_{12}))$$

ersetzt durch eine Liste

$$[\texttt{goto\_landmark}(\texttt{l}_0), \ldots, \texttt{goto\_landmark}(\texttt{l}_k)],$$

so daß die Landmarke l_k innerhalb $\texttt{region}_{12}$ liegt. Wie man sieht, erlaubt das INTERRAP Modell auf diese Weise eine flexible Kombination von Verhaltensmustern und Plänen. In dem verhaltensbasierten Teil ist weiterhin das Basisverhalten der Agenten spezifiziert. Dies beinhaltet die Definition der Aktions- und Interaktionsmuster sowie verschiedene Aspekte der Kontrolle, beispielsweise Bedingungen für die Aktivierung der planbasierten Komponente und Default-Verhalten wie das Warten auf neue Aufgaben oder das Durchführen von Zufallsschritten. Da es Agenten stets möglich ist, sich auf diese allgemeinen Verhaltensmuster zurückzuziehen[3], welches mit den untersten Schichten einer Subsumptionsarchitektur [Bro86] verglichen werden kann, ist die Agenten robust und verhalten sich in den meisten Fällen vernünftig.

[3] Beispielsweise kann ein Agent mit Hilfe einer *random walk* Strategie beide Top-Level Ziele in endlicher Zeit erreichen, setzt man eine begrenzte Fläche des Verladehofes voraus [Chu74].

```
lpb_entry( load_truck(T,Box), s, [find(Box), get(Box), find(T), put(Box)] ).
lpb_entry( find(X), s, [goto_region(X,R), search(X,R)] ).
lpb_entry( goto_region(X,R), s, [generate(find_list_of_landmarks(LL,X,R))] ).
lpb_entry( goto_region(X,R), b, [behaviours:goto_region(X,R)] ).
lpb_entry( goto_landmark(L), s, [generate(find_list_of_fields(LF,L))] ).
lpb_entry( goto_field(F), s, [generate(find_list_of_moves(LM,F))] ).
lpb_entry( turn_right, b, [actions:turn_to(right,90)] ).
                    ...
```

Abbildung 4: Die INTERRAP Plan- und Verhaltensbibliothek

Im folgenden werden wir genauer auf das Problem des Kontrollflusses zwischen der verhaltensbasierten und der planbasierten Komponente eingehen. Im Normalfall übt die BBC die Kontrolle aus. Dies ist vernünftig, da diese Komponente in enger Verbindung mit der Weltschnittstelle steht, und die unmittelbar für das "Überleben" des Agenten wesentlichen Entscheidungen zu treffen hat. Beim Auftreten einer speziellen Situation, die ein komplexere, intelligenteres Verhalten erfordert (beispielsweise erhält der Agent einen Auftrag $\texttt{load\_truck}(\texttt{truck}_1, \texttt{box}_{23})$), geht die Kontrolle auf die PBC über, wo ein Plan ausgearbeitet wird. Bei der Ausführung der einzelnen Schritte des Plans können nun wiederum Verhaltensmuster verwendet werden, d.h. die BBC übernimmt erneut die Kontrolle. So kann beispielsweise die (geradlinige) Bewegung von einer Landmarke zu einer anderen als Verhaltensmuster modelliert werden, d.h. ohne sie in eine feinere explizite Planstruktur aufzuspalten. Nehmen wir nun aber an, daß,

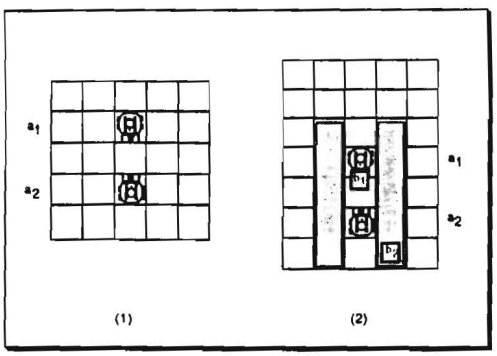


Abbildung 5: Interaktionen zwischen Agenten im Verladehof-Szenario

während der Agent sich auf seinem Weg zwischen zwei Landmarken befindet, es zu einer der in Bild 5 beschriebenen Situationen kommt, ihm also ein anderer Agent unvermittelt den Weg versperrt. In diesem Falle gibt es zwei Möglichkeiten: entweder kann die Behandlung dieser Situation innerhalb des gegenwärtig aktiven Verhaltens-

musters erfolgen (beispielsweise könnte man in Situation (5.1) erwarten, daß der Agent einfach zufällig einen Schritt zur Seite macht), oder, wenn eine kompliziertere Situation vorliegt, kann der Einsatz von planbasierten Methoden der Konfliktlösung erforderlich sein. So könnten die beiden Agenten einen gemeinsamen Plan zur Lösung des Konfliktes aushandeln (dies kann notwendig sein, wenn der Konflikt innerhalb eines engen Ganges auftritt, vgl. Bild (5.2)). Im letzteren Fall wird die Kontrolle erneut von der planbasierten Komponente übernommen. Dieses Wechselspiel wird so lange fortgesetzt, bis der Agent seine aktuellen Ziele erreicht hat.

Implementierung Ein Prototyp des Verladehofszenarios wurde in unserer Gruppe implementiert. Dabei ging man zunächst von je einer rein verhaltensbasierten und einer rein planbasierten Beschreibung der Problemlöseprozesse im Szenario aus. So wurden zwei Agenten konzipiert, die beide in der Lage waren, die in dem Szenario gestellten Aufgaben zu erledigen, der eine unter Verwendung verhaltensbasierter Problemlösestrategien, der andere unter Verwendung planbasierter Mechanismen. Die beiden Agenten wurden als UNIX Prozesse in einer lokalen Netzwerk-Umgebung realisiert. Der verhaltensbasierte Agent wurde in der Multiagenten-Basissprache MAGSY [FW92], einer Erweiterung von OPS-5, implementiert, der rationale Agent dagegen in PROLOG.

Die Funktionalität dieser beiden Agenten entspricht weitgehend den Komponenten BBC und PBC eines INTERRAP Agenten. Der nächste Schritt liegt in der Kopplung der beiden Komponenten gemä3 dem oben beschriebenen Kontrollmechanismus. Dies stellt die Voraussetzung für ein flexibles Testbett dar, das es erlaubt, bezüglich einer vorgegebenen Zielhierarchie den Einsatz verhaltens- und planbasierter Aktions- und Interaktionsmechanismen zu "mischen" und somit mehr Klarheit über die Möglichkeiten und Grenzen des Einsatzes beider Paradigmen zu erhalten. Erste Ergebnisse bestätigen jedenfalls den Nutzen und die Notwendigkeit, dem Entwickler eines Multiagentensystems einen ganzen "Baukasten" sowohl verhaltensbasierter als auch planbasierter Agentenfunktionalitäten zur Verfügung zu stellen.

6 Zusammenfassung

In diesem Artikel wurde das INTERRAP Agentenmodell vorgestellt. Seine Angemessenheit und Leistungsfähigkeit wurde anhand der Verladehofdomäne, einer Robotikanwendung, evaluiert. Der wissenschaftliche Beitrag dieser Arbeit ist wie folgt zu bewerten: INTERRAP ermöglicht es, in flexibler Weise Fähigkeiten verhaltensbasierter und planbasierter Agenten zu kombinieren. Auf diese Weise kann der Designer eines Multiagentensystems beim Entwurf eines konkreten Szenarios aus einem breiten Spektrum von Aktions- und Interaktionsmechanismen die im Spezialfall geeigneten auswählen. Konkret bietet INTERRAP zum einen *Verhaltensmuster* zur Modellierung reaktiver Agenten und zur effizienten Ausführung von Routinetätigkeiten an, zum anderen werden Planungsmechanismen für Ein-Agenten- und Multi-Agenten-Planung bereitgestellt. Durch einen flexibles Zusammenspiel zwischen der verhaltensbasierten

und der planbasierten Komponente bedingt sind die INTERRAP Agenten sehr robust und können die meisten Situationen erfolgreich bewältigen.

Selbstverständlich hat auch der INTERRAP Ansatz seine Grenzen: zum einen, wie bereits Brooks bemerkt hat [Bro91], muß einem bewußt sein, daß auch eine noch so gute Simulation keinen Ersatz für eine reale Robotik-Experimentierumgebung darstellen kann. Durch die für die Simulation notwendige Modellbildung werden einige Probleme, insbesondere diejenigen, welche die Perzeption und die physikalische Kontrolle der Roboter betreffen, wegabstrahiert oder zumindest vereinfacht. Da unsere Arbeit sich auf Multi-Agenten-Aspekte konzentriert, halten wir dies jedoch für verzeihlich. Darüberhinaus ist im Verladehofszenario ein einfaches Perzeptionsmodell für die Gabelstapler implementiert.

Zum anderen ist das Problem, einen guten Kompromiß zwischen verhaltensbasierter und planbasierter Problemlösung zu finden, sicherlich nicht trivial. Erste Erfahrungen scheinen zu belegen, daß, in dem Maße, in dem unsere Agenten planbasiert werden, sie zum einen zunehmend mit den traditionellen KI-Problemen wie dem *Frame-Problem* [Hay73] konfrontiert werden, und sie zum anderen an der Flexibilität verlieren, die in dynamischen Szenarien gewünscht ist - ein Großteil der Rechenleistung muß für lokale Umplanung aufgrund veränderter äußerer Gegebenheiten aufgewendet werden. Andererseits haben reaktive, verhaltensbasierte Agenten oft Schwierigkeiten, sich in "Standardsituationen", in denen mehr oder weniger festgelegte Abfolgen von Aktionen erforderlich sind, und in komplexen Interaktionssituationen intelligent zu verhalten. Hier zahlt sich das flexible INTERRAP Konzept in besonderem Maße aus, da es in einfacher und eleganter Weise erlaubt, mit verschiedenen Mechanismen zur Problemlösung auf verschiedenen Ebenen der Problemhierarchie (siehe Bild 4) zu experimentieren. Dadurch erhoffen wir uns, allgemeinere Regeln zu entdecken, die den Anwendungsbereich und die Grenzen verhaltensbasierter und planbasierter Ansätze beschreiben.

Literatur

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

[BKL92] A. Bernardi, C. Klauck, and R. Legleitner. Pim: Planning in manufacturing using skeletal plans and features. Technical Report RR-92-19, German Research Centre for Artificial Intelligence, 1992.

[BLL91] J. Barraquand, B. Langlois, and J. C. Latombe. Numerical potential field techniques for robot path planning. In *Proc. of the 5th International Conference on Advanced Robotics.* Pisa, Italy, 1991.

[Bro86] Rodney A. Brooks. A robust layered control system for a mobile robot. In *IEEE Journal of Robotics and Automation*, volume RA-2 (1), April 1986.

[Bro91] R. A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–159, 1991.

[Chu74] K. L. Chung. *Elementary Probability Theory with Stochastic Processes.* Springer, New York, 1974.

[FD90] T. Fraichard and Y. Demazeau. Motion planning in a multi-agent world. In Y. Demazeau and J.-P. Müller, editors, *Decentralized A.I.*, pages 137–153. North-Holland, 1990.

[Fer89] J. Ferber. Eco-problem solving: How to solve a problem by interactions. In *Proc. of the 9th Workshop on DAI*, pages 113–128, 1989.

[FHN71] R. E. Fikes, P. E. Hart, and N. Nilsson. Strips: A new approach to the application of theorem proving. *Artificial Intelligence*, 2:189–208, 1971.

[FW92] K. Fischer and H. M. Windisch. MAGSY- ein regelbasiertes Multiagentensystem. In H. J. Müller, editor, *KI1/92, Themenheft Verteilte KI.* FBO-Verlag, 1992.

[Hay73] P. J. Hayes. The frame problem and related problems in artificial intelligence. In A. Elithorn and D. Jones, editors, *Artificial and Human Thinking.* Jossey-Bass, 1973.

[Lat92] J. P. Latombe. How to move (physically speaking) in a multi-agent world. In Y. Demazeau and E. Werner, editors, *Decentralized A.I. 3.* North-Holland, 1992.

[Lew35] K. Lewin. *A dynamic theory of personality.* New York, 1935.

[Mü93] J. P. Müller. Towards a model for flexible agent interaction. Technical Report RR-93-01, German Research Centre for Artificial Intelligence, Saarbrücken, (Forthcoming), 1993.

[OY82] C. Ó'Dúnlaing and C. K. Yap. A retraction method for planning the motion of a disc. *J. of Algorithms*, 6, 1982.

[PR90] M. E. Pollack and M. Ringuette. Introducing the tile-world: Experimentally evaluating agent architectures. In *Proc. of the Conference of the American Association for Artificial Intelligence*, pages 183–189, 1990.

[ST92] Y. Shoham and M. Tennenholtz. On the synthesis of useful social laws for artificial agent societies. In *Proc. of AAAI-92*, pages 276–281, 1992.

[Ste90] L. Steels. Cooperation between distributed agents through self-organization. In Y. Demazeau and J.-P. Müller, editors, *Decentralized A.I.*, pages 175–196. North-Holland, 1990.

[Suc87] L. A. Suchman. *Plans and Situated Actions.* Cambridge Universtiy Press, Cambridge, 1987.

[Wal91] P. Wallich. Silicon babies. *Scientific American*, December 1991.

[Win75] T. Winograd. Frame representations and the declarative / procedural controversy. In D.G. Bobrow and A. Collins, editors, *Representation and Understanding - Studies in Cognitive Science*, pages 185–210. Academic Press, 1975.

Intentionalität in der Modellierung von Agenten

Ulrich Meyer,

TU-Berlin, FR 6-7, Fachgebiet Systemanalyse/EDV, Franklinstr. 28/29, 1000 Berlin 10,

Tel.: (030) 314-73134, FAX: (030) 314-22357, email: umeyer@cs.tu-berlin.de

Stichwörter / Keywords:

Modelle interagierender Agenten, Theoretische Grundlagen von Multiagentensystemen

1. Einleitung

Dieses Papier gibt eine kurze Einführung in die Analytische Philosophie des Geistes (im anglo-amerikanischen Bereich: Philosophy of Mind), um anschließend zwei Begriffsgruppen dieses Bereiches skizzieren zu können. In Anlehnung an sie wird auf einen entwickelten Formalismus für Intentionalität bei der Modellierung von Agenten hingeführt, der im Rahmen des Forschungsprojektes MAGIC[1] an der TU-Berlin entwickelt wurde. Es wurde sich dabei hauptsächlich auf Arbeiten von Dennett und Searle bezogen, die über das spezielle Problem der Intentionalität hinaus eine interessante Perspektive aufzeigen, wie Zusammenarbeit zwischen Philosophen und Wissenschaftlern im Bereich der Künstlichen Intelligenz (KI) aussehen kann, ohne sich dem Anspruch der "adäquaten Formalisierung" zu verpflichten, wie sie in der KI mit dem Namen Bratman verbunden ist.

1.1 Intentionalität und das Leib-Seele Problem

Mit Descartes wurde in der ersten Hälfte des 17. Jahrhunderts das Leib-Seele Problem (auch: Körper-Geist oder im englischen: Mind-Body Problem) in der Philosophie aufgeworfen. Er glaubte, daß die belebten und unbelebten Phänomene der Natur durch gegenseitige physikalische Beeinflussung von Korpuskeln zu erklären seien. Hiervon nahm er die Tätigkeit des menschlichen Geistes aus. Akzeptiert man diese Unterteilung, so wirft sich die Frage auf, wie es dann möglich ist, daß unser Geist und unsere Gedanken physikalische Auswirkungen haben können; d.h. wie wir z.B. überhaupt unseren Körper willentlich bewegen können. Philosophische Positionen, die auf einer, wie von Descartes vertretenen, Trennung von Körper und Geist basieren, werden unter dem Begriff des **Dualismus** zusammengefaßt. Diesen entgegen stehen Positionen, die die Tätigkeit des Geistes und damit unsere mentalen Zustände als Zustände des Gehirns und somit als körperliche Zustände erklären. Diese Positionen werden unter dem Begriff des **Materialismus** zusammengefaßt.

Intentionalität ist nun eng an diese Fragestellung gebunden, da sie von vielen Philosophen als das Kriterium zur Unterscheidung zwischen mentalen und physikalischen Zuständen gesehen wird. Unter Intentionalität wird hier das Phänomen verstanden, daß mentale Zustände **von etwas handeln**, was selber außerhalb des Trägers des mentalen Zustands liegt. Physikalische Zustände können zwar in einer kausalen Beziehung zu anderen physikalischen Phänomenen außerhalb ihrer selbst liegen, sie handeln aber nicht von ihnen.

---

[1]MAGIC: Multi-Agent-Architecture for Intelligent Consulting. Dieses Projekt wird von der Deutschen Forschungsgemeinschaft im Rahmen des DFG-Schwerpunktprogrammes "Verteilte DV-Systeme in der Betriebswirtschaft" gefördert.

Franz Brentano war der erste neuzeitliche Philosoph, der sich im 19. Jahrhundert auf das Thema der Intentionalität konzentrierte. Dabei beschäftigte er sich vor allem mit der Tatsache, daß die Objekte, von denen mentale Zustände handeln, nicht existent zu sein brauchen. Ein kleines Kind, das an den Weihnachtsmann glaubt, hat einen mentalen Zustand, der von einem nicht-existenten Objekt handelt.

Dieses Problem führte in der Folge zur Definition verschiedener Formen der Seins, die neben der physikalischen Existenz noch andere Seinsformen kennen. Solche Positionen entwickelten sich vor allem in der europäisch-kontinentalen Philosophie, hatten aber alle mit dem Problem der ontologischen Begriffsvielfalt zu kämpfen.

1.2 Intentionalität und Sprachphilosophie

Die Philosophen im anglo-amerikanischen Bereich konzentrierten sich mehr auf den Zusammenhang zwischen mentalen Zuständen und der Tatsache, daß wir sie in Sprache ausdrücken. Mit dem Namen Roderick Chisholm verbinden sich die Versuche, Sätze, die intentionale Zustände ausdrücken, durch linguistische Kriterien zu erfassen. Dabei stellte er fest, daß intentionale Sätze gegen eine Reihe von logischen Prinzipien verstoßen können, wie z.B. erstens die funktionale Abhängigkeit des Wahrheitswertes (-> Der Wahrheitswert eines zusammengesetzten Satzes ergibt sich als Funktion aus den Wahrheitswerten seiner Teilsätze), zweitens den extensionalen Charakter symbolischer Logik (-> Der Wahrheitswert eines Satzes wird nur von Objekten - Termen - beeinflußt, welche explizit im Satz vorkommen) und der daraus folgenden Möglichkeit der Substitution gleicher Terme, ohne dabei den Wahrheitswert des Satzes zu ändern und drittens die Möglichkeit der existenziellen Generalisierung.

Zu erstens (funktionale Abhängigkeit):
Der Satz "Ich wünsche, daß vielen Kranken geholfen werden kann." ist wahr und enthält den ebenfalls wahren Teilsatz "Vielen Kranken kann geholfen werden." Ersetzen wir diesen Teilsatz durch "Viele Menschen werden krank.", so bleiben zwar Funktion und logischer Eingabewert gleich, der gesamte Satz: "Ich wünsche, daß viele Menschen krank werden." aber ist falsch.

Zu zweitens (extensionaler Charakter):
"Julias Verwandte wünschen dem jungen Mann mit der Maske nichts Schlechtes." Während dieser Satz durchaus wahr ist, können wir trotzdem nicht einfach den Begriff "junger Mann mit der Maske" durch "Romeo Montague" ersetzen, obwohl damit das gleiche Objekt bezeichnet wird. "Julias Verwandte wünschen Romeo Montague nichts Schlechtes." wäre eindeutig eine falsche Aussage.

Zu drittens (existentielle Generalisierung):
"Das Kind hat einen Ball." Setzen wir diesen Satz als wahr voraus, so kann man schlußfolgern: "Es existiert etwas, derart, daß das Kind es hat." Diese Schlußfolgerung scheitert für folgende beiden Sätze: "Das Kind glaubt an den Weihnachtsmann." "Es existiert etwas, derart, daß das Kind daran glaubt."

2. Das Begriffsfeld Mentale Zustände - Intentionale Zustände

Für diese Arbeit soll die Begrifflichkeit zugrundegelegt werden, wie sie in [Searle 87] beschrieben ist. Sie ha den Vorteil, daß zwischen mentalen und intentionalen Zuständen unterschieden werden kann, was eine Verbesserung gegenüber dem Ansatz von Chisholm darstellt, da seine Kriterien weder hinreichend noch vollständig sind, um Intentionalität charakterisieren zu können. So gibt es Sätze, die mentale Zustände beschreiben, aber keines der Kriterien erfüllen und Sätze, die zwar eines der Kriterien erfüllen, aber nicht mentale Zustände beschreiben. Die Verbesserung liegt in der Unterscheidung der verschiedenen semantischen Eigenschaften und der Einführung des präpositionalen Gehalts für Intentionalität.

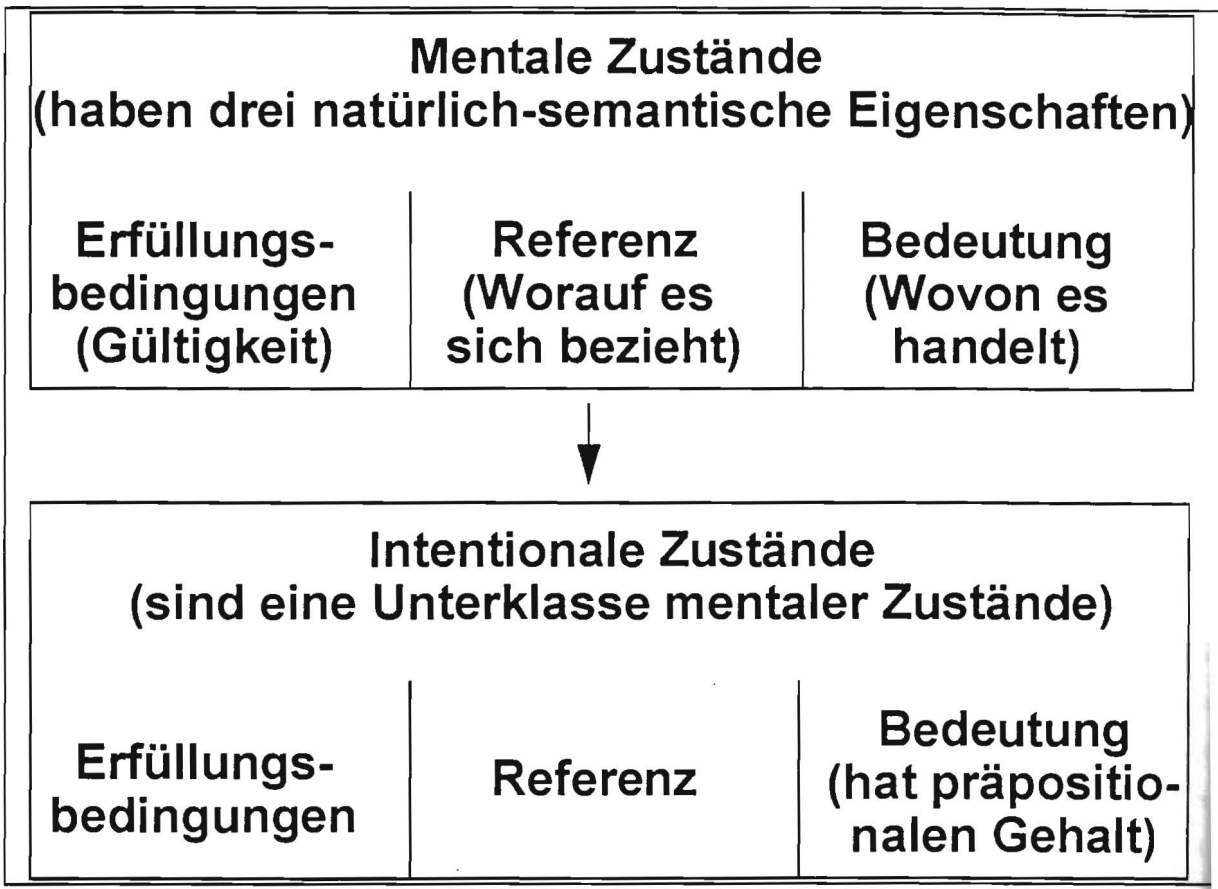


Abb.1: Das Verhältnis von mentalen und intentionalen Zuständen zueinander

Gemäß der Searl'schen Terminologie haben mentale Zustände drei natürlich-semantische Eigenschafte Erfüllungsbedingungen, Referenz und Bedeutung. Erstere stehen für die Bedingungen der Gültigkeit mental Zustände, also jene Bedingungen, bei deren Erfüllung wir z.B. sagen würden, der Satz: "Ich befinde mich i Zustand der Angst." oder einfacher: "Ich habe Angst." sei wahr. Die Referenz als semantische Eigenschaft le fest, worauf sich etwas - in diesem Fall der mentale Zustand - bezieht. Der Bezug im Beispiel wäre d Sprecher selbst und eine Vorstellung von Angst. Die Bedeutung schließlich legt fest, wovon etwas hande Diese am schwierigsten zu fassende semantische Eigenschaft wäre im Beispiel am ehesten durch je zustandsgebundenen Vorstellungen zu beschreiben, die vorliegen, wenn wir davon sprechen, Angst zu haben

Intentionale Zustände sind eine Unterklasse mentaler Zustände. Sie haben ebenfalls Erfüllungsbedingunge Referenz und Bedeutung. Bei ihnen zeichnet sich allerdings die Bedeutung durch einen präpositionalen Geh aus, der sich in einem Nebensatz mit ".... derart, daß ..." formulieren läßt; z.B.:"Ich befinde mich im Zusta des Wünschens, derart, daß ich ein Glas Wasser haben will." Der präpositionale Zusatz ist bei Intention

nicht nur notwendig (so kann z.B. die Bedeutung eines Wunsches erst mit diesem Zusatz festgelegt werden), er individuiert auch Intentionen, d.h. unterschiedliche Intentionen können durchaus die gleichen Erfüllungsbedingungen oder Referenz haben, sie haben aber auf jeden Fall eine unterschiedliche Bedeutung.

3. Das Begriffsfeld Intelligenz - Rationalität - Intentionalität

Zur Bestimmung des Rahmens, innerhalb dessen sich hier mit Intentionalität beschäftigt wird, soll kurz auf die Beziehung zwischen den drei Begriffen Intelligenz, Rationalität und Intentionalität eingegangen werden. Gerade auch für Wissenschaftler im Bereich der KI ist es sinnvoll, sich Gedanken über diese Einordnung zu machen, um sich über die Grenzen und Voraussetzungen klar zu werden, die bei der Behandlung von Intentionalität beachtet werden müssen.

Abb. 2: Das Verhältnis der Begriffe Intelligenz, Rationalität und Intentionalität

Rationales Verhalten wird als eine Form intelligenten Verhaltens angesehen. Es ist natürlich nicht die ausschließliche Form, in der sich Intelligenz äußert, soll aber für den vernunftbetonten Teil stehen und in dieser Arbeit mit dem Begriff der "Rationalität im engeren Sinne" der Philosophy of Mind gleichgesetzt werden. Intentionalität kann wiederum nur sinnvoll im Rahmen von Rationalität interpretiert werden. Ohne die Voraussetzung von Rationalität lassen sich keine Beziehungen zwischen Intentionalität einerseits und Aktionen, Zielen und Handlungen andererseits herstellen. So gibt es z.B. keine logische Garantie, daß ein irrationaler Mensch versuchen wird, seine Wünsche durch zielgerichtetes Handeln zu realisieren.

4. Einige Positionen in der Philosophie des Geistes

An dieser Stelle sollen einige philosophische Positionen kurz skizziert werden, die alle im Rahmen der Behandlung von Intentionalität zu sehen sind. Bei diesen Positionen handelt es sich ausschließlich um Varianten des Materialismus, die die Tätigkeit des Geistes in direkte Beziehung zur Tätigkeit des Gehirns setzen. Die Darstellung orientiert sich dabei an den Möglichkeiten der KI und somit an der Fragestellung,

inwieweit die Bedeutung unserer geistigen Tätigkeiten, also die natürliche Semantik durch "körperliche" sprich materielle, an Symbole, Syntax und formale Semantik gebundene Verfahren dargestellt werden kann.

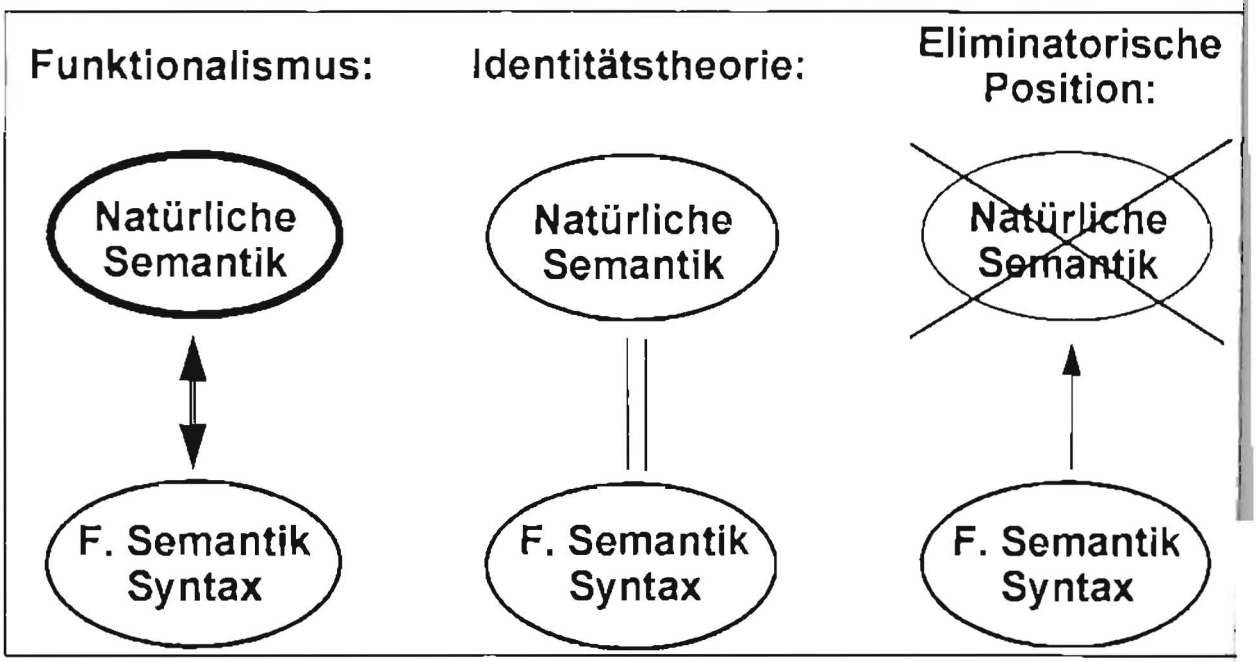


Abb. 3: Einige materialistische Positionen in der Philosophie des Geistes

Der Funktionalismus setzt mentale Zustände in eine funktionale Beziehung zu Zuständen des Gehirns. Wesentlich an dieser Position ist die Meinung, daß ein bestimmter mentaler Zustand von Fall zu Fall von unterschiedlichen Zuständen des Gehirns vertreten werden kann. Dabei kommt es nur auf die funktionale Äquivalenz dieser neuronalen Zustände an [Fodor 87]. Die Identitätstheorie geht dagegen von einer direkten Abbildung zwischen mentalen und neuronalen Zuständen aus. Als eliminatorische Positionen werden diejenigen Meinungen bezeichnet, die den mentalen Zuständen ihre Existenz absprechen. Hier sind nur die neuronalen Zustände des Gehirns wichtig; unsere geistigen Tätigkeiten sind lediglich Epiphänomene der neuronalen Netzzustände [Churchland 86].

Alle diese materialistischen Positionen lassen viel Spielraum für die Betätigung von Angehörigen der empirischen Wissenschaften, wie etwa der Neurobiologie oder den Kognitionswissenschaften. Überlegt man sich, worin sich Unterschiede ergeben, so lassen sich folgende Einteilungen vornehmen:

Der Funktionalismus kommt den Kognitionswissenschaften am meisten entgegen. Er postuliert die Möglichkeit, unabhängig von "neuronalen Realisierungen" die mentalen Zustände zu untersuchen und setzt die Kognitionswissenschaften gleichberechtigt neben die Neurobiologie.

Die Identitätstheorie würde ein Primat der Neurobiologie über die Kognitionswissenschaften bedingen, da mentale Zustände genausogut in ihren neuronalen Entsprechungen untersucht werden könnten und man dort die exakteren Methoden zur Verfügung hätte.

Die eliminatorische Position hingegen macht Kognitionswissenschaften überflüssig, da ihnen schlichtweg die Existenz ihres wissenschaftlichen Gegenstandes abgesprochen wird. Wenn mentale Zustände wie Angst, Wünschen etc. nicht existieren, dann braucht es auch keine Wissenschaft, um sie zu erklären.

Diese kurze Übersicht sollte deutlich machen, warum der Funktionalismus die bei weitem favorisierte Position bei Kognitionswissenschaftlern ist und mit Bratman [Bratman 87] auch weite Verbreitung in der KI gefunden hat.

5. Dennett's Position

"I wish to examine the concept of a system whose behavior can be - at least sometimes - explained and predicted by relying on ascriptions to the system of beliefs and desires (and hopes, fears, intentions, hunches, ...). I will call such systems *intentional systems*,..." Diese Definition Intentionaler Systeme aus [Dennett 85] soll in der folgenden Abbildung visualisiert werden:

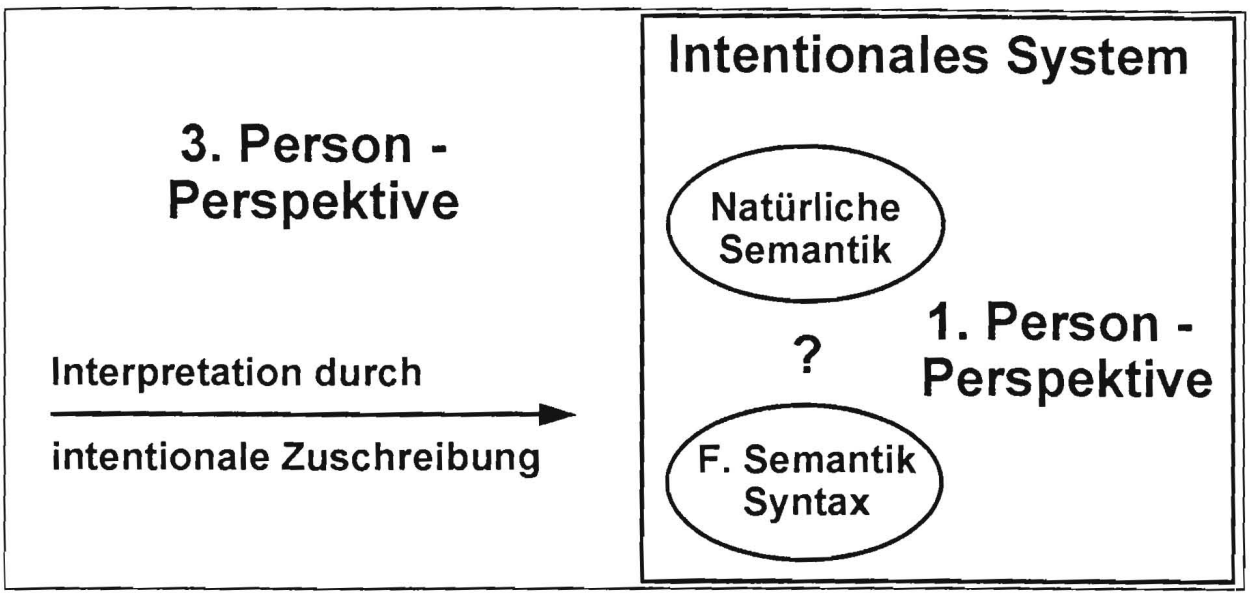


Abb. 4: Dennetts Sichtweise eines Intentionalen Systems

Gemäß Dennett zeichnen sich Intentionale Systeme dadurch aus, daß ihr Verhalten durch **Zuschreibung** von Intentionen zumindestens teilweise erklärt und vorhergesagt werden kann. Wichtig an dieser Definition ist die Perspektive der 3. Person, die gegenüber den Intentionalen Systemen durch den Terminus der Zuschreibung bezogen wird. Die Definition besteht nicht darauf, daß Intentionale Systeme im philosophischen Sinne Intentionen **haben**. Damit umgeht Dennett die Diskussion um das Leib-Seele Problem, da über die Beziehung zwischen mentalen und neuronalen Zuständen keine nähere Aussage gemacht wird. Wesentlich an dieser Position ist auch, daß Intentionalität somit nicht mehr Eigenschaft des **inneren** Zustandes eines Systems ist, sondern von außen kommt. Intentionalität ist Eigenschaft der Interpretation eines Dritten. Für eine ausführliche Diskussion der Dennett'schen Zuschreibungstheorie sei noch auf [Dennett 87] verwiesen.

6. Hinführung zum Formalismus

Agenten werden in MAGIC gemäß der Dennett'schen Definition als Intentionale Systeme gesehen. Wenn diesen Agenten nun **Intentionen** als Teil ihrer Architektur zugeordnet werden, so ist damit ein Teil Funktionalität des programmierten Agenten gemeint, der dafür verantwortlich ist, daß externe Beobachter des Agenten geneigt sein werden, das Verhalten des Agenten durch intentionale Zuschreibungen zu erklären und vorherzusagen. Dieser Teil Funktionalität erhebt den Anspruch, daß Änderungen an seiner Programmierung zu vorhersagbaren Änderungen der intentionalen Zuschreibungen der Beobachter führen wird. Dies sei die Rechtfertigung für die Namenswahl. Es wird nicht beabsichtigt, den Eindruck zu erwecken, daß die

vorgestellten Konzepte eine adäquate Formalisierung des Begriffes der Intentionalität darstellen. Dies ist auch kein Nachteil des Formalismus, da trotzdem mit ihm Intentionale Systeme gebildet werden können und darüber hinaus die Position bezogen wird, daß eine adäquate Formalisierung von Intentionalität im philosophischen Sinne nicht möglich ist.

6.1 Intentionen in MAGIC

Intentionen sind im Formalismus einem Agenten zugeordnet und werden unterteilt in zielgerichtete und handlungsgerichtete Intentionen. Beide Sorten verfügen über die jetzt formal-semantischen Eigenschaften der Erfüllungsbedingungen, Referenzen und Bedeutung. Dabei wird für die Erfüllungsbedingungen und Referenzen kein Unterschied zwischen ziel- und handlungsgerichteten Intentionen gemacht.

6.1.1 Erfüllungsbedingungen

Erfüllungsbedingungen werden durch Intentionsformeln beschrieben, die die logischen Werte "true", "false" und "unknown" annehmen können. Ihre Syntax ist wie folgt definiert:

FORMEL	::=	"(" "and" FORMEL [FORMEL]+ ")" /
		"(" "or" FORMEL [FORMEL]+ ")" /
		"(" "not" FORMEL ")" /
		ATOMAREFORMEL
ATOMAREFORMEL	::=	"(" "true" ")" / "(" "false" ")" / "(" "unknown" ")" /
		"(" PRÄDIKAT [TERM]* ")"
PRÄDIKAT	::=	"<" / ">" / "=" / "<=" / ">=" / "<>" / "isActive" / "isKnown" / "isUnknown"
TERM	::=	TASKKLASSE / VARIABLE / KONSTANTE
TASKKLASSE		ist eine Objektklasse in der Implementierungssprache Smalltalk, die unterhalb der Klasse "GenericTask" definiert wurde.
VARIABLE		ist eine Instanz der Klasse "IFVariable", die einer Objektklasse zugeordnet ist, welche unterhalb von "GenericType" definiert wurde.
KONSTANTE		ist eine Instanz der Klasse "IFConstant", die einer Objektklasse zugeordnet ist, welche unterhalb von "GenericType" definiert wurde.

Semantische Erklärungen:

Im folgenden ist mit "Die Formel ist wahr" gemeint, daß ihr Wahrheitswert "true" ist. Analoges gilt für falsch und unbekannt.

And-Formeln haben zwei oder mehr Teilformeln. Ist mindestens eine Teilformel falsch, so ist die and-Formel falsch. Sind alle Teilformeln wahr, so ist die and-Formel wahr. Ansonsten ist sie unbekannt. Auch or-Formeln haben zwei oder mehr Teilformeln. Ist mindestens eine Teilformel wahr, so ist die or-Formel wahr. Sind alle Teilformeln falsch, so ist die or-Formel falsch. Ansonsten ist sie unbekannt. Not-Formeln haben eine Teilformel. Ist sie wahr bzw. falsch, so ist die not-Formel falsch bzw. wahr. Ist sie unbekannt, so ist es die not-Formel auch.

Die ersten sechs der oben aufgeführten Prädikate können von zwei oder mehr Termen gefolgt werden. Diese Terme dürfen nur Variablen oder Konstanten des gleichen Typs sein. Eine =-Formel ist wahr, wenn alle Werte der Terme gleich sind. Sie ist falsch, wenn sich mindestens zwei Werte unterscheiden. Ansonsten ist sie unbekannt. Eine <-Formel ist wahr, wenn die Werte ihrer Terme eine streng monoton wachsende Folge bilden. Sie ist falsch, wenn ein Term einen gleichen oder kleineren Wert als ein Vorgängerterm hat. Ansonsten ist sie unbekannt. Eine >-Formel ist wahr, wenn die Werte ihrer Terme eine streng monoton fallende Folge bilden. Sie ist falsch, wenn ein Term einen gleichen oder größeren Wert als ein Vorgängerterm hat. Ansonsten ist sie unbekannt. Eine >=-Formel ist wahr, wenn die Werte ihrer Terme eine monoton fallende Folge bilden. Sie ist falsch, wenn ein Term einen größeren Wert als ein Vorgängerterm hat. Ansonsten ist sie unbekannt. Eine <=-Formel ist wahr, wenn die Werte ihrer Terme eine monoton wachsende Folge bilden. Sie ist falsch, wenn ein Term einen kleineren Wert als ein Vorgängerterm hat. Ansonsten ist sie unbekannt. Eine <>-Formel ist wahr, wenn alle Werte der Terme paarweise ungleich sind. Sie ist falsch, wenn mindestens zwei Werte gleich sind. Ansonsten ist sie unbekannt.

"isActive" ist ein unäres Prädikat. Der Term muß eine Taskklasse sein. Eine isActive-Formel ist wahr, wenn ihre Taskklasse mindestens eine aktivierte Instanz hat, ansonsten ist sie falsch. "isKnown" und "isUnknown" sind ebenfalls unäre Prädikate. Ihr Term muß eine Variable sein. Eine isKnown-Formel ist wahr, wenn der Variablen ein Wert ihres Typs zugeordnet wurde, sie ist falsch, wenn die Variable den Wert "nil" hat. Für isUnknown-Formeln gilt das Gegenteil. Konstante haben als Wert einen der Werte ihres Typs.

6.1.2 Referenzen

Die Referenzen einer Intention sind eine Teilmenge aller Szenarioobjekte, wobei letztere sich aus der Menge aller Agenten, der Menge der Environmentobjekte und der Menge der Applikationsobjekte zusammensetzen. Environmentobjekte sind Ressourcen, die den Agenten zur Verfügung stehen (z.B. Datenbanken). Die Applikationsobjekte sind die Typklassen. In ihnen sind Begriffe des Anwendungsgebietes formalisiert und in ihrer Funktionalität zusammengefaßt. Die Typen enthalten also nicht nur eine Aufzählung möglicher Werte, sondern auch Methoden. So gibt es z.B. in der MAGNIFICO[2]-Anwendung die Typklasse "Risiko", welche unter anderem Werte wie "sehr gering", "mittel" oder "spekulativ" enthält. Dazu gibt es Variablen wie "Risikobereitschaft des Anlegers", "Risiko des Anlagevorschlages" und Methoden, die Risikoberechnungen oder -abschätzungen für Anlagen im Aktienbereich, im Bereich festverzinslicher Wertpapiere etc. durchführen können.

6.2 Zielgerichtete Intentionen

Zielgerichtete Intentionen verfügen neben Erfüllungsbedingungen und Referenzen über eine Zielbedeutung, die einerseits aus einem Verweis auf das Ziel besteht, dessen Erreichen bzw. Halten Sinn der Intention ist. Ziele sind dabei als Zustände definiert, in deren Erfüllungsberechnung als Parameter der Agent eingeht, der das Ziel verfolgt. Somit kann gewährleistet werden, daß das gleiche Ziel für bestimmte Agenten erfüllt, für andere aber

---

[2]MAGNIFICO: Multi-Agent-System for Intelligent Financial Consulting. Teilprojekt von MAGIC, welches die Architekturkonzepte von MAGIC im Anwendungsbereich der Allfinanzberatung evaluiert.

nicht erfüllt ist, ohne dabei nur auf den internen Zustand des Agenten beschränkt sein zu müssen. Andererseits verfügt die Zielbedeutung über eine Tasktabelle, in der, ausgehend vom Wechsel des Wahrheitswertes der

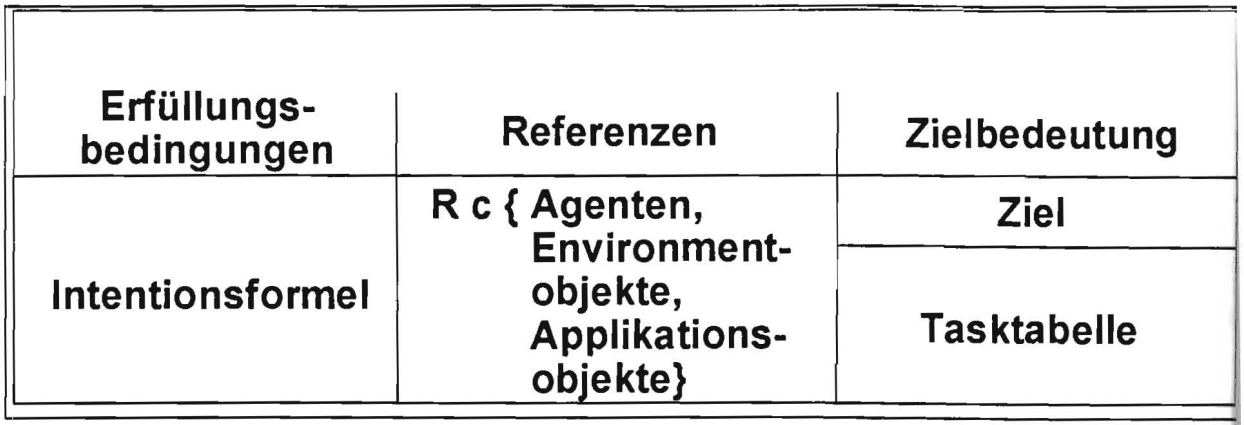


Abb. 5: Die Struktur der zielgerichteten Intentionen

Erfüllungsbedingungen und vom Erfüllungsgrad des Zieles bezüglich des Agenten, zu erledigende Tasks eingetragen sind. Tasks formulieren dabei lediglich **was** gemacht werden soll, **nicht wie** dies zu geschehen hat.

6.3 Handlungsgerichtete Intentionen

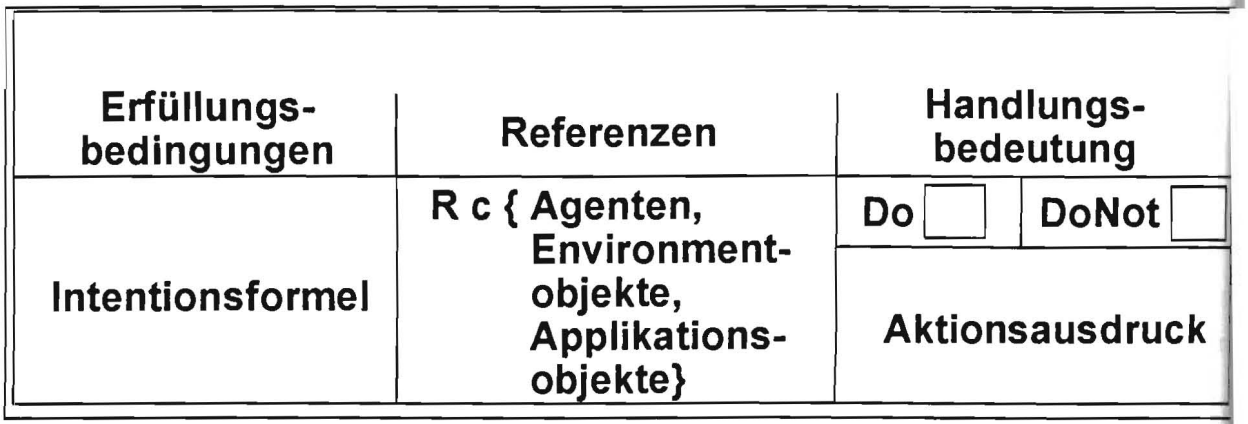


Abb. 6: Die Struktur der handlungsgerichteten Intentionen

Handlungsgerichtete Intentionen reagieren nun auf aktivierte Tasks und können, falls ihre Erfüllungsbedingungen wahr sind, Handlungen vollführen, die mit Hilfe von komplexen Aktionsausdrücken beschrieben werden. Wichtig ist hierbei die Trennung zwischen dem Begriff "Handeln" und dem Begriff "Aktion". Das Konzept des Handelns umfaßt sowohl das **Ausführen von Aktionen** wie auch das wissentliche **Unterlassen von Aktionen**, wobei letzteres einen wesentlichen Bestandteil intentionalen Handelns ausmacht (man denke an unterlassene Hilfeleistungen) und sich hauptsächlich beim Halten erreichter Ziele auswirkt, wo ein erreichter Zustand durch die Ausführung bestimmter Aktionen gefährdet sein könnte. Der Begriff des Handelns ist dabei an eine Entscheidung und damit an einen kognitiven Gehalt gebunden, womit der Begriff vom bloßen agieren im Sinne von "Aktionen ausführen" unterschieden ist. So ist z.B. das Zurückzucken vor einer heißen Herdplatte zwar eine (reaktive) Aktion, es stellt aber keine Handlung dar.

Zielt eine handlungsgerichtete Intention auf das Unterlassen einer Aktion, so wird diese Aktion auf eine "Schwarze Liste" gesetzt und nicht ausgeführt. Eine Ausführung würde auch abgelehnt werden, falls der Agent von anderen Agenten um eine Ausführung ersucht wird. Aktionen werden wieder von der Liste gestrichen,

sobald die Intention, welche die Unterlassung bewirkt hat, nicht mehr erfüllt ist, d.h. ihre Intentionsformel nicht mehr wahr ist.

Die Syntax der Aktionsausdrücke ist wie folgt:

AKTAUSDRUCK	::=	"(" "seq" AKTAUSDRUCK [AKTAUSDRUCK]+ ")" /
		"(" "par" AKTAUSDRUCK [AKTAUSDRUCK]+ ")" /
		"(" "loopCond" BOOLBLOCK AKTAUSDRUCK")" /
		"(" "cond" BOOLBLOCK AKTAUSDRUCK AKTAUSDRUCK ")" /
		ATOMAKTION
BOOLBLOCK		ist eine Instanz der Smalltalkklasse BlockClosure, die als Berechnungswert eine der Smalltalkkonstanten "true" oder "false" haben muß
ATOMAKTION	::=	"(" TASKKLASSE")" /
		"(" AKTION ")"
AKTION		ist ein Smalltalkausdruck, der in der "execution"-Instanzenmethode einer Aktionsklasse definiert ist, die als Unterklasse von "GenericAction" angelegt wurde.
TASKKLASSE		ist eine Objektklasse in der Implementierungssprache Smalltalk, die unterhalb der Klasse "GenericTask" definiert wurde.

Semantische Erläuterungen:

Seq-Ausdrücke stellen die sequentielle Ausführung der in ihnen aufgeführten Aktionsausdrücke dar. Par-Ausdrücke stehen für parallele Ausführungen. LoopCond-Ausdrücke wiederholen ihren Aktionsausdruck, solange der Wert des Boolblocks "true" ist. Cond-Ausdrücke führen den ersten Aktionsausdruck aus, falls der Boolblock den Wert "true" hat; ist der Wert "false", wird der zweite Ausdruck ausgeführt. Das Ausführen einer Taskklasse besteht aus ihrer Aktivierung, d.h. der Generierung einer Instanz des Tasks und der Abarbeitung durch den Agenten mittels anderer handlungsgerichteter Intentionen. Über diesen Mechanismus können handlungsgerichtete Intentionen ineinander geschachtelt werden. Das Abarbeiten einer Aktion besteht aus der Generierung einer Instanz der Aktionsklasse und der Ausführung der zugehörigen "execution"-Methode.

6.4 Der Interpreterlauf der Agenten

Der an den Intentionen ausgerichtete Interpreterlauf der Agenten durchläuft in einer Schleife folgende Phasen.

1. Alle über die Sensorik oder die Kommunikationskomponente neu erhaltenen Informationen werden in die Wissensbasis eingetragen. Dabei werden gegebenenfalls auch die Werte der Variablen neu gesetzt.

2. Jetzt folgt die Auswertung aller Erfüllungsbedingungen der zielgerichteten Intentionen. Da Variablen ihre Werte geändert haben können, kann dies auch für die Intentionsformeln gelten.

3. Abhängig vom Wechsel bzw. dem Bleiben des logischen Wertes der Intentionsformel und der momentanen Zielerfüllung wird nun der entsprechende Task aus der Tasktabelle aktiviert. Der Agent hat nun eine Menge von abzuarbeitenden Taskinstanzen in einer Agenda stehen.

4. Die neuen Aktivierungsstände der Taskklassen werden den Intentionsformeln mitgeteilt. Eventuell ändern einige von ihnen ihren Wert, falls sie sich mit einer isActive-Formel auf einen Task beziehen.

5. Die Erfüllungsbedingungen der handlungsgerichteten Intentionen werden ausgewertet.

6. Alle handlungsgerichteten Intentionen, die eine wahre Intentionsformel haben, werden ihre Handlung vollführen. Diese besteht entweder in der Ausführung des Aktionsausdrucks oder in seinem Verbot.

7. Der Turmbau zu Babel

Die momentan in der KI diskutierten Positionen zur Intentionalität bauen hauptsächlich auf den Arbeiten von Bratman auf und zielen auf eine möglichst adäquate Formalisierung der Intentionalität. Dieses Vorhaben gleicht dem Turmbau zu Babel, nur daß hier mit den Mitteln der formalen Logik versucht wird, sich der natürlichen Semantik zu nähern. Schicht um Schicht werden die bisherigen Konzepte erweitert bzw. verfeinert, in der Hoffnung schließlich die Formalisierung von Intentionalität finden zu können. Dabei laufen die Protagonisten diese Prozesses letztendlich in die Gefahr, sich (und vor allem ihre Leser) nur in die formale Sprachverwirrung zu stoßen.

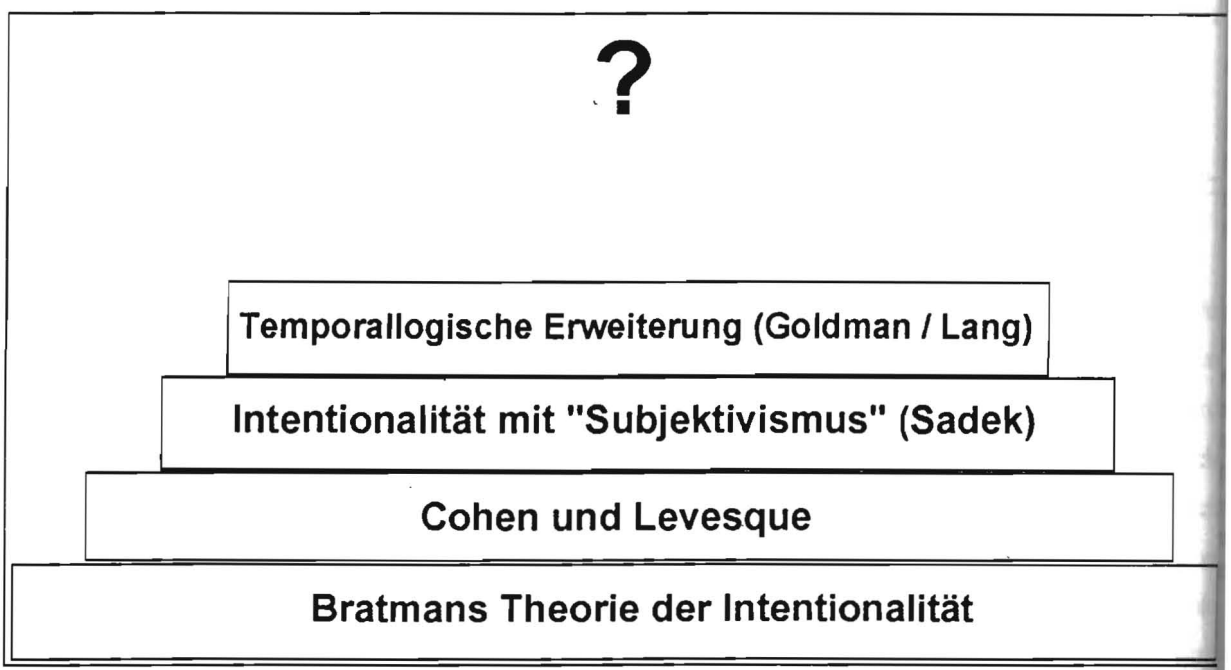


Abb. 7: Der Turmbau zu Babel

Es soll nun nicht die formale Logik kritisiert werden, sondern lediglich die Motivation dieser Vorgehensweise, die darin besteht, die Philosophie als Rezeptgeberin für die empirischen Wissenschaften zu sehen. In [Abel 93] wird nicht nur bereits von einem Scheitern der drei vorgestellten materialistischen Positionen ausgegangen, - was dem oben vorgestellten Vorhaben die Basis entziehen würde - es werden auch jene interpretatorischen

Positionen gestärkt, die sich unter anderem in der Tradition von Kant entwickelt haben und denen auch Dennetts Position zuzurechnen wäre. Es sind diese Denkgebäude, mit denen sich auch KI-ler im Zuge des Ringens der Philosophie um einen modernen Geistbegriff konfrontiert sehen werden, sollten sie sich weiterhin um einen Dialog mit der Philosophie bemühen - was wünschenswert wäre.

Modelle der KI sollten sich in Auseinandersetzung mit philosophischen Positionen bilden und bewähren, sie können keine philosophischen Positionen darstellen. Philosophie kann nicht implementiert werden. So zeigt sich auch die "Wahrheit" der Modelle der KI nicht darin, ob es eine philosophische Position gibt, die sie unterstützt, sondern darin, ob sich das Modell in der praktischen Anwendung bewährt. Nicht der Formalismus für Intentionalität ist der beste, der möglichst viele logische Konzepte umfaßt, sondern jener, bei dessen praktischer Anwendung wir Menschen am ehesten bereit wären, dem System Intentionalität zuzuschreiben.

8. Literaturverzeichnis

[Abel 93] Günter Abel: "Interpretationswelten. Gegenwartsphilosophie jenseits von Essentialismus und Relativismus."; Suhrkamp; Frankfurt/M; 1993

[Bratman 87] Michael E. Bratman: "Intention, Plans and Practical Reason"; Harvard University Press; Cambridge, Massachusetts; 1987

[Churchland 86] Patricia Smith Churchland: "Neurophilosophy. Toward a Unified Science of the Mind-Brain"; MIT Press; Cambridge, Massachusetts; 1986

[Dennett 85] Daniel C. Dennett: "Brainstorms. Philosophical Essays on Mind and Psychology"; MIT Press; Cambridge, Massachusetts; 1985

[Dennett 87] Daniel C. Dennett: "The Intentional Stance"; MIT Press; Cambridge, Massachusetts; 1987

[Fodor 87] Jerry A. Fodor: "Psychosemantics, The Problem of Meaning in the Philosophy of Mind"; MIT Press; Cambridge, Massachusetts; 1987

[Searle 87] John R. Searle: "Intentionalität. Eine Abhandlung zur Philosophie des Geistes"; Suhrkamp; Frankfurt/M; 1987

Lernen und Aktionskoordinierung in Mehragentensystemen

Gerhard Weiß
Institut für Informatik, Technische Universität München
Arcisstr. 21, 8000 München 2, Germany
weissg@informatik.tu-muenchen.de

Zusammenfassung. Lernen in Mehragentensystemen ist ein junges Forschungsgebiet innerhal Verteilten Künstlichen Intelligenz. Trotz verstärkter Forschungsbemühungen gerade in den vergan Jahren gibt es zahlreiche offene Fragen und Probleme. Dieser Artikel untersucht den Zusammen zwischen Lernen und Aktionskoordinierung in solchen Systemen. Es wird ein Verfahren vorgestellt, we mehreren Agenten ermöglicht, die Ausführung ihrer Aktionen zu koordinieren und geeignete Seque von Aktionsmengen zu lernen.

Schlüsselwörter. Mehragentensysteme, verteiltes Lernen, Aktionskoordinierung.

1. Motivation

Das Konzept interagierender Agenten ist in der Künstlichen Intelligenz und der Informatik ke wegs neu, sondern beinflußte bereits vor vielen Jahren die anfänglichen Entwicklungen in Bereichen kognitiven Modellierung [24, 22], Blackboard Systeme [7], objektorientierte Programmierung [15] und dellierung von Nebenläufigkeit [23]. In den vergangenen Jahren lieferte dieses Konzept wichtige Imp für neue Denkansätze in der Informatik (vgl. etwa [3, 4, 20, 21]) und entwickelte sich zu einem Forschu schwerpunkt in der Künstlichen Intelligenz (z.B. [2, 5, 10, 18]). Das Interesse an solchen Systemen grü vor allem auf der Tatsache, daß viele Aufgaben- und Problemstellungen – man denke etwa an auton Fertigung oder Verkehrsflußüberwachung – von sich aus einen verteilten Lösungsansatz erfordern und be mit mehreren anstatt mit einem einzelnen Agenten modelliert werden (vgl. [11]). Insbesondere ermög die Verwendung mehrerer Agenten sowohl eine bessere Handhabung von natürlichen Einschränkungen etwa die begrenzte Leistung eines einzelnen Agenten oder die physikalischen Verteiltheit der zu ve beitenden Daten als auch die Nutzung von inhärenten Eigenschaften verteilter Systeme wie Parallel Robustheit und Fehlertoleranz.

Im Gegensatz zum Problemlösen und Planen wurde dem Lernen in Mehragentensystemen lange nur wenig Aufmerksamkeit geschenkt. Dies steht in deutlichem Widerspruch zu der mittlerweile gewon Einsicht, daß gerade dem Lernen eine wichtige Rolle zukommt: da nämlich solche Systeme typischerw sehr komplex und deshalb in ihrem Verhalten nur schwer zu spezifizieren sind, sollten sie in der Lage s weitgehend selbständig zu lernen, die ihnen gestellten Aufgaben zu bewältigen. Erst seit wenigen Jah gibt es verstärkt Forschungsbemühungen, die sich mit grundlegenden Aspekten des Mehragenten-Lern beschäftigen. So wurden beispielsweise die Wechselwirkungen zwischen verteiltem und nichtverteiltem L nen [27, 25] näher untersucht und Verfahren zum verteilten Erlernen von Konzepten [6] und von plausi Hypothesen [26, 28] entwickelt. Trotz dieser Bemühungen gibt es jedoch eine Vielzahl offener Fragen Probleme im Bereich des Mehragenten–Lernens.

Die hier vorgestellte Arbeit untersucht den Zusammenhang zwischen Lernen und Aktionskoordinier in Mehragentensystemen. Im Mittelpunkt stehen dabei folgende Fragestellungen:

- Wie können mehrere Agenten lernen, welche Aktionen nebenläufig ausgeführt werden sollen?
- Wie können mehrere Agenten lernen, welche Aktionsmengen sequentiell ausgeführt sollen?

Der Artikel ist wie folgt strukturiert. Im Abschnitt 2 wird der Begriff eines Agenten konkretisiert u der Begriff einer Gruppe kompatibler Agenten eingeführt. Im Abschnitt 3 wird aufbauend auf die

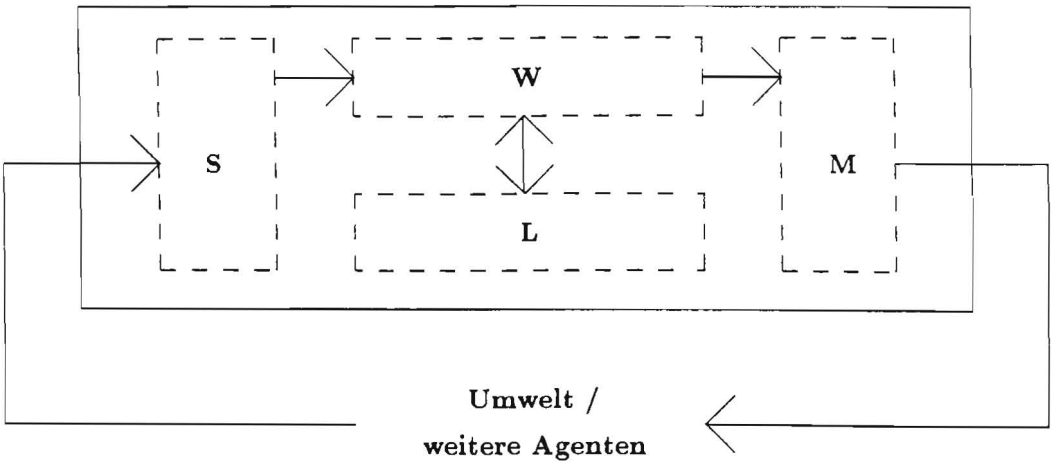


Bild 1: Struktur eines Agenten. Jeder Agent besteht aus einer sensorischen Komponente (S), einer motorischen Komponente (M), einer Wissensbasis (W) und einer Lerneinheit (L). Die sensorische und motorische Komponente erlauben dem Agenten die Interaktion mit seiner Umwelt. Die Wissensbasis enthält das Wissen des Agenten über seine Umwelt. Die Lerneinheit modifiziert die Wissensbasis derart, daß der Agent zu einem verbesserten Verhalten befähigt wird.

Agenten- und Gruppenbegriff ein Lernverfahren vorgestellt, welches auf die Beantwortung der beiden obigen Fragen abzielt. Im Abschnitt 4 werden theoretische und experimentelle Ergebnisse zu diesem Verfahren beschrieben. Im Abschnitt 5 werden mögliche Erweiterungen des vorgestellten Lernverfahrens aufgezeigt.

2. Agenten und Gruppen als organisatorische Einheiten

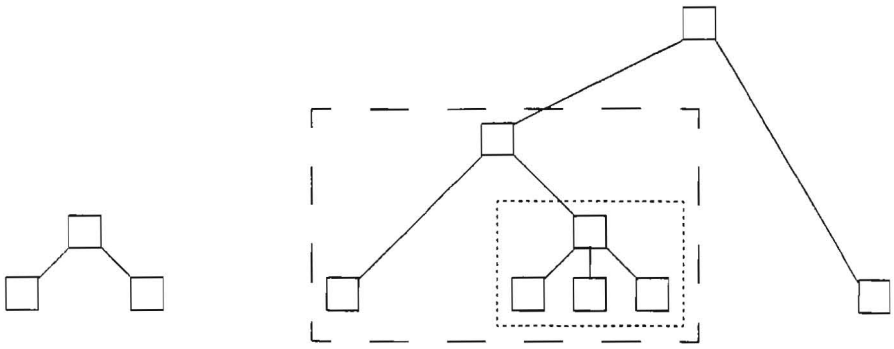
In der Literatur findet sich eine Vielzahl unterschiedlicher Spezifikationen und Implementierungen von Mehragentensystemen. Diesem Artikel liegt die in der Verteilten Künstlichen Intelligenz „prototypischen Sichtweise" zugrunde, derzufolge ein Mehragentensystem aus mehreren autonomen Agenten besteht die in der Lage sind zu interagieren und die sich in ihren Fähigkeiten und ihrem Weltwissen voneinander unterscheiden. Dabei werden zwei Typen von organisatorischen Einheiten unterschieden: einzelne Agenten und Gruppen von Agenten die kompatible Aktionen ausführen.

Jeder Agent besitzt eine sensorische Komponente, eine motorische Komponente, eine Wissensbasis und eine Lerneinheit; siehe Bild 1. Insbesondere ist jeder Agent hinsichtlich seiner Aktivitäten in folgender Weise eingeschränkt:

- aufgrund beschränkter sensorischer Fähigkeiten kennt er nur einen bestimmten Ausschnitt seiner Umwelt,
- aufgrund beschränkter motorischer Fähigkeiten ist er nur zu bestimmten umweltbeeinflußenden Aktionen fähig und
- Aktionen verschiedener Agenten können inkompatibel sein.

Wegen dieser Einschränkungen ist es möglich, daß verschiedene Agenten unterschiedliches Wissen über ihre Umwelt besitzen und auf unterschiedliche und inkompatible Aktionen spezialisiert sind.

Die einzelnen Agenten dienen als Primitive („building blocks") für komplexere Organisations- und Kontrollstrukturen. Eine der elementarsten Strukturen ist eine Gruppe von Agenten die kompatible Aktionen ausführen [8, 9]. Diese elementare Struktur liegt auch dem im nächsten Abschnitt beschriebenen Lernverfahren zugrunde. Der Gruppenbegriff ist wie folgt definiert. Eine Gruppe besteht aus einem Gruppenleiter und mehreren kompatiblen Gruppenmitgliedern, wobei ein Leiter ein einzelner Agent und jedes Mitglied entweder ein einzelner Agent oder wiederum eine Gruppe ist. Dabei bedeutet „kompatibel", daß in einem



(a) (b)

Bild 2: Struktur einer Gruppe. Eine Gruppe besteht aus einer Menge von einzelnen Agenten (□), die dur Leiter–Mitglied Relationen (/ \) hierarchisch strukturiert sind. *(a)* zeigt eine Gruppe, die nur aus d Leiter und zwei Mitgliedern besteht, wobei jedes Mitglied ein einzelner Agent ist. *(b)* zeigt eine Grup die einen Agenten und eine weitere Gruppe (gestricheltes Kästchen) als Mitglieder hat; die „gestriche Gruppe" ihrerseits hat einen Agenten und eine Gruppe (gepunktetes Kästchen) als Mitglieder, wobei „gepunktete Gruppe" ihrerseits drei einzelne Agenten als Mitglieder hat. Alle Mitglieder einer Grup müssen kompatibel sein.

gegebenen Umweltzustand die Aktivität keines der Mitglieder zu Umweltveränderungen führt, welche Aktivität eines anderen Mitgliedes verhindern. Der Gruppenleiter hat die Aufgabe, die Interessen sein Gruppe zu vertreten; insbesondere bewertet er die Zielrelevanz seiner Gruppe und entscheidet, ob sei Gruppe weiterhin als autonome Einheit agiert, Mitglied einer anderen Gruppe wird oder aufzulösen (siehe Abschnitt 3). Wie Bild 2 illustriert, erlaubt diese rekursive Definition die Bildung von einfachen auch von komplex strukturierten Gruppen.

In den nächsten Abschnitten wird folgende Sprechweise und Notation verwendet. Der Aktivitätskonte eines Agenten in einem gegebenen Umweltzustand ist definiert als derjenige Ausschnitt des Umweltzusta des, welcher dem Agenten bekannt ist. In jedem Umweltzustand ist der Aktivitätskontext einer Grup definiert als die Summe der Aktivitätskontexte aller in dieser Gruppe enthaltenen Agenten. Ein Agent potentiell aktiv in einem Umweltzustand, wenn er seine Aktion ausführen könnte; eine Gruppe ist pote tiell aktiv, wenn alle beteiligten Agenten potentiell aktiv sind. Ein Agent und eine Gruppe ist autonom einem Umweltzustand, wenn er bzw. sie nicht Mitglied einer in diesem Umweltzustand potentiell aktiv Gruppe ist. S_j bezeichnet einen Umweltzustand. U_i bezeichnet eine organisatorische Einheit, also ein Agenten oder eine Gruppe. Falls U_i eine Gruppe ist, dann verweist $\overline{U_i}$ auf den Leiter dieser Gruppe; fa U_i ein einzelner Agent ist, dann bezeichnet $\overline{U_i}$ ebenfalls diesen Agenten. Schließlich bezeichnet $[U_i, S_j]$ d Aktivitätskontext der Einheit U_i im Zustand S_j.

3. Der DFG Algorithmus

3.1. Prinzipielle Arbeitsweise

Der DFG Algorithmus – DFG als Akronym für „Dissolution and Formation of Groups" – ist ein Verfa ren zum Erlernen von geeigneten Sequenzen von Aktionsmengen in Mehragentensysstemen [31]. Er basi auf der aktionsorientierten Variante [30] des aus dem Bereich der sog. classifier systems stammenden buck brigade Lernprinzips [17] und stellt eine Weiterentwicklung der in [32, 33] beschriebenen Lernverfahren z verteilten Aktionskoordinierung dar. Der Algorithmus gehört zur Klasse der Reinforcement–Lernverfahr das heißt, die von der Umwelt erforderliche Lernrückkoppelung ist minimal und besteht lediglich aus gewissen Zeitpunkten bereitgestellten skalaren Werten.

Die prinzipielle Arbeitsweise des DFG Algorithmus kann wie folgt zusammengefaßt werden. Es wi zwischen einzelnen Agenten und Gruppen bestehend aus mehreren kompatiblen Agenten als den agier den organisatorischen Einheiten unterschieden (siehe Abschnitt 2). Jede Einheit bewertet die Releva

ihrer Aktivität in Abhängigkeit vom jeweiligen Umweltzustand. Lernen erfolgt durch zwei sich wechselseitig beeinflußende Prozesse: Bewertungsmodifizierung (credit assignment), also die Modifizierung der Aktivitätsbewertungen, und Gruppenentwicklung, also der Bildung neuer und der Zerfall alter Gruppen. In jedem Umweltzustand findet abhängig von den Aktivitätsbewertungen ein Wettbewerb zwischen den Einheiten statt. Nur dem Gewinner eines Wettbewerbes ist es gestattet, tatsächlich aktiv zu werden und damit den aktuellen in den nächsten Umweltzustand zu überführen. Insgesamt ergibt sich die Gesamtaktivität des Mehragentensystems durch die wiederholte Ausführung des folgenden Arbeitszyklus:

1. Aktivitätsauswahl: Die Einheiten prüfen, ob sie im aktuellen Umweltzustand aktiv werden könnten. Basierend auf den Bewertungen ihrer Aktivitäten wird diejenige Einheit bestimmt, die tatsächlich aktiv werden darf.

2. Bewertungsmodifizierung: Die aktiven Einheiten modifizieren ihre Aktivitätsbewertungen entsprechend dem bucket brigade Mechanismus.

3. Gruppenentwicklung: Abhängig von den Aktivitätsbewertungen werden alte (erfolglose) Gruppen aufgelöst und neue (erfolgreichere) Gruppen gebildet.

Im folgenden werden die einzelnen Schritte dieses Zyklus im Detail beschrieben.

3.2. Aktivitätsauswahl und Bewertungsmodifizierung

In jedem Umweltzustand S_j läuft zwischen den Agenten ein Wettbewerb um das Recht, aktiv zu werden. Für jede potentiell aktive und autonome Einheit U_i berechnet $\overline{U_i}$ ein Gebot B_i^j gemäß

$$B_i^j = (\alpha + \beta) \cdot E_i^j \quad , \tag{1}$$

wobei α eine Konstante, β ein Zufallsterm und E_i^j ein skalarer Wert ist, der die von $\overline{U_i}$ vorgenommene Bewertung der Aktivität von U_i im Kontext $[U_i, S_j]$ darstellt. (Die Autonomiebedingung gewährleistet, daß die Mitglieder einer potentiell aktiven Gruppe nicht miteinander konkurrieren. Der Zufallsterm dient der Vermeidung lokaler Lernminima; es sei hier erwähnt, daß im Bereich der classifier systems verschiedene Varianten des zufallsbeeinflußten Wettbewerbs vorgeschlagen wurden.) Die Aktivitätsauswahl erfolgt dann über die Bestimmung der höchstbietenden Einheit. Nur diese Einheit, also der Gewinner dieses Wettbewerbs, wird aktiv und generiert somit einen neuen Umweltzustand.

Die Bewertungsmodifizierung erfolgt durch eine lokale Umverteilung der Bewertungen zwischen den aktiven Einheiten. Sei U_i der Gewinner im aktuellen Zustand S_j und sei U_k der Gewinner im unmittelbar vorausgehenden Zustand S_l. $\overline{U_i}$ reduziert seine Bewertung E_i^j um den deterministischen Anteil $\alpha \cdot E_i^j$ seines Gebots und zahlt diesen Anteil an $\overline{U_k}$; dieser erhöht dann seine eigene Bewertung E_k^l um den erhaltenen Betrag. (Der aktuelle Gewinner zahlt für das Previleg, aktiv zu werden, der vorherige Gewinner wird dafür belohnt, die Umwelt in geeigneter Weise modifiziert zu haben.) Gibt es zudem eine Lernrückkoppelung R^{ext} von der Umwelt, so addiert diese der aktuelle Gewinner zur seiner eigenen Bewertung. Insgesamt ergeben sich also folgende Modifikationen:

$$E_i^j = E_i^j - \alpha \cdot E_i^j + R^{ext} \tag{2}$$

$$E_k^l = E_k^l + \alpha \cdot E_i^j \quad . \tag{3}$$

Informell lassen sich sich Auswirkungen dieser Modifikationen wie folgt beschreiben (vgl. auch [16]). In einer Sequenz von aktiven Einheiten zahlt jede Einheit einen gewissen Betrag an ihren Vorgänger und erhält einen gewissen Betrag von ihrem Nachfolger. Zum einen nimmt dadurch die Bewertung einer Aktivität einer Einheit zu (ab), wenn eine Einheit weniger (mehr) bezahlt als sie erhält. Zum anderen wird jede Bewertungsänderung im Laufe der Zeit innerhalb einer Sequenz „nach hinten weitergereicht". Dies führt zu einer Stabilisierung einer Sequenz, falls die letzte Einheit regelmäßig eine positive Lernrückkoppelung von der Umwelt erhält, und zu einer Destabilisierung, falls dies nicht der Fall ist.

3.3. Gruppenentwicklung

Gruppenentwicklung umfaßt zwei gegenläufige Prozesse: den Zerfall alter und die Formation neuer Gruppen. Beide Prozesse gehorchen folgenden Entwicklungsprinzipien:

- Anfangs ist kein Agent und keine neu gebildete Gruppe bereit, mit anderen Einheiten zu kooperie und neue Gruppen zu bilden (d.h. Gruppenmitglied zu werden).
- Eine Einheit ist nicht bereit zu kooperieren, solange die Bewertung ihrer Aktivität im Zuge Bewertungsmodifizierung eine ansteigende Tendenz aufweist.
- Eine Einheit ist bereit zu kooperieren, sobald die Bewertung ihrer Aktivität nur mehr geringfü ansteigt, stagniert oder sogar abfällt.
- Kompatible und kooperationswillige Einheiten bilden neue Gruppen.
- Eine Gruppe wird aufgelöst und zerfällt damit in ihre Mitglieder, falls die Bewertung ihrer Aktiv unter ein kritisches Minimum abfällt.

Diese Prinzipien sind folgendermaßen realisiert. Um die zeitliche Entwicklung und die Tendenz Bewertung einer Einheit U_i beurteilen zu können, berechnet $\overline{U_i}$ den gleitenden Mittelwert seiner Be tungen über die vergangenen Episoden, wobei eine Episode definiert ist als das Zeitintervall zwischen aufeinanderfolgenden externen Lernrückkoppelungen. Formal bedeutet dies, daß $\overline{U_i}$ während jeder Epis $\tau + 1$ den Mittelwert $M_i^j[\tau + 1]$ seiner Bewertung E_i^j berechnet gemäß

$$M_i^j[\tau + 1] = \frac{1}{\nu} \cdot \sum_{T=\tau-\nu+1}^{\tau} E_i^j[T] \quad ,$$

wobei ν eine Konstante ist, welche den zu berücksichtigenden vergangenen Zeitabschnitt bestimmt, $E_i^j[T]$ die Bewertung E_i^j am Ende der Episode T ist. Mit Hilfe dieses Mittelwertes lassen sich nun Krite für den Zerfall und die Bildung von Gruppen spezifizieren.

Sei $\tau + 1$ die aktuelle Episode, S_j der aktuelle Umweltzustand und U_i eine in S_j potentiell aktive autonome Einheit. Dann entscheidet $\overline{U_i}$, daß U_i im Kontext $[U_i, S_j]$ bereit ist zu kooperieren und anderen Einheiten eine neue Gruppe zu bilden, falls

$$M_i^j[\tau + 1] \leq \sigma \cdot E_i^j[\tau - \nu] \quad ,$$

wobei σ eine Konstante ist, welche das Maß der Kooperationsbereitschaft steuert. Unter allen koop tionsbereiten Einheiten wählt zunächst diejenige Einheit, die mit der höchsten Bewertung assoziiert ihre Kooperationspartner. Hierbei sind verschiedene Auswahlstrategien denkbar. In den gegenwärti Implementierung wird nur ein einzelner Partner gewählt, und zwar derjenige, dessen Aktivität kompat ist und hoch bewertet wird. Diese Auswahl wird iteriert, bis keine weiteren Gruppenneubildungen mög sind.

Umgekehrt entscheidet $\overline{U_i}$, daß seine Gruppe U_i aufzulösen ist, falls

$$M_i^j[\tau + 1] \leq \rho \cdot E^{init} \quad ,$$

wobei ρ eine Konstante ist, welche die Auflösungsbereitschaft der Gruppen reguliert. (Die Autono bedingung spielt dabei eine zweifache Rolle: im Falle der Gruppenbildung verhindert sie die Kooper onsbereitschaft einer Einheit, falls sie bereits Mitglied einer potentiell aktiven Gruppe ist, und im F des Gruppenzerfalls verhindert sie die Auflösung einer Gruppe, falls sie Mitglied einer potentiell akti Gruppe ist.)

Die Bildung und der Zerfall von Gruppen stellen äußerst kontextsensitive Prozesse dar. Dies ist a wünschenswert, da die Aktivität einer Gruppe in unterschiedlichen Umweltzuständen typischerweise a unterschiedliche Bewertungen erfordert.

4. Analyse

4.1. Theoretische Überlegungen

Da der DFG Algorithmus auf das Erlernen geeigneter Sequenzen von Aktionsmengen abzielt, ist scheidend, wie die Bewertungen von Aktivitäten aufeinanderfolgend aktiver Einheiten verändert wer Erweitert man das in [12] beschriebene Konvergenzergebnis für den bucket brigade Algorithmus auf DFG Algorithmus, so erhält man folgendes

Resultat 1 (Lernkonvergenz). Sei $\{U_{i_1}, \ldots, U_{i_n}\}$ eine Menge von organisatorischen Einheiten derart, d

(i) U_{i_k}, $1 \leq k < n$, mit $U_{i_{k+1}}$ „gekoppelt" ist, d.h. immer wenn U_{i_k} in einem Kontext $[U_{i_k}, S_{j_k}]$ aktiv ist, dann ist im darauffolgenden Zyklus $U_{i_{k+1}}$ im Kontext $[U_{i_{k+1}}, S_{j_{k+1}}]$ aktiv, und

(ii) nur U_{i_n} eine eventuelle Lernrückkoppelung von der Umwelt erhält.

Dann gilt: falls $E_{i_n}^{j_n}$ gegen einen konstanten Wert E^* konvergiert, dann konvergiert auch $E_{i_k}^{j_k}$, $1 \leq k < n$, gegen diesen Wert.

Beweis. Sei U_{i_k}, $1 \leq k < n$, während des Zyklus t aktiv. Dann folgt:

$$E_{i_k}^{j_k}[t+2] = E_{i_k}^{j_k}[t] - \alpha \cdot E_{i_k}^{j_k}[t] + \alpha \cdot E_{i_{k+1}}^{j_{k+1}}[t+1] \quad ,$$

wobei $E_{i_k}^{j_k}[t+2]$ den Wert von $E_{i_k}^{j_k}$ zu Beginn des Zyklus $t+2$ bezeichnet. Daraus läßt sich die Gleichung

$$E_{i_k}^{j_k}[\overline{s}+2] = (1-\alpha)^n \cdot E^{init} + + \textstyle\sum_{r=1}^{s} \alpha \cdot (1-\alpha)^{s-r} \cdot E_{i_{k+1}}^{j_{k+1}}[\overline{r}+1] \quad ,$$

ableiten, wobei $s \in \mathbb{N}$ und $\overline{s}$ definiert ist als $\overline{s} = t$ genau dann wenn U_{i_k} während des Zyklus t zum s-ten Mal im Kontext $[U_{i_k}, S_{j_k}]$ aktiv ist. Konvergiert nun $E_{i_{k+1}}^{j_{k+1}}$ gegen E^*, dann ergibt sich

$$\lim_{t\to\infty} E_{i_k}^{j_k}[t] = lim_{s\to\infty} E_{i_k}^{j_k}[\overline{s}+2] = E^* \; . \qquad \blacksquare$$

Unter dem DFG Algorithmus konvergieren also die Bewertungen aufeinanderfolgend aktiver Einheiten gegen ein Gleichgewichtsniveau. Unter Gleichgewichtsbedingungen zahlt dann jede Einheit an ihren Vorgänger denselben Betrag, den sie von ihrem Nachfolger erhält.

Entscheidend ist auch die Frage nach der Güte der gelernten Lösungen, wobei eine Lösung wie üblich definiert ist als jede Sequenz $S_1, \ldots, S_n$ von Umweltzuständen mit S_1 = Startzustand, S_n = Zielzustand und $S_i \neq S_n$ für alle $i \in \{1, \ldots, n-1\}$. Eine Antwort auf diese Frage liefert folgendes

Resultat 2 (Lösungsqualität). Jede unter dem DFG Algorithmus gelernte Lösung ist zyklenfrei.

Beweis. Sei $S_1, \ldots, S_n$ eine gelernte Lösung die nicht zyklenfrei ist. Dann gibt es ein $i \in \{1, \ldots, n-2\}$ und ein $k \in \{1, \ldots, n-i\}$ mit $S_i = S_{i+k}$. Somit ist in S_i und S_{i+k} dieselbe organisatorische Einheit aktiv. Daraus folgt unmittelbar $S_{i+1} = S_{i+k+1}$, oder allgemeiner, $S_{i+m} = S_{i+k+m}$ für jedes $m \in \{1, \ldots, n-i-k\}$. Somit ist $S_{n-k} = S_n$, und dies ist im Widerspruch zur Annahme, daß $S_1, \ldots, S_n$ eine Lösung darstellt. ■

Bemerkenswert an diesem Ergebnis ist, daß die „globale Eigenschaft" der Zyklenfreiheit erreicht wird, obwohl jeder einzelne Agent nur einen lokalen Ausschnitt des jeweiligen Umweltzustandes kennt und kein expliziter Wissenstransfer zwischen den Agenten erfolgt.

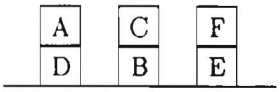
4.2. Experimentelle Ergebnisse

Um Erfahrungen mit dem DFG Algorithmus zu sammeln, wurde die blocksworld als Experimentierumgebung gewählt. Im folgenden werden die Ergebnisse zu der in Bild 3 gezeigten Aufgabe beschrieben. Jeder Agent ist in eine bestimmte Tätigkeit spezialisiert (z.B. kann Agent A_1 den Block A auf den Boden stellen und der Agent A_2 kann Block A auf Block B legen). Vorbedingung für die Ausführbarkeit einer Aktion $put(x, y)$ ist, daß keine anderen Blöcke auf x und y positioniert sind. Jeder Agent besitzt in jedem Umweltzustand nur minimales Wissen: er weiß nur, ob die Vorbedingung seiner Aktion erfüllt ist. Aufgrund dieser Einschränkung ist ein Agent nicht in der Lage, alle verschiedenen Umweltzustände zu differenzieren; vielmehr kann er nur zwischen der Klasse der Zustände, in denen er seine Aktion ausführen könnte, und der Klasse der Zustände, in denen dies nicht der Fall ist, unterscheiden (siehe auch [32, 33]). Diese Einschränkung erschwert die Aufgabe erheblich, da kein Agent zu irgendeinem Zeitpunkt einen „globalen Überblick" über die Blockkonstellation besitzt. Insbesondere ist es nun möglich, daß ein Agent nicht mehr in der Lage ist, zwischen Zuständen, in denen seine Aktion nützlich ist, und Zuständen, in denen sie nutzlos ist, zu unterscheiden. Je zwei Agenten gelten als kompatibel, wenn ihre Aktionen nicht folgende Bedingungen verletzen:

- ein Block kann nicht zugleich auf verschiedene Positionen gestellt werden,
- verschiedene Blöcke können nicht zugleich auf dieselbe Position gestellt werden und
- ein Block kann nicht auf einen anderen Block gestellt werden, dessen Position gerade verändert wird.

(Beispiele für inkompatible Agenten sind A_1 und A_2, A_3 und A_7, und A_5 und A_6.) Mehrere Agenten gelten kompatibel, wenn sie paarweise kompatibel sind.

Eine Analyse des Suchraums der in Bild 3 gezeigten Aufgabe zeigt, daß es nur eine einzige Lösung der Länge 2, 24 Lösungen der Länge 3, und 210 Lösungen der Länge 4 gibt. Keine Lösung enthält weniger als



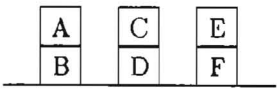
Startkonfiguration

Agenten:

A_1: $put(A, \perp)$ A_2: $put(A, B)$ A_3: $put(B, F)$

A_4: $put(C, \perp)$ A_5: $put(C, D)$ A_6: $put(D, A)$

A_7: $put(E, F)$ A_8: $put(F, \perp)$ A_9: $put(F, E)$



Zielkonfiguration

Zeitintervall: maximal 4 Zyklen

Bild 3: Eine vorgegebene Menge von Agenten soll lernen, innerhalb eines begrenzten Zeitintervalls Start- in die Zielkonfiguration zu transformieren.

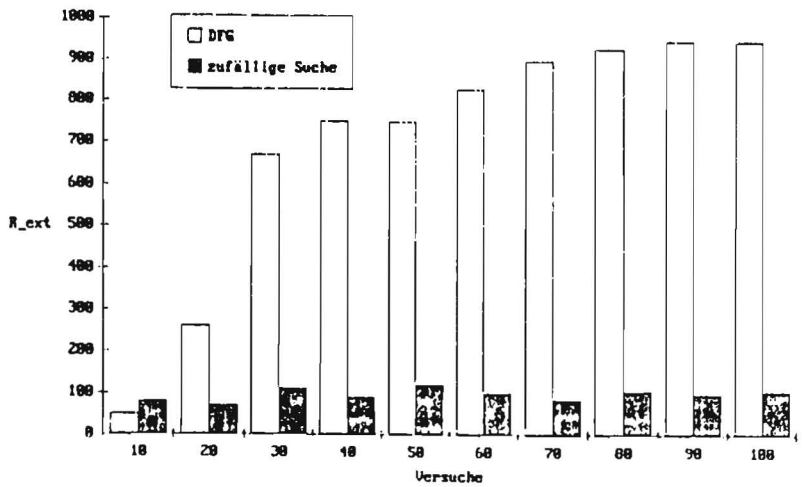


Bild 4: Lernperformanz des DFG Algorithmus.

5 Einzelaktionen, weshalb diese Aufgabe ohne Gruppenbildung nicht lösbar ist. Die Wahrscheinlichk daß eine zufällig gewählte und anwendbare Sequenz der Länge 2 (3, 4) die Aufgabe löst, ist geringe 1 (4, 5) Prozent; insgesamt es damit die Wahrscheinlichkeit, daß eine zufällig gewählte und anwendb Sequenz der Maximallänge 4 die Aufgabe löst geringer als 10 Prozent.

Bild 4 zeigt die Lernresultate des DFG Algorithmus für folgende Parameterkonstellation: $\alpha = 0$ $\beta \in [-\alpha/5 \ldots + \alpha/5]$ (zufällig generiert), $\nu = 4$, $\sigma = 1 + 3\alpha$, $\rho = 1 - \alpha$, und $E^{init} = R^{ext} = 1000$. (Wei experimentelle Resultate finden sich z.B. in [31].) Jeder Datenwert zeigt, gemittelt über 10 Läufe unterschiedlicher Initialisierung des Pseudozufallszahlengenerators, die in den jeweils letzten 10 Versuc erzielte durchschnittliche externe Lernrückkoppelung. Ein Versuch ist dabei definiert als jede Sequ bestehend aus maximal 4 Arbeitszyklen, welche zur Lösung der Aufgabe führt (erfolgreicher Versuch), jede Sequenz der exakten Länge 4, welche die Aufgabe nicht löst (erfolgloser Versuch). Nach jedem Vers wird die Startkonstellation erneut präsentiert. Die externe Lernrückkoppelung wird nur am Ende e erfolgreichen Versuches bereitgestellt. Wie die Lernkurve zeigt, liegt die erzielte Lernperformanz deut über der zufälligen Performanz. Dies illustriert insbesondere, daß der DFG Algorithmus mehreren Age ermöglicht, geeignete und stabile Lösungen zu lernen, obwohl jeder einzelne Agent nur ein sehr beschrän Wissen von seiner Umwelt besitzt.

5. Schlußbetrachtungen

Mit dem DFG Algorithmus wurde ein Verfahren zum Erlernen geeigneter Sequenzen von Aktionsmengen in Mehragentensystemen vorgestellt. Dieses Verfahren liefert eine breite Grundlage für weiterführende Untersuchungen im Umfeld des verteilten Lernens und der Aktionskoordinierung.

Ausgehend von den bisher gewonnenen Ergebnissen lassen sich folgende zentrale Themen und Fragestellungen hinsichtlich möglicher Erweiterungen des DFG Algorithmus ableiten:

- Der Lernerfolg des DFG Algorithmus ist an eine explizite Bewertung genügend vieler Umweltzustände Aktionsmengen gebunden. Welche in der Künstlichen Intelligenz entwickelten Methoden zur Generalierung, Wissensakquistion und Planung können dazu verwendet werden, diese Bindung zu mindern?
- Das dem DFG Algorithmus zugrundeliegende Gruppenkonzept impliziert eine strenge Hierarchie zwischen den Agenten. Welche alternativen und möglicherweise flexibleren Gruppenkonzepte sind anwendbar?
- Die Gruppenentwicklung erfolgt statisch nach fest vorgegebenen Kriterien. Welche alternativen und möglicherweise adaptiven Kriterien und Strategien für den Zerfall und die Bildung von Gruppen sind anwendbar?

Diese Fragestellungen müssen Gegenstand zukünftiger Forschung sein. Da bei der Beantwortung dieser Fragen weitgehend Neuland in der Künstlichen Intelligenz und in der Informatik betreten wird, ist es zweckmäßig und ratsam, auch einschlägige Literatur aus der Sozialpsychologie (z.B. [13, 19]) und den Wirtschaftswissenschaften (z.B. [1, 9, 14, 29] heranzuziehen.

Literatur

[1] Argyris, C., & Schön, D.A. (1978). *Organizational learning.* Addison Wesley.

[2] Bond, A. H., & Gasser, L. (Eds.). (1988). *Readings in distributed artificial intelligence.* Morgan Kaufmann.

[3] Brauer, W. (1991). The new paradigm of informatics. In H. Maurer (Ed.), *New Results and Trends in Computer Science*, Lecture Notes in Computer Science, Vol. 555, pp. 15–24. Springer.

[4] Brauer, W., & Brauer, U. (1992). Wissenschaftliche Herausforderungen für die Informatik: Änderungen von Forschungszielen und Denkgewohnheiten. In W. Langenheder, G. Müller, & B. Schinzel (Eds.), *Informatik Aktuell* (pp. 11–19). Springer.

[5] Brauer, W., & Hernández, D. (Eds.). (1991). *Verteilte Künstliche Intelligenz und kooperatives Arbeiten.* Springer.

[6] Brazdil, P., & Muggleton, S. (1991). Learning to relate terms in a multiple agent environment. In Y. Kodratoff (Ed.), *Machine learning — EWSL-91* (pp. 424–439). Springer.

[7] Erman, L. D., & Lesser, V. E. (1975). A multi–level organization for problem–solving using many, diverse, cooperating sources of knowledge. In *Proceedings of the 1975 International Joint Conference on Artificial Intelligence* (pp. 483–490).

[8] Fox, M.S. (1981). An organizational view of distributed systems. In *IEEE Transactions on Systems, Man, and Cybernetics* (Vol. SMC–11, No. 1, pp. 70–80).

[9] Galbraith, J.R. (1973). *Designing complex organizations.* Addison Wesley.

[10] Gasser, L., & Huhns, M. N. (Eds.). (1989). *Distributed artificial intelligence* (Vol. 2). Pitman.

[11] Georgeff, M. P. (1987). *Many agents are better than one.* Technical Report 417, SRI International, Menlo Park, CA.

[12] Grefenstette, J. J. (1988). Credit assignment in rule discovery systems based on genetic algorithms. *Machine Learning*, 3, 225–245.

[13] Guzzo, R.A. (Ed.) (1982). *Improving group decision making in organizations - Approaches from the and research*. Academic Press.

[14] Herriott, S. R., Levinthal, D., & March, J. G. (1985). Learning from experience in organizations. *American Economic Review*, Vol. 75(2), pp. 298–302.

[15] Hewitt, C. E. (1977). Viewing control structures as patterns of passing messages. *Artificial Intelligen 8(3)*, 323–364.

[16] Holland, J. H. (1985). Properties of the bucket brigade algorithm. In J. J. Grefenstette (Ed.), *Proc dings of the First International Conference on Genetic Algorithms and Their Applications* (pp. 1- Lawrence Erlbaum.

[17] Holland, J. H. (1986). Escaping brittleness: The possibilities of general-purpose learning algorith to parallel rule-based systems. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Mach learning: An artificial intelligence approach* (Vol. 2, pp. 593–632). Morgan Kaufmann.

[18] Huhns, M. N. (Ed.). (1987). *Distributed artificial intelligence*. Pitman.

[19] Laughlin, P. R. (1988). Collective induction: group performance, social combination processes, a mutual majority and minority influence. *Journal of Personality and Social Psychology*, Vol. 54(254–267.

[20] Lewe, H., & Krcmar, H. (1991). Groupware. In *Informatik-Spektrum*, Vol. 14, pp. 345–348 (I aktuelle Schlagwort).

[21] Malone, T. W. (1988). Organizing information processing systems: Parallels between human organi tions and computer systems. In W. Zachary, S. Robertson, & J. Black (Eds.), *Cognition, Cooperati and Computation*. Ablex Publishing Corp.

[22] Minsky, M. (1979). The society theory of thinking. In *Artificial intelligence: an MIT perspecti* (pp. 423–450). MIT Press.

[23] Petri, C. A. (1962). *Kommunikation mit Automaten*. Schriften des Instituts für Instrumentelle M thematik, Universität Bonn, Germany.

[24] Selfridge, O. G. (1959). Pandemonium: a paradigm for learning. In *Proceedings of the Symposium Mechanisation of Thought Processes* (pp. 511–529). Her Majesty's Stationery Office, London.

[25] Shaw, M. J., & Whinston, A. B. (1989). Learning and adaptation in distributed artificial intelligen In [10] (Gasser & Huhns, 1989, pp. 413–429).

[26] Sian, S. S. (1990). The role of cooperation in multi-agent learning. In S. M. Deen (Ed.), *Proceedi of the First International Conference on Cooperating Knowledge Based Systems*. Springer.

[27] Sian, S. S. (1991). Extending learning to multiple agents: issues and a model for multi-agent machi learning (MA-ML). In Y. Kodratoff (Ed.), *Machine learning - EWSL91* (pp. 440–456). Springer.

[28] Sian, S. S. (1991b). Adaptation based on cooperative learning in multi-agent systems. In Y. Demaze & J.-P. Muller (Eds.), *Decentralised AI* (Vol. 2). Elsevier.

[29] Sikora, R. & Shaw, M. (1990). *A double-layered learning approach to acquiring rules for financ classification*. Faculty Working Paper No. 90–1693, College of Commerce and Business Administrati University of Illinois at Urbana-Champaign.

[30] Weiß, G. (1992). Learning the goal relevance of actions in classifier systems. In B. Neumann (Ed *Proceedings of the 10th European Conference on Artificial Intelligence* (pp. 430–434). Wiley.

[31] Weiß, G. (1992). Action selection and learning in multi-agent environments. In *Proceedings of t Second International Conference on Simulation of Adaptive Behavior*.

[32] Weiß, G. (1993). Collective learning of action sequences. Erscheint in *Proceedings of the 13th Inte national Conference on Distributed Computing Systems*.

[33] Weiß, G. (1993). Learning to coordinate actions in multi-agent systems. Erscheint in *Proceedings the 13th International Joint Conference on Artificial Intelligence*.

Rollenverteilung unter gleichberechtigten Agenten

Klaus Fischer
Deutsches Forschungszentrum für Künstlichen Intelligenz (DFKI)
Stuhlsatzenhausweg 3
6600 Saarbrücken 11
e-mail: kuf@dfki.uni-sb.de

Zusammenfassung

Der Artikel diskutiert das Problem der Rollenverteilung unter gleichberechtigten Agenten. Dieses Problem tritt immer dann auf, wenn eine Menge von Aufgaben an eine Gruppe von gleichberechtigten Agenten bekanntgegeben wird, die diese Aufgaben kooperativ lösen sollen. Die Agenten sollen dabei selbst entscheiden können, welchen Teil von welcher Aufgabe sie übernehmen möchten. Ein Anwendungsbeispiel ist das im Projekt AKA-MOD am DFKI in Saarbrücken untersuchte Verladehofszenario, bei dem eine Menge von Gabelstaplern einen Lastwagen be- bzw. entladen sollen, wobei sie manche Be- und Entladevorgänge nur kooperativ ausführen können [Müller 93]. Ein anderes Beispiel dieses Problems tritt in einer flexiblen Fertigung auf, weil hier eine Instanz (das Fertigungsleitsystem) existiert, die zentral plant und Aufgaben bekanntgibt, die jeweils von einer Gruppe von Agenten (autonomen Systemen) in Kooperation gelöst werden müssen [Fischer 91, Fischer 92b]. Die autonomen Systeme sind selbständig arbeitende Einheiten, so daß es bei der Koordination bzgl. der Aufgaben zwischen den autonomen Systemen zu Konfliktsituationen kommen kann. Die Behandlung dieser Konfliktsituationen sind Gegenstand des vorliegenden Beitrages. Wenn in der Literatur über Konfliktlösung geschrieben wurde, dann wurde in vielen Fällen vorausgesetzt, daß die Agenten in der Lage sind, den Nutzen abzuschätzen, den sie aus einer bestimmten Aktion ziehen [Zlotkin 89, Zlotkin 90, Zlotkin 91a, Zlotkin 91b]. Im vorliegenden Fall ist dies nicht in sinnvoller Weise möglich, weil es nicht darum geht, einen Nutzengewinn für die Agenten zu erzielen. Es geht vielmehr darum, daß ein Agent die Entscheidungen eines anderen Agenten adaptieren muß, damit überhaupt ein Weiterarbeiten möglich ist. Andere Arbeiten untersuchen das Konfliktlösungsverhalten bei aus Menschen gebildeten Gruppen [Klein 91], aber auch diese Ansätze lassen sich nicht auf den beschriebenen Fall übertragen.

1 Einleitung

Der Artikel diskutiert das Problem der Rollenverteilung unter gleichberechtigten Agenten. Dieses Problem tritt immer dann auf, wenn eine Menge von Aufgaben an eine Gruppe von gleichberechtigten Agenten bekanntgegeben wird, die diese Aufgaben kooperativ lösen sollen. Die Agenten sollen dabei selbst entscheiden können, welchen Teil von welcher Aufgabe sie übernehmen möchten. Ein Anwendungsbeispiel ist das im Projekt AKA-MOD am DFKI in Saarbrücken untersuchte Verladehofszenario, bei dem eine Menge von Gabelstaplern einen Lastwagen be- bzw. entladen sollen, wobei sie manche Be- und Entladevorgänge nur kooperativ ausführen können [Müller 93]. Ein anderes Beispiel dieses Problems tritt in einer flexiblen Fertigung auf, weil hier eine Instanz (das Fertigungsleitsystem) existiert, die zentral plant und Aufgaben bekanntgibt, die jeweils von einer Gruppe von Agenten (autonomen Systemen) in Kooperation gelöst werden müssen [Fischer 91, Fischer 92b]. Die autonomen Systeme sind selbständig arbeitende Einheiten, so daß es bei der Koordination bzgl. der Aufgaben zwischen den autonomen Systemen zu Konfliktsituationen kommen kann. Die Behandlung dieser Konfliktsituationen sind Gegenstand des vorliegenden Beitrages. Wenn in der Literatur über Konfliktlösung geschrieben wurde, dann wurde in vielen Fällen vorausgesetzt, daß die Agenten in der Lage sind, den Nutzen abzuschätzen, den sie aus einer bestimmten Aktion ziehen [Zlotkin 89, Zlotkin 90, Zlotkin 91a, Zlotkin 91b]. Im vorliegenden Fall ist dies nicht in sinnvoller Weise möglich, weil es nicht darum geht, einen Nutzengewinn für die Agenten zu erzielen, es geht vielmehr darum, daß ein Agent die Entscheidungen eines anderen Agenten adaptieren muß, damit überhaupt ein Weiterarbeiten möglich ist. Andere Arbeiten untersuchen das Konfliktlösungsverhalten bei aus Menschen gebildeten Gruppen [Klein 91], aber auch diese Ansätze lassen sich nicht auf den beschriebenen Fall übertragen.

Prinzipiell ließen sich die im vorliegenden Artikel diskutierten Probleme auch durch das Einführen von zentralen Sperrmechanismen lösen, weil es sich im Kern um das Problem des gegenseitigen Ausschluß handelt. Zum einen ist dies aber keine natürliche Modellierung der Problemlösung und zum anderen würde dadurch ein synchroner Ablauf erzwungen, der einen hohen Grad an Ineffizienz mit sich bringen würde. Die Dekker'sche Lösung für das Problem des gegenseitigen Ausschluß, die bekanntlich ohne zentralen Sperrmechanismus auskommt, ist auch kein adäqutes Problemlösungsmittel, weil hier globale Variablen verwendet werden, die wiederum einer zentralen Instanz entsprechen. Darüberhinaus ist bei dieser Lösung über globale Variablen eine Rangfolge unter den Agenten festgelegt. Im vorliegenden Artikel soll aber genau die Rangfolge ausdiskutiert werden.

2 Problemstellung

Gegeben sei eine Menge von Agenten $\mathcal{A} = \{a_1, \ldots, a_n\}, n \in \mathbb{N}$, eine Menge von Aufgaben $\mathcal{T} = \{t_1, \ldots, t_m\}$.

Für Aufgaben sollen folgende Funktionen definiert sein:

$time : \mathcal{T} \longrightarrow I\!N$

Ergebnis ist der Zeitpunkt, zu dem die Aufgabe spätestens begonnen werden soll.

$duration : \mathcal{T} \longrightarrow I\!N$

Ergebnis ist der Erwartungswert der Dauer der Aufgabe.

$roles : \mathcal{T} \longrightarrow 2^{\mathcal{R}}$

Ergebnis ist die Menge der Rollenbeschreibungen der Aufgabe. Dabei ist

$$\mathcal{R} = \{r_1, \ldots, r_k\}, k \in I\!N$$

die Menge aller bekannten Aufgabenbeschreibungen.

Für Agenten sollen folgende Funktionen definiert sein:

$next : \mathcal{A} \longrightarrow \mathcal{T} \cup \perp$

Gibt an, welche Aufgabe der Agent als nächstes bearbeiten möchte. Wenn

$$next(a) = \perp$$

gilt, dann sucht der Agent gerade nach einer neuen Aufgabe oder die Menge $\mathcal{T}$ der Aufgaben ist leer.

$role : \mathcal{A} \longrightarrow \mathcal{R} \cup \perp$

Gibt an, welche Rolle ein Agent bei der Aufgabe, die er als nächstes ausführen möchte, übernehmen will. Es gilt:

$$\forall a \in \mathcal{A} : next(a) = \perp \Longrightarrow role(a) = \perp .$$

Ein Agent a kann eine Rollenbeschreibung r erfüllen ($a \approx r$), wenn a eine Menge von Skripten kennt, die es a erlauben, die durch r spezifizierte Teilaufgabe zu lösen. Für eine Aufgabe t ist durch die Menge:

$$team(t) = \{a \mid next(a) = t\}$$

ein Team definiert. $team(t)$ heißt vollständig, wenn gilt

$$\begin{aligned} \mid team(t) \mid &= \mid roles(t) \mid \wedge \\ \forall a_1, a_2 \in team(t) : role(a_1) &\neq role(a_2). \end{aligned}$$

Die Fragestellung, die nun diskutiert werden soll, ist nun: Gibt es ein Verhandlungsprotokoll, das es den Agenten für jede beliebige Menge von Aufgaben gestattet, eine gemeinsame Ausführungsreihenfolge für die Aufgaben zu bestimmen.

erzeugt wird entspricht einer vom Agenten an den Kommunikationspartner gesendeten Nachricht.

Stellen können durch $\rightarrow$, $—$ und $—\bullet$ mit einer Transition verbunden sein. Sind alle Stellen, die durch $\rightarrow$ bzw. $—$ mit einer Transition verbunden sind, mit einer Marke belegt, dann kann die Transition schalten. Beim Schalten wird von jeder Stelle, die mit $\rightarrow$ mit der Transition verbunden ist, eine Marke entfernt. Ist eine Stelle durch $—\bullet$ mit einer Transition verbunden, dann kann die Transition nur dann schalten, wenn sich auf der Stelle keine Marke befindet. Die Verbindung einer Stelle mit einer Transition kann mit einem Symbol beschriftet sein. In diesem Fall kann die Transition nur dann schalten, wenn der Term, der der Marke entspricht, mit dem Symbol übereinstimmt. Ist die Verbindung einer Transition mit einer Stelle mit einem Symbol beschriftet, so wird, wenn die Transition schaltet, das entsprechende Symbol als Marke auf die Stelle erzeugt. Gehen von einer Stelle mehrere Verbindungen zu Transitionen, so kann den Verbindungen eine Wahrscheinlichkeit p zugeordnet werden, die besagt, daß die entsprechende Transition mit der Wahrscheinlichkeit p schaltet. Einer Transition kann eine Zeit t zugewiesen werden, die besagt, daß der Schaltvorgang der Transition künstlich auf die Zeit t ausgedehnt wird.

Bei der Interaktion zwischen den Agenten wird in einem Agenten jeweils ein Interaktionsmuster mit einer Menge von Interaktionspartnern instantiiert, was bedeutet, daß die entsprechenden Verbidungen zwischen den $\lhd$ und $\rhd$ Stellen in den einzelnen Agenten automatisch richtig miteinander verbunden werden, wenn der Interaktionsvorgang beginnt.

Die EST-Netz-Darstellung der Protokollebene eines Interaktionsmuster wird verwendet, weil sie sich einerseits praktisch eins zu eins in eine Regelprogrammiersprache übersetzen läßt [Fischer 92a, Fischer 92b, Fischer 93]. Andererseits abstrahiert sie von den Details eines Regelprogramms und ist somit kompakter und intuitiv leichter verständlich. Nicht zuletzt besteht auch die Hoffnung, daß sich über die Theorie der Petrinetze Verifikationsmethoden für mit Hilfe von EST-Netzen beschriebene Verhandlungsprotokolle finden lassen [Starke 90].

Abbildung 1 zeigt das EST-Netz für das Interaktionsmuster $\mathcal{I}^E$. Unter folgenden Annahmen:

A–I Beide Agenten haben den Konflikt erkannt, d.h. R_1 wird irgendwann mit einer Marke belegt.

A–II Jede Transition, die in einem Agenten ausgeführt werden kann, wird auch irgendwann ausgeführt.

A–III $t >$ maximale Nachrichtenlaufzeit.

A–IV Ein Agent ist in der Lage, beliebig viele Interaktionsmuster parallel zueinander auszuführen.

kann gezeigt werden, daß mit diesem Interaktionsmuster immer eine Einigung zwischen zwei zueinander in Konflikt stehenden Agenten möglich ist, in dem Sinne,

daß einem der beiden Agenten eine Vorrangstellung eingeräumt wird. Der Agent, der die Vorrangstellung erhält, darf dann den bestehenden Konflikt aus seiner lokalen Sicht optimal lösen. Der "unterlegene" Verhandlungspartner muß diese Lösung übernehmen.

Lemma 1 *Unter den Annahmen A–I, A–II und A–III wird, falls in einem Agenten eine Marke auf Platz R_4 liegt, irgendwann eine Nachricht vom Verhandlungspartner eingehen.*

Der Beweis von Lemma 1 ist sehr technisch und in seiner Darstellung relativ lange, so daß er den Rahmen des Artikels sprengen würde.

Lemma 2 *$P(n, R_4)$ sei die Wahrscheinlichkeit, daß bei einem Einigungsprozeß n-Mal eine Marke auf den Platz R_4 gelangt, dann gilt:*

$$P(n, R_4) \leq p^{2n} + (1-p)^{2n}.$$

Es gilt weiter

$$0 < p < 1 \Longrightarrow \lim_{n \to \infty} P(n, R_4) = 0,$$

d.h. statistisch ist es unmöglich, daß ein Einigungsprozeß ein unbeschränkte Zeitspanne benötigt (auch wenn es im Einzelfall sehr lange dauern kann).

Beweis: Es gilt:

$$P(n, R_4) = \prod_{i=0}^{n} p_i^2, \text{ wobei } p_i = p \text{ oder } p_i = (1-p)$$

Weiter gilt:

$$p_i^2 \leq p^2 \text{ oder } p_i^2 \leq (1-p)^2$$

Daraus folgt:

$$P(n, R_4) \leq p^{2n} \text{ oder } P(n, R_4) \leq (1-p)^{2n}$$

□

Satz 1 *Unter den Annahmen A–I, A–II und A–III terminiert das Interaktionsmuster $\mathcal{I}^E$ mit dem Ergebnis, daß genau ein Agent die Marke* me *auf Platz R_6 erhält.*

Beweis: Die Behauptung folgt unmittelbar aus Lemma 1 und Lemma 2. □

Abbildung 2 zeigt das — im Vergleich zum EST-Netz des Interaktionsmusters $\mathcal{I}^E$ um einen ◇-Eingang und einen ◇-Ausgang erweiterte — EST-Netz des Interaktionsmusters $\mathcal{I}^{E_n}$. Der ◇-Ausgang eines EST-Netzes hat die Semantik, daß eine Marke, die auf diese Stelle erzeugt wird, automatisch auf allen ◇-Eingängen der in einem Agenten aktiven EST-Netze erscheint. Im Falle des Interaktionsmusters $\mathcal{I}^{E_n}$ bewirkt dies, daß der Einigungsprozeß mit einer Menge von Agenten mit dem Ergebnis terminiert, daß das Vorrangrecht dem Kommunikationspartner eingeräumt wird, sobald der Agent einem der Kommunikationspartner das Vorrangrecht abtreten muß.

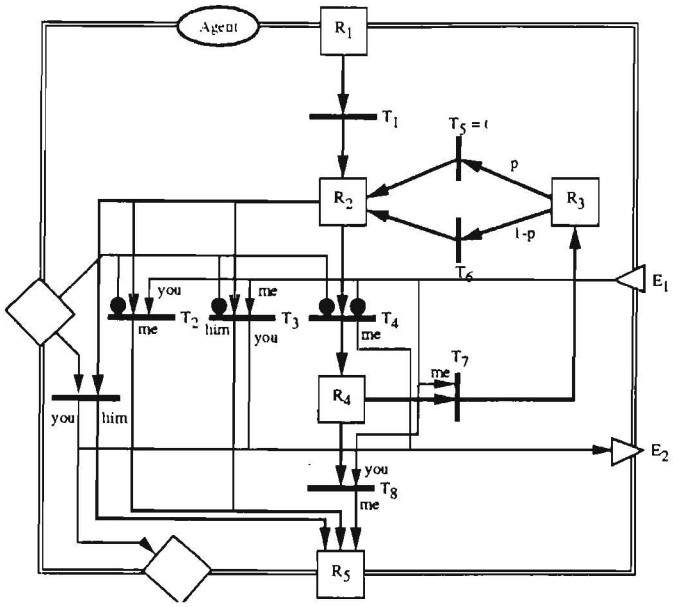


Abbildung 2: Interaktionsmuster zur Konfliktlösung zwischen n Agenten $\mathcal{I}^{E_n}$.

Satz 2 *Unter den Annahmen A-I, A-II, A-III und A-IV terminiert der mit Hilfe von Interaktionsmuster $\mathcal{I}^{E_n}$ zwischen n−Agenten geführte Einigungsprozeß mit dem Ergebnis, daß genau ein Agent die Marke* me *auf Platz R_6 erhält.*

Beweis: Der Beweis wird mit Induktion über n geführt.
$n = 2$: folgt unmittelbar aus Satz 1.
$n \longrightarrow n+1$: Satz 1 besagt, daß zwischen beliebigen zwei der $n+1$ Agenten nach endlicher Zeit eine Einigung erzielt wird. Damit scheidet einer der beiden Agenten aus der Menge der $n+1$ Agenten aus. Für die restlichen n Agenten gilt die Aussage wegen der Induktionsvoraussetzung. □

5 Initiierung einer Konfliktlösung

Für den im vorigen Abschnitt beschriebenen Einigungsprozeß war es wesentlich, daß beide Agenten den Konflikt erkannt haben und somit das Interaktionsmuster $\mathcal{I}^E$ instantiiert und aktiviert hatten. Ein Konflikt wird aber immer zunächst in einem Agenten lokal erkannt, ohne daß dieser davon ausgehen kann, daß der Agent, mit dem er in Konflikt steht, diesen Konflikt ebenso erkannt hat.

Das Problem, einen Einigungsprozeß zu initiieren, könnte durch einen zentralen Sperrmechanismus gelöst werden. Immer wenn ein Agent einen lokalen Konflikt erkennt, dann versucht er zentral eine Sperre für sich zusetzen, die es ihm als einzigem Agenten im System erlaubt, einen Einigungsprozeß zu initiieren. Erst wenn der Agent, mit dem er in Konflikt steht, in den Einigungsprozeß einwilligt, dann wird die zentrale Sperre wieder freigegeben. Durch die Sperre wird verhindert, daß bei der Initiierung eines Einigungsprozesses erneut eine Konfliktsituation entstehen kann.

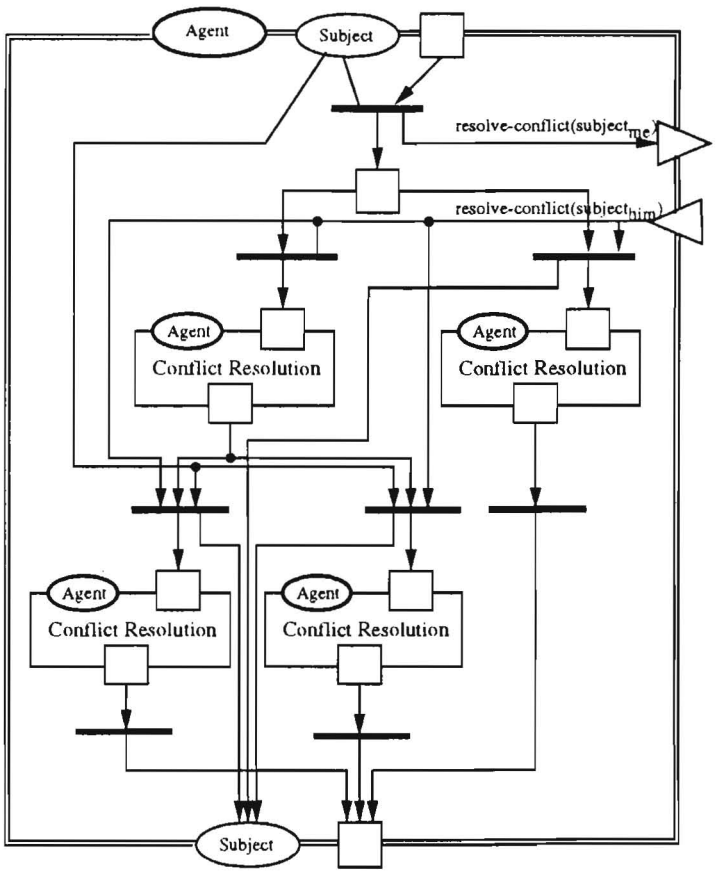


Abbildung 3: Interaktionsmuster zur Initiierung einer Konfliktlösung $\mathcal{I}^{I_1}$.

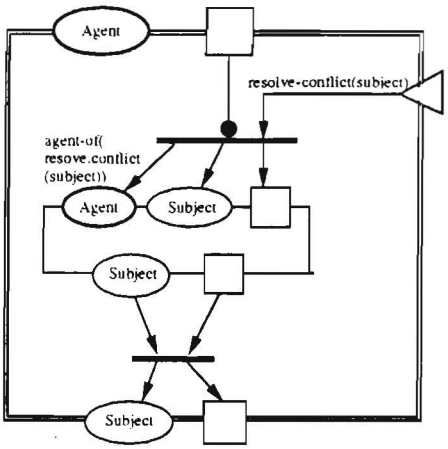


Abbildung 4: Interaktionsmuster zur Initiierung einer Konfliktlösung $\mathcal{I}^{I_2}$.

Obwohl diese Lösung zum gewünschten Ziel führt, hat sie doch den Nachteil, daß die Agenten ihre Aktionen sehr stark synchronisiert ausführen. Schlimmer noch, wenn ein Agent sehr stark belastet ist, so kann es unter Umständen relativ lange dauern, bis er in einen Einigungsprozeß, der von außen initiiert wird, einwilligt. Das bedeutet, daß in diesem Fall alle Agenten des Systems blockiert werden können, nur weil ein einziger Agent im Augenblick eine zeitaufwendige Berechnung durchführt, die ihn davon abhält, unmittelbar auf seine Umgebung zu reagieren.

Wenn man die Initiierung eines Eingungsprozesses ohne zentrale Sperre lösen möchte, besteht das Problem, daß man immer damit rechnen muß, daß die beiden miteinander in Konflikt stehenden Agenten zeitlich in etwa gleichzeitig den Einigungsprozeß initiieren. Dabei ist es im allgemeinen sogar möglich, daß die beiden Agenten bzgl. mehrerer Subjekte miteinander in Konflikt stehen und daß die Agenten den Einigungsprozeß für unterschiedliche Subjekte initiieren, so daß aus der Initiierung des Einigungsprozesses erneut eine Konfliktsituation entsteht. Glücklicherweise kann hier davon ausgegangen werden, daß beide Agenten das Eintreten dieses neuerlichen Konflikts erkennen und somit automatisch einen Einigungsprozeß bzgl. dieses Konflikts aktivieren.

Abbildung 3 zeigt das EST-Netz des Interaktionsmusters $\mathcal{I}^{I_1}$, mit dessen Hilfe ein Agent einen Einigungsprozeß initiieren kann. Abbildung 4 zeigt das Interaktionsmuster $\mathcal{I}^{I_2}$, mit dem ein Agent auf die Initiierung einer Konfliktlösung durch einen anderen Agenten reagieren kann.

6 Zusammenfassung und Ausblick

In dem Artikel wurde das Problem der Rollenverteilung unter gleichberechtigten Agenten diskutiert. Es wurde erläutert, daß eine solche Rollenverteilung ohne zentrale Steuerung nur dann möglich ist, wenn zwischen miteinander in Konflikt stehenden Agenten ein Einigungsprozeß möglich ist. Zunächst wurde hier gezeigt, daß es ein Verhandlungsprotokoll gibt, das einen Einigungsprozeß zwischen zwei Agenten beschreibt. Dann wurde gezeigt, daß dieses Verhandlungsprotokoll auf den Fall für n Agenten verallgemeinert werden kann. Am Ende wurde dann erläutert, daß ein Konflikt zwischen zwei oder auch mehreren Agenten immer in einem der Agenten zuerst festgestellt wird. Dieser kann nicht davon ausgehen, daß dieser Konflikt von allen anderen Agenten auch bereits erkannt wurde oder daß die anderen Agenten den Konflikt auf jeden Fall in der Zukunft erkennen werden. Aus diesem Grund wurde ein weiteres Verhandlungsprotokoll definiert, das es einem Agenten erlaubt, einen Einigungsprozeß mit einem oder mehreren anderen Agenten zu initiieren.

Bei der bisherigen Darstellung wurde die Entscheidungsebene bei der Beschreibung der Verhandlungsprotokolle völlig außer Acht gelassen. Dies liegt vor allem daran, daß die angegebenen Beweise im allgemeinen nicht geführt werden können, wenn die EST-Netze für den Einigungsprozeß um Funktionen erweitert wären, die den Einigungsprozeß beeinflussen. Hier liegt ein großes Potential für weiterführende Überlegungen.

Literatur

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

[Fischer 92a] Klaus Fischer und Hans-Michael Windisch, "MAGSY: Ein regelbasiertes Multiagentensystem", *KI*, 1/92, 1992.

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

[Fischer 93] Klaus Fischer, "The Rule-Based Multi-Agent System MAGSY", In *Proceedings of the '92 CKBS Workshop (forthcomming)*, 1993.

[Klein 91] Mark Klein, "Supporting Conflict Resolution in Cooperative Design Systems", *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6), November/December 1991.

[Kuhn 93] N. Kuhn, H. J. Müller und J. P. Müller, "Simulating Cooperative Transportation Companies", In *Proceedings of the European Simulation Multiconference (ESM-93)*, Lyon, France, June 1993, Forthcoming.

[Müller 93] J. P. Müller und M. Pischel, "Modelling Robot Societies using InteR-RaP", In *Working Notes of the IJCAI Workshop on Dynamically Interacting Robots*, Chambery, France, August 1993, Forthcoming.

[Starke 90] Peter H. Starke, *Analyse von Petri-Netz-Modellen*, B.G. Teubner, Stuttgart, Juli 1990.

[Werner 90] Eric Werner, "Distributed Cooperation Algorithms", In Yves Demazeau und Jean-Pierre Müller, Hrsg., *MAAMAW'91: Proceedings of the First European Workshop on "Modelling Autonomous Agents in a Multi-Agent World", August 16–18, 1989*, Seiten 17–31, 1990.

[Zlotkin 89] Gilad Zlotkin und Jeffrey S. Rosenschein, "Negotiation and Task Sharing Among Autonomous Agents in Cooperative Domains", In *Proceedings of the Eleventh International Joint Confernece on Artificial Intelligence*, Seiten 912–917, Detroit, Michigan, August 1989.

[Zlotkin 90] Gilad Zlotkin und Jeffrey S. Rosenschein, "Negotiation and Conflict Resolution in Non-Cooperative Domains", In *Proceedings of the National Conference on Artificial Intelligence, The American Association for Artificial Intelligence*, Seiten 100–105, Boston, Massachusetts, August 1990.

[Zlotkin 91a] Gilad Zlotkin und Jeffrey S. Rosenschein, "Negotiation and Goal Relaxation", In Yves Demazeau und Jean-Pierre Müller, Hrsg., *MAAMAW'91: Proceedings of the Second European Workshop on "Modelling Autonomous Agents in a Multi-Agent World", August 13–16, 1990*, 1991.

[Zlotkin 91b] Gilad Zlotkin und Jeffrey S. Rosenschein, "Cooperation and Conflict Resolution via Negotiation Among Autonomous Agents in Noncooperative Domains", *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6), November/December 1991.

Sich selbst organisierende Produktionsprozesse als Möglichkeit zur flexiblen Fertigungssteuerung

Werner Dilger, Stephan Kassel
European Business School, Schloß Reichartshausen
D-6227 Oestrich-Winkel

Zusammenfassung

Sich selbst organisierende Produktionsprozesse (SOPP) stellen eine Alternative zu bisherigen Verfahren der Werkstattsteuerung dar. Sie folgen damit zeitlich auf die Zeit- und Kapazitätsplanung von PPS-Systemen. Die grundlegende Idee der SOPPs ist, die am Fertigungsprozeß beteiligten Einheiten (flexible Fertigungszellen und Transporteinheiten) als autonom handelnde Agenten zu betrachten, die miteinander um die Durchführung von Fertigungsaufträgen konkurrieren. Das Ziel jedes Agenten ist es, die gesamte Wertschöpfung zu maximieren, und zwar durch Minimierung der Bearbeitungszeiten, Umrüstzeiten, Lager usw. Die Kommunikation zwischen den Agenten, die für die Organisation des Fertigungsprozesses erforderlich ist, läuft über einen speziellen Agenten, den Broker, der vor allem die Funktion eines Blackboards hat. SOPPs lassen sich nur in Fertigungsumgebungen anwenden, in denen Fertigungszellen mit identischen oder teilweise identischen Funktionen zur Verfügung stehen, d.h. in denen ein gewisser Grad an Parallelität vorliegt.

1. Einleitung

Multiagentensysteme sind für die Lösung komplexer Probleme vorgeschlagen und z.T. schon entwickelt worden. Sie erlauben es, die Problemlösungsfähigkeiten einzelner Systeme zu erweitern und sie steigern die Effizienz. In verschiedenen Branchen der fertigenden Industrie, insbesondere im Maschinen- und Anlagenbau, gewinnen flexible Fertigungssysteme (FFS) immer mehr an Bedeutung. Sie gestatten die Produktion relativ kleiner Losgrößen und die rasche Umstellung auf neue Produktlinien, entsprechend den Kundenwünschen. Flexible Fertigungssysteme basieren auf der Kooperation mehrerer flexibler Fertigungszellen und automatischer, d.h. fahrerloser Transportsysteme (FTS). Die Zellen und die Förderzeuge haben eigene Steuerprogramme, die die Aktivitäten der jeweiligen Einheit mit der Steuerung des Gesamtprozesses koordinieren. Sie haben damit bereits schon einen gewissen Grad an Autonomie.

Übliche Steuerungssysteme für FFS erfassen die Aktivitäten aller am Fertigungsprozeß beteiligten Einheiten und orientieren sich an einem globalen Plan. Der Hauptnachteil dieser Art der Steuerung ist der Bedarf nach Neu- und Umplanung, sobald eine Störung im Prozeß auftritt oder die Konfiguration der Fertigungsanlage geändert wird. Solche Ereignisse kommen in FFS häufig vor, deshalb wird relativ viel Zeit für Neu- und Umplanung verbraucht. Die *sich selbst organisierenden Produktionsprozesse* (SOPP) als alternativer Ansatz zur Steuerung von FFS haben diesen Nachteil nicht. Die grundlegende Idee der SOPP ist, den Prozeß ohne globalen Plan und ohne zentrale Kontrolle ablaufen zu lassen. Satt dessen werden die Einheiten des FFS als eine Gesellschaft von autonom handelnden Agenten betrachtet, die sich selbst so organisieren, daß die Ziele der Produktionsplanung erreicht werden. Dadurch hat SOPP einige evidente Vorteile gegenüber konventionellen Steuerungssystemen. Erstens ist kein globaler Plan und keine zentralisierte Steuerung erforderlich. Zweitens haben Stö-

rungen und Änderungen der Anlagenkonfiguration keinen Einfluß auf den Fertigungs zeß, solange für jede benötigte Funktion Fertigungseinheiten zur Verfügung stehen, d.h ist keine Umplanung erforderlich. Drittens kann sich die Organisation von Wartungst keiten an der aktuellen Belegung der Fertigungseinheiten orientieren.

Das SOPP-Konzept läßt sich nicht auf beliebige Fertigungen anwenden. Voraussetzun ein bestimmtes Maß an Parallelität, und zwar sowohl in der Fertigungsanlage als auch möglichen Produktionsverlauf, vgl. [Dilger et al. 92]

In den folgenden Abschnitten werden zunächst die Voraussetzungen an ein FFS angegeb unter denen der SOPP-Ansatz sinnvoll ist. Abschnitt 2 behandelt das Konzept des Brok der für den Informationsaustausch zwischen den Agenten zuständig ist. In Abschnitt 3 w den Modelle für die verschiedenen Typen von Agenten beschrieben. Die Wirkung der pr orientierten Verhaltensweise der einzelnen Agenten auf den Gesamtablauf ist in Abschn dargestellt. In Abschnitt 5 folgen Ideen zur Erweiterbarkeit des Ansatzes in Richtung e mehrstufigen hierarchischen Systems. Die Implementierung einer Testumgebung ist in schnitt 6 skizziert und in Abschnitt 7 wird der SOPP-Ansatz mit anderen Arbeiten ver chen.

2. Das Broker-Konzept

Eines der wichtigsten Probleme bei Multiagentensystemen ist die Organisation der Kom nikation zwischen den Agenten. In SOPP ist die Kommunikation mittels des Broker-K zepts organisiert. Der Broker ist ein spezieller Agent. Seine Aufgabe ist, für den Inform onsaustausch zwischen den anderen Agenten zu sorgen. Die Kommunikation zwischen Agenten erfolgt ausschließlich über den Broker, d.h. sie ist zentralisiert. Dadurch wird reicht, daß die Agenten sich nicht alle gegenseitig kennen müssen, sie müssen nur Broker kennen. Die Zahl der am Geschehen beteiligten Agenten braucht damit auch ni fest zu sein. Das entspricht der Tatsache, daß in FFS die Zahl der verfügbaren Produkti einheiten sich oft aufgrund von Ausfällen ändert. Zu sonstigen Eigenschaften des Brok Konzepts vgl. [Dilger et al. 92]

3. Die Agenten-Typen

Die Klasse der SOPP-Agenten besteht aus zwei Unterklassen. Die erste Unterklasse best aus den Agenten, die unmittelbar in das FFS involviert sind; sie werden deshalb *Produ onsagenten* genannt. Diese Klasse kann wiederum in zwei Klassen unterteilt werden, Klassen der *Fertigungsagenten* und der *Transportagenten*. Beide können weiter in verschie ne Typen unterteilt werden, von denen jeder mehrere Instanzen hat. Die zweite Unterkla der SOPP-Agenten, die sog. *Mittleragenten*, besteht ebenfalls aus zwei Unterklassen, Planungsagentenklasse und der Brokerklasse. Diese beiden Klassen haben jeweils nur e Instanz. Die Agentenhierarchie ist in Abbildung 1 dargestellt.

Der Informationsfluß zwischen den Agenten ist in Abbildung 2 dargestellt. Der Broker st im Zentrum des Informationsaustauschs und der Planungsagent dient als Bindeglied z schen dem Produktionsplanungssystem und dem Agentensystem über den Broker. In d folgenden Abschnitten werden die vier Agenten-Typen genauer dargestellt.

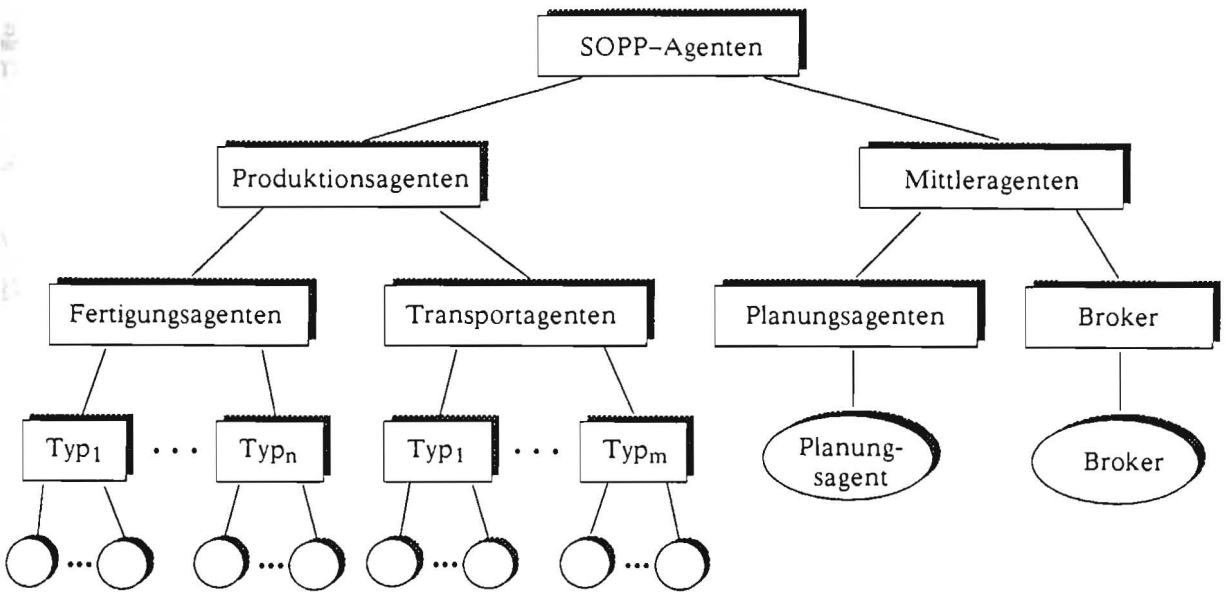


Abbildung 1: Die Hierarchie der SOPP-Agenten.

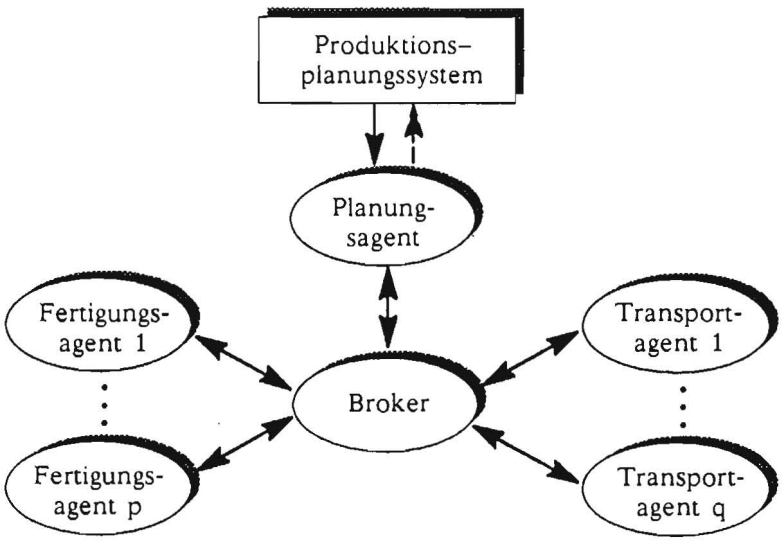


Abbildung 2: Der Informationsfluß zwischen den Agenten.

3.1. Der Planungsagent

Der Planungsagent bekommt vom Produktionsplanungssystem einen Grobplan des Produktionsprozesses als Eingabe. Dieser Plan enthält die spätesten Bearbeitungsendzeiten und Prioritäten für die Fertigungsaufträge. Für jeden Fertigungsauftrag sind in dem Plan alle notwendigen Schritte für die Produktion eines einzelnen Teils und ihre Reihenfolge angegeben. Diese Reihenfolge kann eine partielle Ordnung sein, falls irgendwelche Schritte nebenläufig durchgeführt werden können. Der Planungsagent bearbeitet den Produktionsplan und erzeugt Folgen von einzelnen Operationen. Dann wird die voraussichtliche Bearbeitungszeit plus der durchschnittlichen Rüstzeit, die späteste Anfangszeit und die späteste Endzeit der ersten Operation bestimmt. Der Planungsagent übergibt schließlich die so im Detail festgelegten Operationsfolgen an den Broker.

Während des Ablaufs des Fertigungsprozesses erhält der Planungsagent Informationen den Prozeßzustand vom Broker. Damit kann er die früheste Anfangszeit für die nachfol de Operation bestimmen. Dieser Zeitpunkt kann erst dann vollständig bestimmt wer wenn alle erforderlichen Materialien und Bauteile zur Verfügung stehen.

3.2. Der Broker

In Abschnitt 2 wurde bereits das Broker-Konzept beschrieben. Der Broker-Agent is Realisierung dieses Konzepts. Seine Struktur, dargestellt in Abbildung 3, spiegelt die

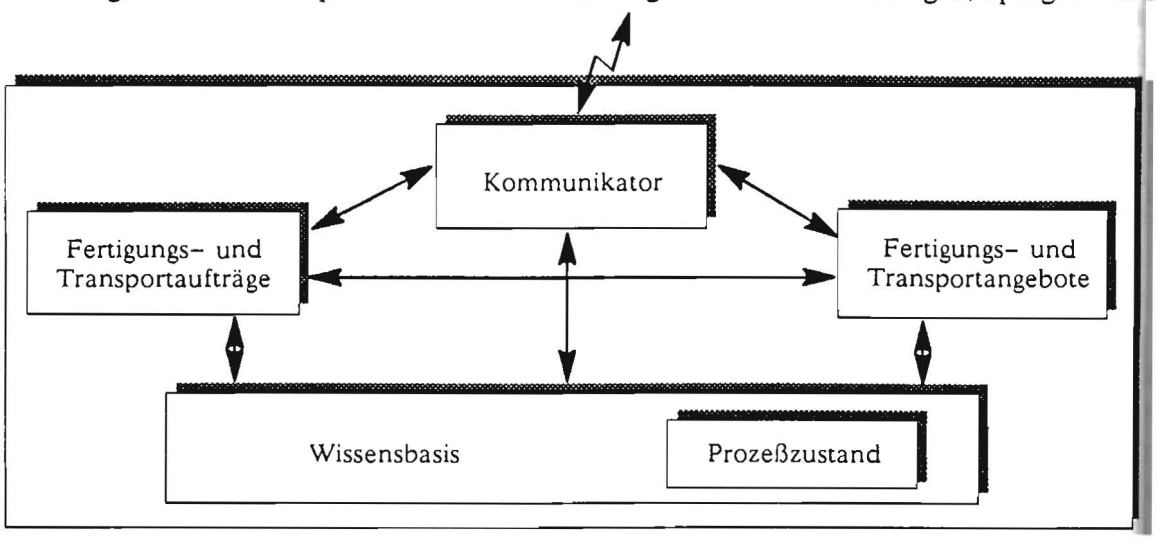


Abbildung 3: Die Struktur des Broker-Agenten.

fältigen Aufgaben des Brokers wider. Der Broker-Agent muß die Arbeitsaufträge des nungsagenten bearbeiten, die Angebote der Fertigungsagenten für verschiedene Bea tungsschritte und die Transportangebote der Transportagenten auswerten, er muß der tuellen Prozeßstatus anzeigen und er muß die ein- und ausgehenden Nachrichten von zu den anderen Agenten handhaben.

Der Broker-Agent nimmt die zentrale Stellung im SOPP-Prozeß ein. Seine Hauptaufg liegen im Verwalten der Werkstatt- und Transportaufträge und im Auswerten der zuge gen Angebote der Produktionsagenten. Der Prozeß der Angebotsabgabe wird genau Abschnitt 4 beschrieben.

3.3. Die Produktionsagenten

Die Grobstruktur der Produktionsagenten ist für beide Teilklassen gleich. Sie ist in A dung 4 dargestellt. Da die Fertigungsagenten untereinander in ihrem Verhalten ähnlich wird im Folgenden nur von *dem* Fertigungsagenten gesprochen; gleiches gilt für die T portagenten.

Der Fertigungsagent

Der Fertigungsagent spielt die Rolle eines Meisters in einer Werkstatt, der für eine Gr von Mitarbeitern und eine Anzahl von Maschinen verantwortlich ist und mit diesen zu men eine Fertigungseinheit bildet, die bestimmte Aufträge erledigen kann. Um diese

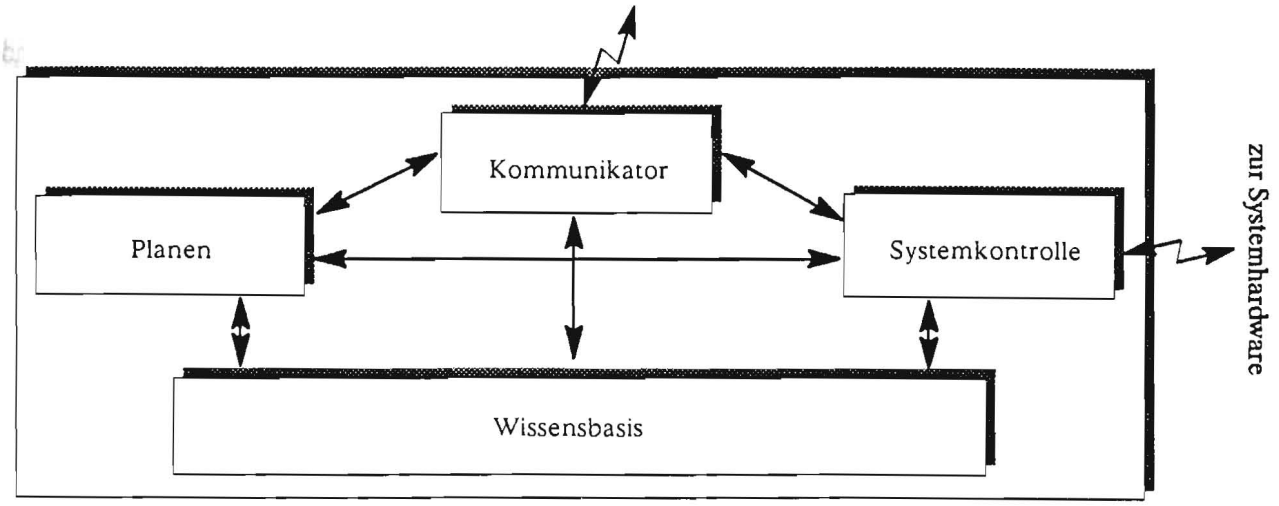


Abbildung 4: Die Struktur der Produktionsagenten.

spielen zu können, ist der Fertigungsagent aus mehreren Modulen zusammengesetzt, die unabhängig voneinander arbeiten.

Der Kommunikator besorgt die Kommunikation zwischen dem Agenten und dem Broker. Er erhält die Ankündigung von zur Bearbeitung anstehenden Aufträgen. Diese übergibt er an die Planungskomponente zum Zweck einer genauen Abschätzung der Kosten und der benötigten Zeit und zur Planung der einzelnen Arbeitsschritte auf den verfügbaren Maschinen. Das Ergebnis wird dem Broker durch den Kommunikator des Fertigungsagenten übermittelt.

Wenn der Fertigungsagent informiert wird, daß ein Auftrag für ihn reserviert ist, dann werden durch die Planungskomponente Transportaufträge generiert. Dazu müssen die für den Auftrag benötigten Werkzeuge, Materialien und Bauteile ermittelt und dementsprechend eine Anzahl von Transportaufträgen erzeugt werden. Jedem Transportauftrag wird der spätesten Anlieferungszeit versehen. Die Transportaufträge werden dem Broker durch den Kommunikator des Fertigungsagenten übermittelt.

Die Systemkontrolle des Fertigungsagenten ist das Bindeglied zwischen den für die Planung zuständigen Modulen und der Hardware der Fertigungszelle. Sie erhält über Sensoren Informationen über den aktuellen Zustand aller Teile der Fertigungszelle und steuert die Aktivitäten dieser Teile. Insbesondere ist sie für die korrekte Materialübergabe von den und an die Transportagenten, für die Qualitätskontrolle und für den Dialog mit dem menschlichen Bediener zuständig. Generell zeigt sie den Zustand des Fertigungsprozesses innerhalb der Zelle an, indem sie laufend die Wissensbasis auf den neuesten Stand bringt.

Der Transportagent

Die Aufgabe des Transportagenten ist, Güter zwischen verschiedenen Plätzen zu transportieren. Dazu muß er die Umgebung kennen, in der er sich bewegt, und er muß auf mögliche Störungen reagieren können. Für das SOPP-Konzept braucht er noch weitere Fähigkeiten. Er muß mit dem Broker kommunizieren, Transportaufträge entgegennehmen und Angebote für sie erstellen, d.h. hauptsächlich Kosten für Wege berechnen, und die Angebote dem Broker übermtteln.

Die Aufgabe der Planungskomponente ist, Wege zwischen verschiedenen Stellen des FF planen, die Kosten der Wege zu berechnen, Transportangebote zu generieren und übe Systemsteuerung die Hardware des Fahrzeugs zu steuern. Das dafür erforderliche W über die Koordinaten der Stellen, über verschiedene Wege und über die Kosten ist i Wissensbasis enthalten. Wenn eine Störung eintritt, muß der Planer sie untersuchen falls möglich, einen alternativen Weg vorschlagen.

Der Transportagent kann vier Zustände annehmen: *Ruhestellung*, *Leerfahrt*, *Lastfahr* *Störung*. Der Planer kontrolliert die Zustände des Agenten und wechselt von einem anderen, in Abhängigkeit von der zu erledigenden Aufgabe und der Umgebung. Ist z.B Agent im Zustand *Ruhestellung*, dann wechselt er in den Zustand *Leerfahrt*, falls er holen muß. Nach der Beladung geht er in den Zustand *Lastfahrt* über. Stößt er bei der F auf ein Hindernis, dann geht er in *Störung* über. Der Planer informiert den Kommuni über das Ereignis und dieser sendet eine Nachricht an den Broker und an den menschli Bediener.

4. Sich selbst organisierende Produktionsprozesse

Jeder SOPP–Agent verhält sich egoistisch. Dies bedeutet, daß jeder Agent nur daran int siert ist, seinen eigenen Gewinn zu steigern, ohne dabei die Ziele oder Wünsche der and Agenten zu berücksichtigen. Damit das Gesamtsystem funktioniert (im Sinne einer glob Optimierung des flexiblen Fertigungsprozesses), müssen die Ziele der Agenten so aufe der abgestimmt werden, daß sie zusammen gute Lösungen für das Fertigungsproble fern.

Der Planungsagent spielt bei diesem Geschehen nur eine sekundäre Rolle. Er bereite Fertigungsaufträge für die Verteilung vor, indem er alle fertigungstechnisch möglichen henfolgen für jeden Auftrag ermittelt und dann die jeweils ersten Schritte jeder dieser lichkeiten mit zugeordneten Bearbeitungsendzeiten an den Broker übergibt. (Fertig technisch möglich sind Arbeitsschritte gdw. alle Arbeitsschritte, die technisch vorher gefunden haben müssen, bereits abgearbeitet wurden.) Diese durchführbaren Schritte ßen Werkstattaufträge und die Menge der aktuell durchführbaren Werkstattaufträg zeichnen wir mit **W**. Auf der Menge der übergebenen Arbeitsschritte **W** wird unter Be sichtigung der frühesten Anfangs– und spätesten Endzeiten eine Rangordnung besti Diese Ordnung ist total und ordnet **W** in der Form $(W_1, W_2, \ldots, W_p)$ an mit W_1 als "wic stem" Auftrag.

Nach diesen Vorberechnungen beginnt die eigentliche Selbstorganisation:

Jeder Fertigungsagent, der in absehbarer Zeit einen neuen Werkstattauftrag bearb kann, sendet eine Nachricht an den Broker, der seine (Kommunikations–)Adresse, Beschreibung seiner Fähigkeiten und die frühestmögliche Anfangszeit für einen neuen trag beinhaltet.

Für jeden dieser Fertigungsagenten FA_k bestimmt der Broker eine Teilmenge von Werk aufträgen $\{W_{k1}, W_{k2}, \ldots, W_{kp(k)}| \subseteq \mathbf{W}$, die von diesem Agenten bearbeitet werden kö und am dringlichsten sind. Diese eingeschränkte Auftragsliste wird FA_k gesendet.

Ist die vom Broker gesendete Liste leer (keine passenden und dringlichen Aufträge), muß der Fertigungsagent einige Zeit warten und danach erneut versuchen, Aufträge zu erhalten. Ist die Liste nicht leer, berechnet er für jeden Werkstattauftrag W_{kj} in der übermittelten Liste die zu erwartende Zeit und die voraussichtlich entstehenden Kosten. Diese Daten werden als Angebot für diesen Auftrag dem Broker übermittelt.

Nach Ablauf einer vordefinierten Wartezeit wertet der Broker die eingegangene Angebote aus. Obwohl nur der dringlichste Auftrag W_1 an einen Fertigungsagenten FA_{W1} zugeteilt wird, werden für die Berechnung des günstigsten Angebots die ersten *n* Werkstattaufträge $W_1, W_2, \ldots, W_n$ auf einmal betrachtet. Dadurch erweitert sich der Planungshorizont des Brokers auf *n* Aufträge. Der Broker bestimmt *n*-Tupel von Fertigungsagenten ($FA_{j(1)}$, $FA_{j(2)}, \ldots, FA_{j(n)}$), wobei $FA_{j(i)}$ den Auftrag W_i bearbeiten soll. Von diesen *n*-Tupeln werden die *m* (m fest) errechnet, die sich in der ersten Komponente unterscheiden und die niedrigsten Gesamtkosten aufweisen. Die *m* Agenten, die den Auftrag W_1 in den selektierten *n*-Tupeln bearbeiten sollten, $FA_1, FA_2, \ldots, FA_m$, werden daraufhin informiert, daß W_1 vorläufig für sie reserviert ist.

Aufgrund dieser Meldung berechnen diese Agenten FA_i die benötigten Transportkapazitäten und erzeugen daraus Transportaufträge $\{T_{i1}, T_{i2}, \ldots, T_{ik(i)}\}$. Diese werden an den Broker zur Ausschreibung für die Transportagenten übergeben.

Sind alle Transportaufträge ausgewertet (d.h. an Transportagenten verteilt), so bestimmt der Broker den "besten" der *m* Fertigungsagenten und erteilt ihm den endgültigen Zuschlag für W_1.

Dieser Fertigungsagent FA_{W1} bereitet nun die Ausführung von W_1 vor. Sobald er mit der Bearbeitung von W_1 beginnt, sendet er eine Nachricht an den Broker mit der aktuellen Fertigungsanfangszeit. Diese wird dazu verwendet, die globale Datenhaltung des PPS-Systems zu unterstützen und es dem Planungsagenten zu ermöglichen, die frühest mögliche Anfangszeit für nachfolgende Werkstattaufträge zu bestimmen. Der Planungsagent kann dann nach dieser Berechnung die nachfolgenden Werkstattaufträge an den Broker zur Bearbeitung übertragen. Dadurch beginnt der Zuteilungszyklus von vorne.

Die Transportaufträge werden ähnlich wie die Werkstattaufträge behandelt. Jeder Transportagent, der in Bälde einen Transportauftrag ausführen kann, sendet an den Broker eine Nachricht, in der er mitteilt, was für Güter er laden kann, ob er be- oder entladen ist, und ab wann er einen Auftrag ausführen kann. An diese Nachricht wird automatisch vom Nachrichtenübermittlungssystem die symbolische Adresse des Transportagenten angefügt.

Der Broker bestimmt daraufhin aufgrund der Angaben des Transportagenten TA_i eine Teilmenge $\{T_{i1}, T_{i2}, \ldots, T_{in(i)}\}$ mit Transportaufträgen, die dieser Agent ausführen kann. Diese Teilmenge wird an den Agenten TA_i übermittelt.

Der Transportagent bestimmt für jeden Auftrag T_{ik} die Zeit, die er brauchen würde, um diesen Auftrag auszuführen. Danach gibt er Angebote für die für ihn günstigsten Transportaufträge ab. Es macht im allgemeinen wenig Sinn, wenn er mehrere Angebote für von einem Fertigungsagenten generierte Transportaufträge abgibt, da diese meist zeitlich paral-

lel liegen. Er muß sich also nur einen für ihn profitablen Auftrag pro Fertigungsagent a chen. Hingegen ist es sinnvoll, daß der Transportagent gleichzeitig Angebote für Tran aufträge verschiedener Fertigungsagenten abgibt, auch wenn diese zeitlich parallel li Diese Aufträge können nämlich von um einen Werkstattauftrag konkurrierenden gungsagenten ausgeschrieben worden sein. Von diesen konkurrierenden Fertigungsag kann aber nur einer den endgültigen Zuschlag für den Werkstattauftrag bekommen un für diesen werden dann die Transportaufträge auch ausgeführt.

Wenn nach diesem (ersten) Ausschreibungsschritt noch Transportaufträge ohne Ang eines Transportagenten existieren, werden diese in einem zweiten Schritt nochmals n ner höheren Priorität ausgeschrieben, was zu besseren Resultaten bei der Berechnung die Transportagenten führt und dadurch diese Aufträge attraktiver macht.

Der Broker wartet wieder eine vorgegebene Zeitspanne und wertet dann die abgege Angebote für jeden Transportauftrag aus. Wenn alle Transportaufträge, die von einer tigungsagenten stammen, ausgewertet worden sind, kann der Broker daraus endgül Güte der Angebote der Fertigungsagenten bestimmen. Sobald also alle Transportau aller *m* Fertigungsagenten, die um den Werkstattauftrag W_1 konkurrieren, ein Angeb halten haben, läßt sich der Fertigungsagent bestimmen, der (inklusive Transportzeite günstigsten fertigen kann. Dieser Fertigungsagent FA_1 erhält dann den endgültige schlag für W_1. Ebenso erhalten alle Transportagenten, die die besten Angebote für d FA_1 generierten Transportaufträge abgegeben haben, einen endgültigen Zuschlag.

Sobald die Transportagenten auf- bzw. abgeladen haben, senden sie eine Nachricht, u globale Datenbank des PPS-Systems auf den neuesten Stand zu bringen.

5. Erweiterung des Ansatzes auf mehrere Hierarchiestufen

Da eine flexible Fertigungszelle intern wieder aus mehreren Bearbeitungszentren und zelleninternen Materialflußsystem bestehen kann, kann die Notwendigkeit bestehe oben vorgestellten Ansatz zu erweitern.

Um eine solche Erweiterung durchzuführen, muß der Fertigungsagent intern die Ro Brokers und des Planungsagenten übernehmen, um in der flexiblen Fertigungsze Werkstattaufträge weiter in feinere Aufträge zu zerlegen, die innerhalb der einzelne beitungszentren der Zelle als elementar zu erledigende Arbeiten gehandhabt werde Fertigungsagent hat also eine Komponente, die die Werkstattaufträge in geeigneter zerlegen kann. Weiterhin muß er eine Komponente enthalten, die diese Subaufträge geeigneten Stellen vergibt. Dadurch ergibt sich intern dieselbe Rollenverteilung wie a Werkstattebene. Die Rolle der Fertigungsagenten auf Werkstattebene wird auf der Z bene von den Rechnern übernommen, die die einzelnen Bearbeitungszentren steue

Der größte Unterschied zwischen zelleninternem und auf Werkstattebene angesie *Prozeß besteht im verschieden gestalteten Materialflußsystem. Innerhalb der einzelne* tigungszelle gibt es nämlich in der Regel keine fahrerlosen Transportsysteme, vielme das zelleninterne Transportsystem wesentlich unflexibler und festgelegter im Bearbeite Transportanforderungen. Deshalb kann auch die Rolle der Transportagenten hier nich

ter an einzelne Agenten aufgeteilt werden, sondern es gibt nur einen Prozeß, der alle Transportanforderungen gemeinsam bearbeitet.

Für die Verknüpfung des zelleninternen SOPP mit dem Werkstatt-SOPP gibt es drei verschiedene Strategien: Zum einen könnte schon zum Zeitpunkt der Angebotsabgabe des Fertigungsagenten die interne Aufwandsabschätzung geschehen. Zum anderen kann die Abgabe der Angebote auf Schätzungen des Fertigungsagenten über die Kapazitäten der Zelle beruhen, und die konkrete Berechnung der eigentlichen Fertigungsdauer wird auf den Zeitpunkt der vorläufigen Reservierung eines Auftrags verschoben. Zum dritten wird die Berechnung sogar auf den Zeitpunkt des endgültigen Zuschlags für einen Auftrag verschoben.

Die erste der drei vorgeschlagenen Strategien ist sicher recht schwierig durchzuführen. Sie impliziert, daß für jeden überhaupt technisch fertigbaren Auftrag die konkrete Planung der zelleninternen Ressourcen durchgeführt wird. Dies führt aber wiederum dazu, daß diese Aufträge die Ressourcen auch (zumindest zeitweise) reservieren, um die Planungsergebnisse dann auch praktisch durchführen zu können. Damit blockiert sich das zelleninterne System selbst, da die internen Reservierungen erst viel später wieder zurückgenommen werden können, nämlich erst dann, wenn es endgültig feststeht, daß der Auftrag nicht durch diesen Fertigungsagenten bearbeitet wird.

Aus demselben Grund bietet sich auch die zweite Variante nur dann an, wenn die Zahl m der Agenten, die einen vorläufigen Zuschlag bekommen, so klein ist, daß der zelleninterne Planungsaufwand nicht zu groß ist. Da diese interne Planung schon zu einer Blockierung der entsprechenden Ressourcen für andere Aufträge führt, wird es auf jeden Fall schwierig, mehrere mögliche Fertigungsschritte parallel zu verarbeiten.

Die dritte Variante hat im Gegensatz zu den anderen den Nachteil, daß der Fertigungsagent im Fall eines internen Problems die Zusage der Fertigung erst in dem Augenblick verweigert, wenn er bereits einen endgültigen Zuschlag bekommen hat. Dies verlangt eine entsprechend höhere Flexibilität des Verhaltens des Brokers, der die Absagen an die anderen Fertigungsagenten erst dann senden kann, wenn die zelleninterne Verteilung der Aufgabe erfolgreich war und vom Agenten zurückgemeldet wurde.

6. Implementierung

Eine Testumgebung für SOPP wurde auf einer Siemens WS-30 Workstation unter UNIX implementiert. Als graphisches Window-System wurde X11, Release 3, mit Motif Window Manager verwendet. Die Implementierungssprache ist C++. Diese Sprache wurde gewählt, weil sie objektorientiert ist und die Features von C benutzt. Eine graphische Ausgabe der Ergebnisse in Form von Gantt-Charts wurde ebenfalls objektorientiert programmiert. Die Benutzung von C als der C++ zugrundeliegende Sprache vereinfachte die Programmierung der Kommunikationsroutinen für alle Agenten. Die in UNIX vorhandenen Möglichkeiten der Handhabung multipler Prozesse wurden genutzt, um eine Quasi-Parallelität der Agenten zu erreichen.

Jeder Agent ist als separater Prozeß implementiert. Die Prozesse werden vom Betriebssystem verwaltet. Um den Ablauf eines SOPP zu simulieren, wurde ein Master-Prozeß entwik-

kelt. Die Aufgabe des Masters ist, alle Agenten zu starten, eine Uhr zu simulieren und Benutzer die Möglichkeit zum Einbringen von Störungen zu geben. Die Produktionsag simulieren ihr Verhalten, indem sie den realen Fertigungs- bzw. Transportvorgang eine entsprechende Zeitverzögerung ersetzen. Diese Verzögerung wird durch den A der simulierten Uhr im Master-Prozeß gesteuert. Der Benutzer des Systems hat drei griffsmöglichkeiten: Erstens kann er über die (simulierte) Schnittstelle zum Produktion nungssystem neue Arbeitsaufträge in das System eingeben. Zweitens kann er neue Proc onsagenten hinzufügen oder vorhandene entfernen, was der sich verändernden Fertig umgebung entspricht. Drittens kann er die simulierte Zeit steuern, d.h. den Produktion zeß beschleunigen oder verlangsamen.

7. Vergleich mit anderen Ansätzen

SOPP ist im wesentlichen ein DcAI-Ansatz (Decentralized AI) [Castelfranchi 90], [D zeau/Müller 90], [Galliers 90]. Dieser Ansatz geht davon aus, daß eine Gesellschaf autonomen Agenten vorhanden ist. Jeder Agent existiert und agiert unabhängig vor anderen und hat seine eigenen Intentionen. Dasselbe gilt für die SOPP-Agenten. Als für die Kommunikation zwischen den Agenten wurde von Smith das *Contract Net* entw [Smith 88]. Nach diesem Modell kann jeder Agent die Rollen eines Managers und Vertragsnehmers spielen. In der Rolle des Managers muß der Agent eine gegebene Au in Teilaufgaben zerlegen und sie potentiellen Vertragsnehmern, die sich in der Lage s eine der Teilaufgaben zu lösen, bekannt machen. Der Manager wählt dann für jede Te gabe den günstigsten Anbieter aus und gibt ihm den Zuschlag. Ein Vertragsnehmer entweder die Teilaufgabe direkt lösen oder selbst die Rolle des Managers annehmen seine Teilaufgabe weiter zerlegen usw. Ein prototypisches System für die Fertigungss rung nach dem Contract Net-Modell wurde von Parunak entwickelt [Parunak 87].

Ein wesentlicher Nachteil des Contract Net ist, daß der Kommunikationsaufwand fü Lösung nicht trivialer Aufgaben sehr hoch ist. In SOPP ist die Kommunikation zentrali wodurch die Zahl der notwendigen Kommunikationskanäle vermindert wird. Ein we Nachteil des Contract Net (aus der Sicht von SOPP) ist, daß die Agenten sich gegen kennen müssen, zumindest teilweise. Deshalb ist es aufwendig, die Menge der verfügb Agenten zu verändern. In SOPP muß nur der Broker alle Agenten kennen, deshalb einfach, einen existierenden Agenten zu entfernen oder einen neuen hinzuzufügen.

Sycara et al. haben ein prototypisches Steuerungssystem entwickelt, bei dem die Ag zwar miteinander kommunizieren, aber nur in begrenztem Umfang [Sycara/Roth/S Fox 91]. Sie wissen über ihre Umwelt teilweise Bescheid und berücksichtigen sie bei Entscheidungen. Die Kommunikation ist vor allem zur Auflösung von Konflikten beim griff auf gemeinsame Ressourcen erforderlich. Die Ressourcen sind passive Einheiten keine Agenten, der Zugriff wird aber durch Agenten kontrolliert, die mit Überwach funktion ausgestattet sind. Diese bedienen die eingehenden Anforderungen unter Be sichtigung von Prioritäten nach dem first come first served-Prinzip. Im Sinne von SOP gemeinsame Ressourcen als autonome Agenten zu modellieren. Sie müssen die Int haben, die Vorgaben des Produktionsplans bei der Vergabe der Zugriffsrechte möglich zu erfüllen.

Als zentrale Stelle für den Informationsaustausch ist der Broker einem Blackboard ähnlich (vgl. die Artikel in[Engelmore/Morgan 88]). Die Wissensquellen, die auf einem Blackboard operieren, können mit den Agenten verglichen werden. Die Informationen im Broker sind strukturiert wie am Blackboard, allerdings nicht hierarchisch. Die Kontrollinstanzen des Blackboards geben ihm ein gewisses Maß an Autonomie, d.h. es ist mehr als ein reiner Datenspeicher. Die Aufgaben des Brokers gehen allerdings ein beträchtliches Stück über die üblichen Aufgaben eines Blackboards hinaus (vgl. Abschnitt 2). Er ist ein autonom handelnder Agent mit eigenen Intentionen und Entscheidungsfähigkeit, die den Fertigungsprozeß beeinflußt. Andererseits ist er aber kein Steuerungsorgan, das den anderen Agenten Anweisungen geben kann, und ähnelt damit wiederum dem Blackboard.

SOPP hat als Modell für die Fertigungssteuerung Ähnlichkeiten mit dem Kanban-Prinzip (vgl. [Browne/Harhen/Shivnan 88]) und mit der Belastungsorientierten Auftragsfreigabe (BOA) [Wiendahl 87]. Nach Kanban wird ein Produktionsprozeß durch die Entnahme eines fertigen Produkts aus einem Lager in Gang gesetzt. Dadurch werden nämlich Produktionsaufträge für das entnommene Produkt erzeugt, die ihrerseits ähnliche Wirkung haben wie die erste Entnahme. Kanban-Produktionsprozesse sind in starkem Maße selbst organisierend, wie SOPP-Prozesse. Kanban eignet sich aber am besten für Prozesse mit klar definierten Folgen von Arbeitsschritten, wie z.B. in der Fließfertigung, deshalb ist es weniger flexibel als SOPP. BOA dagegen ist eine zentralisierte Produktionssteuerung. Sie kontrolliert die Warteschlangen an den Fertigungszellen und entscheidet in Abhängigkeit von der Länge der Schlange an einer Zelle, wann die nächste Arbeitsgangfolge an die Zelle übergeben wird. Dadurch ist BOA gut für FFS geeignet. Da es ein zentralisierter Ansatz ist, kann es jedoch die Modularität von FFS (im Gegensatz zu SOPP) nicht genügend ausnützen.

8. Schluß

Wir haben ein System von autonomen Agenten beschrieben, die einen Fertigungsprozeß flexibel steuern können. Die Steuerung kommt dadurch zustande, daß jeder Agent seinen eigenen Intentionen folgt und keiner dem anderen Befehle erteilen kann. Deshalb kann ein so gesteuerter Fertigungsprozeß als selbst organisierend betrachtet werden. Wir haben die Struktur und Funktionsweise der einzelnen Agenten und des Gesamtprozesses beschrieben.

Das SOPP-Modell soll in verschiedene Richtungen weiterentwickelt werden. Zum einen ist geplant, die Hierarchisierung des Systems in mehrere Hierarchiestufen zu untersuchen. Interessant ist dabei insbesondere die Ausweitung nach oben in Richtung eines dezentralisierten PPS-Systems. Dies führt in das Gebiet des verteilten Scheduling. Zum anderen soll untersucht werden, ob der Broker selbst sinnvoll in mehrere Subprozesse zerlegt werden kann, die die Aufgaben des Brokers untereinander aufteilen und damit die Effizienz des Systems steigern.

Danksagung

Wir danken Alexander Cappius, Hans-Peter Güllich, Cristopher Herzfeld, Ralf Peters, Burkhard Schüttler und Frank Ziech für substantielle Beiträge zu dieser Arbeit.

Referenzen

[Browne/Harhen/Shivnan 88]
J. Browne, J. Harhen, J. Shivnan: Production management systems, a CIM perspe
Addison-Wesley Publ., Wokingham 1988.

[Castelfranchi 90]
C. Castelfranchi: Social power, a point missed in multi-agent, DAI, and HCI. In: Y. D
zeau, J.-P. Müller (eds.): Decentralized A.I. Proceedings of the First European Work
on Modelling Autonomous Agents in a Multi-Agent World. Cambridge, England,
1989; Elsevier Science Publ. B.V., Amsterdam, 1990, 49 - 62.

[Demazeau/Müller 90]
Y. Demazeau, J.-P. Müller: Decentralized Artificial Intelligence. In: Y. Demazeau,
Müller (eds.): Decentralized A.I. Proceedings of the First European Workshop on Mode
Autonomous Agents in a Multi-Agent World. Cambridge, England, Aug. 1989; Els
Science Publ. B.V., Amsterdam, 1990, 3 - 13.

[Dilger et al. 92]
W. Dilger, C. Herzfeld, S. Kassel, R. Peters: A system of competing agents for job
control. Proceedings of ICSC '92 - Second International Computer Science Confer
Hong Kong 1992, 453 - 459.

[Engelmore/Morgan 88]
R.S. Engelmore, A.J. Morgan (eds.): Blackboard systems. Addison-Wesley Publ., Wo
ham 1988.

[Galliers 90]
J.-R. Galliers: The positive role of conflict in cooperative multi-agents systems. In: Y
mazeau, J.-P. Müller (eds.): Decentralized A.I. Proceedings of the First European W
shop on Modelling Autonomous Agents in a Multi-Agent World. Cambridge, England,
1989; Elsevier Science Publ. B.V., Amsterdam, 1990, 33 - 46.

[Parunak 87]
H.V.D. Parunak: Manufacturing experience with the contract net. In: M.N. Huhns (
Distributed Artificial Intelligence. Pitman Publ., London 1987, 285 - 310.

[Smith 88]
R.G. Smith: The contract net protocol: High level communication and control in a di
uted problem solver. In: A.H. Bond, L. Gasser (eds.): Readings in Distributed Arti
Intelligence. Morgan Kaufmann Publ., San Mateo 1988, 357 - 366.

[Sycara/Roth/Sadeh/Fox 91]
K.P. Sycara, S.F. Roth, N. Sadeh, M.S. Fox: Resource allocation in distributed factory
duling. In: IEEE Expert, Febr. 1991, 29 - 40.

[Wiendahl 87]
H.-P. Wiendahl: Belastungsorientierte Fertigungssteuerung. Grundlagen, Verfahren
bau, Realisierung. Hanser-Verlag, München 1987.

Konzept einer aktionsflußbasierten verteilten Planung für Fertigungsumgebungen*

Joachim Draeger

TU München
Institut für Informatik
Orleansstr. 34
W-8000 München 80
email: draeger@informatik.tu-muenchen.de

Zusammenfassung

Es wird ein verteilter Algorithmus für Planungsaufgaben in einer Fertigungsumgebung vorgestellt, dessen Steuerung weitgehend implizit durch den sogenannten Aktionsfluß erfolgt. Das hier verwendete spezielle Konzept gewährleistet ein hohes Maß an Flexibilität, vermeidet dabei aber für Multiagentensysteme ohne zentrale Instanz typische Anomalien. Trotz hochgradiger Parallelität bleibt der notwendige Kommunikationsaufwand weitgehend begrenzt.

1 Einleitung

Fertigungsumgebungen werden immer häufiger als Multiagentensysteme modelliert [4, 8, 17]. Bei diesen unterscheidet man solche mit zentraler Steuerung und solche ohne. Während die Agenten im ersten Fall nur der Ausführung dienen, partizipieren sie im zweiten an Entscheidungen und an koordinierenden Maßnahmen. Obwohl infolge ihrer Einfachheit und Durchschaubarkeit manchmal bevorzugt, besitzen Multiagentensysteme mit zentraler Steuerung eine ganze Reihe von Nachteilen [9]. Der wohl schwerwiegendste betrifft die NP-Vollständigkeit vieler Planungsprobleme: Ab einer gewissen Problemgröße kann die gewünschte Lösung wegen der eintretenden kombinatorischen Explosion möglicher Varianten nicht mehr in einem akzeptablen Zeitraum erstellt werden. In dieselbe Richtung weist die zusätzliche Belastung der Zentralinstanz durch Störungen, welche infolge vieler eng miteinander verknüpfter Festlegungen meist nicht lokal behandelbar sind und mit zunehmender Systemkomplexität immer häufiger auftreten. Die so fortschreitend verlängerte Berechnungsdauer führt zu starken Einbußen hinsichtlich der Ergebnisqualität, da der reale Systemstatus immer mehr von dem der Planung zugrunde gelegten abweicht.

Zur Vermeidung derartiger Schwierigkeiten weicht man auf Multiagentensysteme ohne zentrale Komponente aus. Zahlreiche Berechnungsinstanzen helfen dort die Problemkomplexität zu dämpfen. Dies kann allerdings nicht zum Nulltarif erreicht werden: Weil die Agenten ihre Planungen aufeinander abstimmen müssen, erhöht sich der notwendige Kommunikationsaufwand. Darüber hinaus können je nach dem gerade verwendeten Verfahren verschiedene Anomalien auftreten. Parunak [14] nennt für das häufig benutzte Contract-Net Modell [3] insbesondere Zeit- und Lastanomalien. Zeitanomalien entstehen durch Entscheidungen der Agenten über

*Diese Arbeit wurde von der Deutschen Forschungsgemeinschaft (DFG) im Rahmen des Sonderforschungsbereichs 331 *Informationsverarbeitung in autonomen mobilen Handhabungssystemen* Teilprojekt Q4 gefördert.

Auftragsannahmen noch vor Eintreffen aller relevanter Daten. Dadurch verteilen sich die anstehenden Aufgaben mitunter sehr unvorteilhaft. Demgegnüber resultieren Lastanomalien aus der Wissensbeschränkung der einzelnen Agenten auf lokale Belange, wodurch man unter Umständen eine stark asymmetrische Lastverteilung erhält.

Die vorliegende Arbeit gibt einen Algorithmus zur verteilten Planung in Fertigungsumgebungen [6] an, der durch gleichberechtigte Beteiligung aller im System enthaltenen Agenten eine hohe Parallelität erreicht. Durch Aufteilung des Algorithmus in mehrere Phasen — Einbettung der Produktionsschritte in sogenannte Aktionen, Test auf Ausführbarkeit der Aktionen, Aufgabenverteilung, eventuelle dynamische Korrektur des erstellten Plans — vergrößert sich der Parallelitätsgrad nochmals [7]. Weil die einzelnen Phasen weitgehend unabhängig voneinander arbeiten, wird das Kommunikationsnetz kaum zusätzlich belastet. Infolge der Möglichkeit der Agenten, bei Bedarf die Produktion von zur Plandurchführung benötigten Hilfsmitteln und Bauteilen selbsttätig zu veranlassen, besitzt das betrachtete Modell ein hohes Maß an Flexibilität. Die zur Aufgabenauswahl verwendete Grundidee, Agenten ihre Eignung für einen Arbeitsschritt unter Einbeziehung von Informationen globaler Natur selbst bewerten und so lokal die Aktionsverteilung regeln zu lassen, vermeidet die oben erwähnten Zeit- und Lastanomalien. Der Algorithmus wird implizit durch in den einzelnen Aktionen enthaltenen Informationen gesteuert, die ihre Stellung im Gesamtplan beschreiben. Entsprechend den gängigen Begriffen 'Datenfluß' und 'Informationsfluß' — in Zusammenhang mit CIM sei an die GRAI-Methode [1, 2, 5] erinnert — wird dafür der Begriff *Aktionsfluß* eingeführt. Durch die Möglichkeit der dynamischen Korrektur erhält das System die Fähigkeit, einmal getroffene Entscheidungen veränderten Gegebenheiten anzupassen.

2 Das Modell

2.1 Agenten und Objekte

Die von uns betrachtete Fertigungsumgebung ist als Multiagentensystem modelliert und wird mit der Menge der in ihr enthaltenen Agenten $\mathcal{A}$ identifiziert. Jedem Agenten $a \in \mathcal{A}$[1] ist ein Tupel $(P_a, \Delta_a, O_a, A_a, E_a, D_a, B_a)$ von Variablen zugeordnet. Die Agenten selbst sind als spezielle endliche Automaten formalisiert [10]. Um deren Aufbau besser verstehen zu können, machen wir zunächst einige Anmerkungen zu den passiven Elementen einer Fertigungsumgebung, den sogenannten Objekten. Diese werden in der Menge O zusammengefaßt. Beziehungen der Gestalt 'Agent a kontrolliert das Objekt o' — man schreibt dafür auch $o \in O_a$ — führen in natürlicher Weise eine Zuordnung zwischen Objekten und Agenten ein. Für diese gelte $O = \bigcup O_a$. Die so eingeführten notationellen Hilfsmittel gestatten eine Beschreibung der Übergangsrelation Δ_a eines Agenten a als $Z \times \mathcal{P}(O) \rightarrow Z \times \mathcal{P}(O)$; $\mathcal{P}$ bezeichnet dabei die Potenzmengenbildung. Die Elemente $(z, \tilde{O}, z', \tilde{O}')$ von Δ_a, sogenannte Aktionen, modifizieren den gerade vorliegenden Agentenzustand z zu z' und überführen gleichzeitig eine Menge von Objekten $\tilde{O}$ in eine neue Menge $\tilde{O}'$. Ein Übergang $(z, \tilde{O}, z', \tilde{O}')$ ist dabei nur möglich, wenn sich a im Zustand z befindet und außerdem $\tilde{O} \subseteq O_a$ gilt; nach Ausführung der Aktion ergibt sich $O_{a\,\mathrm{neu}} := (O_{a\,\mathrm{alt}} \setminus \tilde{O}) \cup \tilde{O}'$.

Die zur Verfügung stehenden Produktionspläne P sind wie die Objekte in Form von Teilmengen $P_a \subseteq P$ auf die einzelnen Agenten verteilt. Vereinbarungsgemäß sei $p \in P_a$ nur möglich, wenn a den letzten Produktionsschritt von p ausführen kann. Alle P_a sind ebenso wie die Δ_a zeitlich invariant. Die Mengen A_a, E_a und

[1] Grundsätzlich sind alle hier verwendeten Mengen als Multimengen aufzufassen; alternativ kan man sich auch durch die Einführung von Identifikatoren behelfen.

D_a enthalten zur Ausführung durch den Agenten a bestimmte Aktionen $\delta \in \Delta_a$, die um zusätzliche Informationen ergänzt sind. Während die A_a der Aufstellung, der Bewertung und der Auswahl von Plänen dienen, gehören die in E_a und D_a gespeicherten Aktionen zu fertigen Plänen. Die Mengen E_a unterscheiden sich von den D_a durch die Möglichkeit einer nachträglichen Korrektur. Beiden gemeinsam ist die Existenz einer Ordnung auf ihren jeweiligen Elementen, induziert durch eine sich aus Bewertungsinformationen ergebende zeitliche Einordnung dieser Aktionen. Die B_a sind Arbeitsspeicher für Planbewertungen.

2.2 Pläne

Zur Beschreibung der Arbeiten, die vom System durchzuführen sind, dienen sogenannte Pläne. Im Zusammenhang mit Fertigungsumgebungen von besonderer Bedeutung sind dabei Darstellungen von Produktionsabläufen; diese erfolgen in Produktionsplänen, welche aus der Literatur wohlbekannt sind (siehe etwa [11]). Sie enthalten alle notwendigen Informationen hinsichtlich der vorzunehmenden Objektverarbeitung; darüber hinausgehende Daten (etwa in Bezug auf Transporte oder Werkzeugeinsatz) fehlen allerdings. Solche finden sich erst in den detaillierteren Aktionsplänen, die man als verfeinerte Produktionspläne ansehen kann. Formal gesehen sind sowohl Produktionspläne als auch Aktionspläne zusammenhängende und zyklenfreie Bedingungs/Ereignis-Petrinetze [15], wobei für Produktionspläne noch zusätzlich gefordert wird, daß die Stellen des zugrundeliegenden Netzes weder vorwärts- noch rückwärtsverzweigend sind und die Zielstellenmenge — d.h. die Menge der Stellen mit leeren Nachbereich — nur aus einem Element besteht. Alle Stellen eines Planes sind mit Objekten beschriftet. Die Menge der Transitionen T_p eines Planes p gewinnt man durch die Funktion d, d.h. $T_p = d(p)$. Die Transitionen eines Produktionsplans werden als Produktionsschritte bezeichnet, die eines Aktionsplans als Aktionen.

Entsprechend dem Grundgedanken des hier vorgestellten Algorithmus verteilt das System die Produktionsschritte eines Produktionsplanes in Form von Aktionen an die einzelnen Agenten; dort findet dann eine Ergänzung zu einem vollständigen Aktionsplan statt. Ein Problem stellt dabei der Wechsel der Betrachtungsweise dar [16]: Während Produktionspläne und die aus ihnen entstehenden Aktionspläne einen Fertigungsprozeß o.a. in seinem kausalen Zusammenhang beschreiben, ist nach erfolgter Verteilung für den einzelnen Agenten a statt dessen die Verknüpfung von Aktionen unterschiedlicher Pläne zu einem Art persönlichen Arbeitsschema von Bedeutung. Die damit verbundenen Schwierigkeiten sind nichttrivialer Natur: a kann nach der Aktion $\delta \in \Delta_a$ die Aktion $\delta' \in \Delta_a$ nur ausführen, wenn der Zielzustand von δ mit dem Startzustand von δ' übereinstimmt; ansonsten muß a durch Zwischenschaltung weiterer Aktionen eine Zustandstransformation vornehmen. Abgesehen von der Möglichkeit, dabei die Menge O_a in unvorhergesehener Weise zu verändern, ist die Existenz einer derartigen Transformation keineswegs immer gesichert. Mit Rücksicht auf die Verständlichkeit des Algorithmus treffen wir daher die folgenden vereinfachenden Vereinbarungen:

- Die Zustandsmenge jedes Agenten enthält mindestens ein ausgezeichnetes Element, das man als Grundzustand bezeichnet.
- Für je zwei Grundzustände eines Agenten a gibt es für jede Gestalt der Menge O_a eine verbindende Folge von Aktionen, die O_a invariant läßt.
- Für je zwei Aktionen eines Agenten a, die weder die Menge O_a verändern noch an sie irgendwelche Anforderungen stellen, gibt es eine verbindende Folge von Aktionen mit denselben Eigenschaften hinsichtlich O_a.

- Alle Aktionen $\delta \in \Delta_a$, die Hilfsmittel zu ihrer Durchführung benötigen, werden mit den entsprechenden vor- und nachbereitenden Aktionen zu sogenannten Rüstplänen zusammengefaßt und eventuell in mehreren Varianten in den Mengen $\tilde{P}(\delta, a)$ zur Verfügung gestellt. Ist für δ und a ein $\tilde{P}(\delta, a)$ definiert, fordert man zusätzlich $\tilde{P}(\delta, a) \neq \emptyset$. Alle Rüstpläne $\tilde{p} \in \tilde{P}(\delta, a)$ beginnen und enden in Agentengrundzuständen.

Ein weiteres Problem stellen Objektbelegungen durch nebenläufige Pläne dar. Besonders evident ist dies bei Objekten, die in irgendeiner Weise weiterverarbeitet werden. Hier schließt ein spezielles Verfahren zur Markierung von Verbrauchsobjekten deren Mehrfachverwendung aus. Doch auch Objekte, die die Agenten lediglich benutzen, verlangen zusätzliche Überlegungen. Prinzipiell besteht nämlich die Möglichkeit, daß das betrachtete Objekt infolge einer verwendungsbedingten Ortsveränderung wegen nun fehlender Transportpläne einer weiteren Nutzung entzogen ist. Um dies auszuschließen, gelte die folgende Symmetriebedingung: Kann das System ein als Hilfsmittel benutztes Objekt von einem Ort x zu einem Ort y transportieren, muß auch ein Transport von y nach x möglich sein. Formal läßt sich dies als Bedingung an die Objektzustände formulieren. Damit vermeiden wir prinzipielle Inkonsistenzen zwischen dem realen und dem der Planung zugrundeliegenden Systemstatus.

2.3 Planbewertung

Die Bewertung eines Aktionsplans erfolgt durch Zuordnung eines Tupels $\omega \in V$ mit $V \subseteq \mathsf{R}^n$, $n \in \mathsf{N}$. Um ein Vergleich von Bewertungen zu ermöglichen, definiert man auf V eine Halbordnung. Durch Projektionen auf Teilräume von V sind Informationen wie der zur Ausführung des Plans benötigte Zeitbedarf, die dabei anfallenden Betriebskosten, der Rohstoffverbrauch u.a. direkt von ω ablesbar. Die Berechnung einer Planbewertung erfolgt ausgehend von den Einzelaktionen. Die Beschreibung des betrachteten Aktionsplans mit Hilfe einer Grammatik [10] liefert eine formale Definition der Planstruktur. Man faßt nun die Bewertungen der einzelnen Aktionen als Attribute [12] der in der Grammatik vorhandenen Terminale auf. Abhängig von der jeweilig angewandten Ableitungsregel, die die lokale Planstruktur beschreibt, berechnet das System daraus die Bewertungen der Nonterminale bis hinauf zum Startsymbol. Die diesem zugeordnete Bewertung stellt die Bewertung des gesamten Aktionsplanes dar. Um die Auswahl von Plänen hinsichtlich ihrer Bewertung mit der Komposition von Teilplänen verträglich zu machen, treffen wir die folgende Vereinbarung: Besitzen zwei Pläne p, p' die Bewertungen $\omega(p), \omega(p')$ mit $\omega(p) \leq \omega(p')$ und tragen die aus ihnen durch Verlängerung um dieselbe Aktionenfolge entstehenden Pläne $\tilde{p}, \tilde{p}'$ die Bewertungen $\omega(\tilde{p}), \omega(\tilde{p}')$, so gelte $\omega(\tilde{p}) \leq \omega(\tilde{p}')$.

3 Der Algorithmus

Im Prinzip kann jeder Agent des beschriebenen Multiagentensystems einen Fertigungsauftrag initiieren. Die Anforderung des zu produzierenden Objekts o durch einen Agenten $a \in \mathcal{A}$ erfolgt dabei entweder intern durch einen bereits existierenden Plan, der o zu seiner Ausführung benötigt, oder extern durch manuellen Eingriff. In beiden Fällen startet a zur Generierung geeigneter Produktionspläne unmittelbar nach der Beauftragung die Methode ***prodplan***. Als Parameter werden das Objekt o und der anfordernde Agent a übergeben. Das so eingeleitete Planungsverfahren wird im folgenden ausführlich geschildert. Dabei benutzen wir die in Tabelle 1 aufgeführte Notation.

postprodacts(δ)	Die Menge der Nachfolgeaktionen $\delta^{\cdot\cdot}$ hinsichtlich des zugrundeliegenden Produktionsplans
preactions(δ)	Die Menge der vorbereitenden Aktionen für δ
postactions(δ)	Die Menge der nachbereitenden Aktionen für δ
preacts(δ)	Die Menge der Vorgängeraktionen von δ, die nicht vorbereitend sind
postacts(δ)	Die Menge der Nachfolgeraktionen von δ, die nicht nachbereitend sind
postends(δ)	Die Menge der Aktionen, die die Folgen nachbereitender Aktionen von δ abschließen

Tabelle 1:

3.1 Produktionsschritte als Aktionen

In *prodplan* wird zunächst sichergestellt, daß sich alle Agenten an der Entwicklung des gewünschten Produktionsplans beteiligen. Dazu prüft *self*, ob er mit dem Agenten a identisch ist, der das Objekt o anfordert. Trifft das zu, veranlaßt er alle anderen außer ihm im System vorhandenen Agenten $a' \in \mathcal{A} \setminus \{\text{self}\}$ ebenfalls zum Start der Produktionsplanung; andernfalls erübrigt sich eine Benachrichtigung durch *self*, da dies bereits der Agent a übernommen hat. Um die Verfügbarkeit des durch *self* zu produzierenden Objekts für a zu gewährleisten, ist im Falle *self* $\neq a$ stattdessen die Entwicklung eines zusätzlichen Transportplans notwendig. Dazu ruft *self* die Methode *moveplan* auf. Kann diese keinen geeigneten Transportvorgang in Gestalt eines entsprechenden Aktionsplans abliefern, ist die Bereitstellung des Objekts o an der benötigten Stelle in Frage gestellt und die weitere Arbeit des Agenten *self* an einem Produktionsplan für o somit sinnlos; in diesem Fall bricht die Methode *prodplan* ab. Nach dieser Vorbereitungsphase wird durch Beschränkung der lokalen Produktionsplanmenge P_{self} auf eine Teilmenge $P' \subseteq P_{\text{self}}$ bestehend aus Produktionsplänen für o der eigentliche Planungsvorgang eingeleitet. Es erfolgt das Aufbrechen aller $p \in P'$ in einzelne Produktionsschritte $s \in d(p)$ und deren Verteilung an die Agenten mittels der Methode ***distribute***. Wie ***distribute*** arbeitet, wird weiter unten näher erläutert. Es ist zu beachten, daß das geschilderte Verfahren in der Regel ein s mehreren Agenten zuteilt; eine konkrete Zuordnung zwischen Produktionsschritten und ausführenden Agenten findet erst im Rahmen der späteren Planauswahl statt. Nicht verteilbare Produktionsschritte s führen zum Ausschluß der betreffenden Produktionspläne p aus allen weiteren Betrachtungen. Dies beinhaltet sowohl die Entfernung von p aus der Menge P' als auch die Löschung aller bereits separierten Schritte von p aus den Mengen $A_{a'}$. Wird durch das voranstehende Verfahren P' zur leeren Menge reduziert, bricht *self* den Planungsvorgang ab; dasselbe gilt natürlich auch für den Fall, daß die Menge P' schon bei ihrer Bildung leer war.

method prodplan(self, $o \in O, a \in \mathcal{A}$)
if self= a
then for $a' \in \mathcal{A} \setminus \{a\}$ **do** sendmessage(a',prodplan,o, a) **endfor**
else sendmessage(self,moveplan,o,self $\rightarrow a$);
  if no moveplan **then exit fi**
fi;
$P' := \{p \in P_{\text{self}} \mid p \text{ produces } o\}$;
for $p \in P'$ **do for** $s \in d(p)$ **do**
  for $a' \in \mathcal{A}$ **do** sendmessage(a',distribute,s,p) **endfor**;
  if s not distributed **then** remove(p); **exitloop fi**
endfor endfor
if $P' = \emptyset$ **then exit else fi**

Nun zu der für die Verteilung der Produktionsschritte s verantwortlichen Methode *distribute*. In ihr überprüft der Agent *self* jede in Δ_{self} enthaltene Aktion $\delta \in \Delta_{\text{self}}$ auf die Möglichkeit, durch sie den Schritt $s \in T$ realisieren zu können. Dies ist genau dann möglich, wenn gemäß der Stellenbeschriftung m Vor- und Nachbereich [15] der Aktion δ Obermengen des Vor- und Nachbereichs von s sind, d.h. in formaler Schreibweise $s \subseteq \delta :\Longleftrightarrow m({}^\bullet s) \subseteq m({}^\bullet \delta) \wedge m(s^\bullet) \subseteq m(\delta^\bullet)$. Zu jeder s gemäß $s \subseteq \delta$ einbettenden Aktion $\delta \in \Delta_{\text{self}}$ werden entsprechende, in $\tilde{P}(\delta, \text{self})$ aufgeführte Rüstpläne in Einzelaktionen $\tilde{\delta}$ aufgebrochen und die $\tilde{\delta}$ anschließend in der agenteneigenen Aufgabenliste A_{self} vermerkt. Ein der Unterscheidung dienender Identifikator i und ein Label 'new' zur Kennzeichnung des Planungszustandes der betreffenden Aktion ergänzen die jeweiligen Einträge in A_{self}.

method distribute(self, $s \in T, p \in P$)
for $\delta \in \Delta_{\text{self}}$ **do**
  if $s \subseteq \delta$ **then**
    for $\tilde{p} \in \tilde{P}(\delta, \text{self})$ **do for** action $\tilde{\delta} \in d(\tilde{p})$ **do**
      $A_{\text{self}} \leftarrow A_{\text{self}} \cup \{(\tilde{\delta}, i(o, a, p, s, \ldots), \text{new})\}$
    endfor endfor
  fi
endfor

3.2 Test auf Ausführbarkeit

In der nächsten Phase werden die vom System entwickelten Pläne auf Ausführbarkeit getestet und, sofern notwendig, um weitere Aktionen ergänzt. Diese Aufgaben übernimmt die Methode *check-plan*. Sie bearbeitet jede Stelle $\alpha \in {}^\bullet\delta \cup \delta^\bullet$ einer in der Aufgabenliste A_{self} des Agenten $\textit{self} \in \mathcal{A}$ vermerkten Aktion $(\delta, i, \text{new}) \in A_{\text{self}}$ abhängig von ihrem Typus. Für diesen unterscheidet *check-plan* vier Fälle:

1) Erfüllt α das Prädikat *starting-place*, handelt es sich um eine Startstelle im Sinne des zugrunde liegenden Produktionsplans. Für Startstellen von Rüstplänen gilt stets *starting-place*(α) = *false*. Die vorgenommene Unterscheidung resultiert aus der Verschiedenartigkeit des Systemverhaltens gegenüber Bauteilen/Rohstoffen und Hilfsmitteln/Werkzeugen. Während letztere im Prinzip immer wieder neu verwendbar sind, werden erstere verarbeitet und stehen somit nach ihrem Gebrauch nicht mehr zu Verfügung. Daraus ergibt sich folgende spezielle Behandlung für α: Durch Aufruf der Methode *getplan* wird versucht, einen Plan zum Antransport eines Objekts $m(\alpha)$ zu entwickeln. Ein Markierungsverfahren stellt sicher, daß keine Mehrfachbelegungen erfolgen. Die Markierungsinformationen für ein einzelnes Objekt $o \in O$ verwaltet dabei genau der Agent a, für den $o \in O_a$ gilt; sie sind im Falle von Objektbewegungen nach Bedarf an andere Agenten weiterzugeben. Existiert kein geeignetes Objekt oder versagt das System hinsichtlich der Konstruktion eines entsprechenden Transportplans, versucht *check-plan* mit Hilfe der Methode *prod-plan* einen Aktionsplan zur Produktion von $m(\alpha)$ zu erstellen. Scheitert auch dies, ist (δ, i, new) nicht ausführbar und wird ersatzlos aus der Aufgabenliste A_{self} entfernt. Gibt es darüber hinaus keine alternative Realisierung der Aktion δ, schließt die Prozedur *remove* den gesamten zu δ gehörenden Plan $p(i)$ aus den weiteren Berechnungen aus.

2) Im Falle von *intermediate-place*(α) = *true* stellt α eine Zwischenstelle des zugrunde liegenden Produktionsplans dar, die zusätzlich $\alpha \in \delta^\bullet$ erfüllt. Aufgabe von *check-plan* ist hier die Entwicklung geeigneter Transportpläne zur Verbindung der Stelle α aus dem Nachbereich von δ mit Stellen aus den Vorbereichen von Aktionen δ', die hinsichtlich des zugehörigen Produktionsplans unmittelbar auf δ folgen. Dazu wird für alle Nachfolgeaktionen $(\delta', i', \text{new}) \in \textit{postprodacts}(\delta)$ von δ mit $\alpha \in {}^\bullet\delta'$ die Methode *moveplan* aufgerufen. Kann *moveplan* für kein δ' einen Transportplan

bereitstellen (es muß mindestens ein δ' existieren wegen *intermediate-place*(α) = *true*), ist der intendierte Plan nach Beendigung von δ nicht mehr fortsetzbar; dann entfernt *check-plan* mit Hilfe der Prozedur *remove* (δ, i, new) aus der Liste A_{self}. Gibt es keine Aktion δ' als Ersatz für δ, wird der gesamte zu δ gehörende Plan $p(i)$ aus den Aufgabenlisten gelöscht.

3) Startstellen von Rüstplänen sind gekennzeichnet durch *pre-place*(α) = *true*. Bei den für sie angeforderten Objekten $m(\alpha)$ handelt es sich stets um Hilfsmittel. Deren Wiederverwendbarkeit ermöglicht den Verzicht auf ein Markierungsverfahren zur Vermeidung von Mehrfachbelegungen, zumal die weiter oben erwähnte Symmetriebedingung für Objekttransporte den Erhalt der Verfügbarkeit auch bei Durchführung beliebiger Objektbewegungen sicherstellt. Derartige Bewegungen können etwa infolge von Aktionen nebenläufiger Pläne auftreten. *check-plan* begnügt sich daher im Gegensatz zu Fall 1 mit der Beauftragung der Methode *preactplan*, die versucht, ein Objekt des benötigten Typs $m(\alpha)$ im System zu finden und einen geeigneten Aktionsplan zum Antransport zu konstruieren. Liefert *preactplan* nicht das gewünschte Ergebnis, verfährt *check-plan* in völliger Übereinstimmung zu Fall 1: Ist auch *prodplan* nicht in der Lage, das Objekt $m(\alpha)$ bereitzustellen, wird die Aktion (δ, i, new) oder sogar der gesamte zugehörige Plan $p(i)$ aus dem Planungsverfahren ausgeschlossen.

4) Der letzte Fall behandelt die Zielstellen von Rüstplänen. Er ist charakterisiert durch *post-place*(α) = *true*. Seine Aufgabe besteht darin, vom Agenten *self* nicht mehr benötigte Hilfsmittel abzutransportieren und so *self* wieder für neue Aufgaben freizumachen. Dieser Teil der Methode *check-plan* entspricht in seinem Aufbau dem von Teil 3; als einzige Änderung ruft *check-plan* hier anstelle der Methode *preactplan* die Methode *postactplan* auf.

Am Ende von *check-plan* werden bereits überprüfte Einträge der Aufgabenliste durch Abänderung des Labels für den Planstatus von 'new' auf 'tested' gekennzeichnet.

method check-plan(self)
for $(\delta, i, \text{new}) \in A_{\text{self}}$ **do**
  for $\alpha \in {}^\cdot\delta \cup \delta^\cdot$ **do**
    if starting-place(α) **then** sendmessage(self,getplan,α, i,self);
      if nogetplan **then** sendmessage(self,prodplan,$m(\alpha)$, self);
        if noprodplan **then** remove(δ, i, new);
          if ∄ Alternativaktion mit existierenden Startobjekten **then**
            remove($p(i)$); **exitloop fi fi fi**
    orif intermediate-place(α) **then**
      for $(\delta', i', \text{new}) \in$ postprodacts(δ) **do**
        if $\alpha \in {}^\cdot\delta'$ **then** sendmessage(self,moveplan,$m(\alpha)$, self $\rightarrow$ agent(i')) **fi**
      endfor;
      if nomoveplan **then** remove((δ, i, new));
        if $\nexists a' \in \mathcal{A}, (\delta', i', \text{new}) \in A_{a'} : \delta' \neq \delta \wedge s(i') = s(i)$ **then** remove($p(i)$) **fi fi**
    orif pre-place(α) **then**
      sendmessage(self,preactplan,$m(\alpha)$, self);
      if nopreactplan **then** sendmessage(self,prodplan,$m(\alpha)$, self);
        if noprodplan **then** remove(δ, i, new);
          if $\nexists a' \in \mathcal{A}, (\delta', i', \text{new}) \in A_{a'} : \delta' \neq \delta \wedge s(i') = s(i)$ **then**
            remove($p(i)$) **exitloop fi fi fi**
    orif post-place(α) **then**
      sendmessage(self,postactplan,$m(\alpha)$, self);
      if nopostactplan **then** remove((δ, i, new));
        if $\nexists a' \in \mathcal{A}, (\delta', i', \text{new}) \in A_{a'} : \delta' \neq \delta \wedge s(i') = s(i)$ **then** remove($p(i)$) **fi fi fi**
  endfor

$A_{\text{self}} \leftarrow A_{\text{self}} \setminus \{(\delta, i, \text{new})\} \cup \{(\delta, i, \text{tested})\}$
endfor

Die Methode *getplan* ist dafür zuständig, Bauteile und Rohstoffe für den Produktionsvorgang bereitzustellen. Wie bereits bei *check-plan* erwähnt, muß das System derartige Verbrauchsobjekte mit Hilfe von Markierungen von einer mehrfachen Verwendung ausschließen. Der Aufruf von *getplan* erfolgt so, daß zunächst getestet wird, ob der anfordernde Agent $a \in \mathcal{A}$ das gewünschte Objekt $m(\alpha)$ zur Verfügung stellen kann. Dazu prüft a in *getplan* alle Elemente o seiner lokalen Objektmenge O_{self} auf Typübereinstimmung mit $m(\alpha)$; gilt diese und ist o noch nicht für eine anderweitige Verwendung markiert, beauftragt a die Methode *moveplan* mit der Konstruktion eines Transportplans für o. Sobald ein solcher vorliegt, markiert *getplan* das Objekt o und beendet seine Arbeit. Hat der $m(\alpha)$ anfordernde Agent a so keinen Erfolg, werden nacheinander alle anderen im System existierenden $a' \in \mathcal{A} \setminus \{a\}$ zu entsprechenden Kalkulationen veranlaßt. Auch hier gilt die Regel, daß bei Erhalt eines geeigneten Ergebnisses *getplan* sofort abbricht. Ist *self* nicht mit a identisch, erübrigt sich die Benachrichtigung, da eine solche bereits vom Agent a vorgenommen wird.

Im Gegensatz zu den anderen Teilen des Algorithmus zur Planerstellung nimmt *getplan* keine vollständige Generierung aller möglichen Objektbelegungen vor, sondern ist mit der Aufstellung einer einzelnen ausführbaren Planvariante zufrieden. Dies stellt ein Zugeständnis an die Einfachheit des verwendeten Markierungsverfahrens dar. Durch entsprechende Änderungen kann *getplan* aber jederzeit verallgemeinert werden.

method getplan(self,$\alpha \in S$,$i \in \mathsf{N}$,$a \in \mathcal{A}$)
for $o \in O_{\text{self}}$ **do**
  if $o = m(\alpha) \wedge$ not marked(o, i, α) **then** sendmessage(self,moveplan,o, self $\rightarrow a$);
    if moveplan **then** mark(o, i, α); **exitloop fi**
  fi
endfor
if self= a **then**
  for $a' \in \mathcal{A} \setminus$ {self} **do**
    sendmessage(a',getplan,o, α, a);
    if *getplan* ***then exitloop fi***
  endfor
fi

3.3 Planauswahl

Zur Bewertung der Pläne, die *check-plan* im Laufe seiner Arbeit entwickelt, und zur Auswahl der zu realisierenden Planvariante aufgrund dieser Bewertung starten die Agenten die Methode *select-plan*. Sie bearbeitet alle Einträge $(\delta, i, \text{tested}) \in A_{\text{self}}$ der Aufgabenlisten A_{self} in dreifacher Hinsicht:

1) Ähnlich der Funktionsweise einer attributierten Grammatik [12] berechnet *select-plan* abhängig von den globalen Bewertungen der Nachbaraktionen und der lokalen Bewertung von δ je nach der in δ gerade vorliegenden Planstruktur die globale Bewertung von δ. Dabei beurteilen lokale Bewertungen einzelne Aktionen, globale dagegen den gesamten Teilplan von den Startstellen bis zur angegebenen Aktion.

2) Aus Planvarianten, die in δ zusammenlaufen, wählt *select-plan* aufgrund der ihnen jeweils zugeordneten Bewertungen die günstigste aus. Alle anderen Varianten werden anschließend gelöscht.

3) Abschließend wird die für δ ermittelte globale Bewertung an alle Nachfolgeaktionen weitergeleitet.

Eine direkte Umsetzung dieser Aufgaben in einen entsprechenden Algorithmus verkompliziert sich durch die Existenz nachbereitender Aktionen. Man erhält für *select-plan* schließlich die folgende Gestalt:

Sind alle Vorgängeraktionen von δ bewertet und sind die Planstatuslabel aller Nachbaraktionen $preacts(\delta) \cup preactions(\delta) \cup postactions(\delta)$ auf 'tested' gesetzt, leitet *select-plan* die Berechnung der globalen Bewertung für δ ein. Fehlen nachbereitende Aktionen, erfolgt diese sofort durch die Funktion $evaluate(\delta, \omega(\delta'_1), \ldots, \omega(\delta'_n))$. Andernfalls nimmt man unter Benützung von $preevaluate(\delta, \omega(\delta'_1), \ldots, \omega(\delta'_n))$ zunächst eine Vorbewertung $\omega(\delta)$ vor. Sie dient der Aktualisierung einiger wichtiger globaler Bewertungsparameter und soll unter anderem eine zeitliche Einordnung der nachbereitenden Aktionen ermöglichen. An diese gibt *select-plan* $\omega(\delta)$ unmittelbar nach Ausführung von *preevaluate* weiter. Liegt bereits eine Vorbewertung von δ vor und ist die Bewertung aller nachbereitenden Aktionsfolgen abgeschlossen, führt *select-plan* unter Benützung der Funktion $evaluate(\delta, \omega(\delta'_1), \ldots, \omega(\delta'_m))$ abhängig von den globalen Bewertungen der Vorgängeraktionen und der nachbereitenden Aktionsfolgen die eigentliche globale Bewertung von δ durch.

Unmittelbar nach erfolgter Bewertung, gekennzeichnet durch $evaluated(\delta) = true$, bereinigt *select-plan* mit Hilfe der Prozedur *prune* die betrachteten Pläne von ungünstigen Varianten. Anschließend gibt *select-plan* die ermittelte Bewertung $\omega(\delta)$ an alle Folgeaktionen weiter, die nicht zu den nachbereitenden gehören. Handelt es sich bei δ dagegen um eine Aktion ohne Nachfolger, muß unterschieden werden zwischen dem Abschluß einer nachbereitenden Aktion ($postend(\delta) = true$) und dem Abschluß durch Produktion des gewünschten Objekts ($postend(\delta) = false$). Während bei $postend(\delta) = true$ *check-plan* die Bewertung $\omega(\delta)$ an den Agenten $preagent(i)$ sendet, der die vorliegende nachbereitende Aktion initiiert hat, ist im zweiten Fall der das zu produzierende Objekt anfordernde Agent $owner(i)$ der Empfänger. *select-plan* beendet seine Arbeit durch Korrektur des Planstatuslabels der Aktion $(\delta, i, \text{tested})$ von 'tested' auf 'value' als Zeichen dafür, daß δ bereits bewertet worden ist.

method select-plan(self)
for $(\delta, i, \text{tested}) \in A_{\text{self}}$ **do**
  if $\forall(\delta'_k, i_k, l'_k) \in$ preactions$(\delta) \cup$ preacts(δ): evaluated$(\delta') \wedge l'_k =$'tested' $\wedge$
    $\forall(\delta'_k, i_k, l'_k) \in$ postactions(δ): $l'_k =$'tested' **then**
      if postactions$(\delta) = \emptyset$
      then $\omega(\delta) \leftarrow$ evaluate$(\delta, \omega(\delta'_1), \ldots, \omega(\delta'_n))$
      else if not preevaluated(δ)
        then $\omega(\delta) \leftarrow$ preevaluate$(\delta, \omega(\delta'_1), \ldots, \omega(\delta'_n))$;
          for $(\delta'_k, i_k, \text{tested}) \in$ postactions(δ) **do**
            sendmessage(agent(i),$(\omega(\delta), i) \rightarrow B_{\text{agent}(i)}$) **endfor**
        else if $\forall(\delta'_k, i_k, \text{tested}) \in$ postends(δ): evaluated(δ') **then**
          $\omega(\delta) \leftarrow$ evaluate$(\delta, \omega(\delta'_1), \ldots, \omega(\delta'_m))$ **fi fi fi fi**
  if evaluated(δ) **then** prune(δ);
    if postacts$(\delta) \neq \emptyset$
    then for $(\delta'_k, i_k, \text{tested}) \in$ postacts(δ) **do**
      sendmessage(agent(i),$(\omega(\delta), i) \rightarrow B_{\text{agent}(i)}$) **endfor**
    else if postend(δ)
      then sendmessage(preagent(i),$(\omega(\delta), i) \rightarrow B_{\text{preagent}(i)}$)
      else sendmessage(owner(i),$(\omega(\delta), i) \rightarrow B_{\text{owner}(i)}$)
      fi
    fi;
    $A_{\text{self}} \leftarrow A_{\text{self}} \setminus \{(\delta, i, \text{tested})\} \cup \{(\delta, i, \text{value}, \omega(\delta))\}$ **fi**
endfor

Die bisherigen Arbeitsschritte dienten dazu, geeignete Pläne zur Produktion des gewünschten Objekts o zu entwerfen, sie hinsichtlich der vorgegebenen Bewertungsfunktion zu beurteilen und variantenfrei zu machen. Es verbleibt die Auswahl eines Planes gemäß den festgelegten Bewertungen. Dies geschieht mit Hilfe der Methode *move*. In ihr bearbeitet der Agent $self \in \mathcal{A}$ alle Einträge $(\omega(\delta), i)$ seiner Bewertungsmenge B_{self}. Ist δ eine Finalaktion und sind alle Finalaktionen der Pläne, die hinsichtlich des zu produzierenden Objekts dem δ zugehörigen Plan entsprechen, bewertet, ruft $self$ die Prozedur *prune* auf und nimmt somit die beabsichtigte Planauswahl vor. Danach ändert die Methode *move* das Label für den Planungsstatus der verbliebenen abschließenden Aktion von 'value' auf 'move', um anzuzeigen, das die betreffenden Aktionen nun den Status der Ausführbarkeit erreicht haben. Generell geschieht dies mit allen Einträgen der Aufgabenliste, sobald für einen Vorgänger oder Nachfolger der gerade betrachteten Aktion der Status 'move' festgestellt wird. Schrittweise arbeitet sich das System so bis zu den Startaktionen durch. Trifft $self$ auf eine Startaktion, d.h. eine Aktion ohne Vorgänger, mit bereits auf 'move' gesetzten Label, übergibt er diese von der Aufgabenliste A_{self} an die Ausführungsliste E_{self}.

Das etwas aufwendige mehrmalige Scannen der Aktionen eines Plans hat den Zweck, die Aktionen so gut wie möglich in der 'richtigen' Reihenfolge an die Ausführungslisten zu übergeben und damit aufwendige Umsortierungen basierend auf den Informationen der Zeitbewertung weitgehend einzusparen.

method move(self)
for $(\omega(\delta), i) \in B_{\text{self}}$ **do**
  if postacts$(\delta) = \emptyset \wedge$ not postend$(\delta) \wedge$
    $\forall(\delta, i, \text{value}, \omega(\delta)) \in$ lastactions$(\omega(\delta))$: evaluated(δ) **then**
      prune(plan(i));
      for $(\delta, i, \text{value}, \omega(\delta)) \in$ lastactions(ω) **do**
        $A_{\text{agent}(i)} \leftarrow A_{\text{agent}(i)} \setminus \{(\delta, i, \text{value}, \omega(\delta))\} \cup \{(\delta, i, \text{move}, \omega(\delta))\}$ **endfor fi**
endfor
for $(\delta, i, \text{value}, \omega(\delta)) \in A_{\text{self}}$ **do**
  if $\exists(\delta', i', \text{move}, \omega(\delta')) \in$ postactions$(\delta) \cup$ preactions$(\delta) \cup$ postacts$(\delta) \cup$ preacts(δ) then
    $A_{\text{self}} \leftarrow A_{\text{self}} \setminus \{(\delta, i, \text{value}, \omega(\delta))\} \cup \{(\delta, i, \text{move}, \omega(\delta))\}$ **fi**
endfor
for $(\delta, i, \text{move}, \omega(\delta)) \in A_{\text{self}}$ **do**
  if $\nexists(\delta', i', \ldots, \omega(\delta')) \in$ preactions$(\delta) \cup$ preacts(δ) **then**
    transfer$(\delta, i, \text{move}, \omega(\delta)) \in A_{\text{self}}$ to $(\delta, i, \omega(\delta)) \in E_{\text{self}}$ **fi**
endfor

3.4 Dynamische Korrektur

Hat ein Plan in Gestalt einzelner Aktionen die Methode *move* erfolgreich durchlaufen, ändert sich für ihn die Betrachtungsweise durch das System. Er gehört nicht mehr zu einer Menge ausführbarer Pläne, aus denen der am besten geeignete ausgewählt werden soll, sondern ist die aktuell zu befolgende Arbeitsvorschrift zur Herstellung des gewünschten Objekts. Etwaige Modifikationen dieses Plans sollten mit der zwischenzeitlich eventuell bereits geleisteten Arbeit verträglich sein, um diese nicht wertlos zu machen; insofern sind Änderungen im großen Maßstab wenig sinnvoll. Kleinere Korrekturen dagegen haben in zweierlei Hinsicht durchaus ihre Berechtigung:

- Abweichungen der Planungsgrundlage vom realen Systemstatus können zuvor verworfene Varianten wieder aktuell machen.

- Aufwendige Verfahren zur Planoptimierung, dessen Kosten/Nutzen-Verhältnis bisher zu schlecht war, können durch Beschränkung auf einen einzigen konkreten Plan wieder aktuell werden.

Der Durchführung derartiger während der Planrealisierung stattfindender dynamischen Korrekturen dient die Methode *dyncorr*. Welche verschiedenen Verfahren geeignet sind, ist noch nicht ganz geklärt; letztendlich müssen hier wohl praktische Experimente entscheiden. Wir begnügen uns daher mit einer überblicksartigen Aufzählung verschiedener Techniken.

- Durch Permutationen der Reihenfolge von Aktionen, die in den Listen E_a auf ihre Ausführung warten, sind eventuell große Teile der vorzunehmenden Auf- und Abrüstvorgänge einsparbar. Einen ähnlichen Effekt kann der Austausch von Aktionen zwischen den Agenten haben, doch ist hier im Bedarfsfall der Änderungsaufwand umfangreicher.

- Ein weites Betätigungsfeld für dynamische Korrekturen bieten Transporte oftmals eingesetzter Hilfsmittel. Hier lassen sich hinsichtlich des Zeitbedarfs durch Verschmelzungen vor- und nachbereitender Aktionsfolgen unter Umständen drastische Einsparungen erzielen. Auch eine genaue zeitliche Abstimmung der Aktionen aufeinander kann hilfreich sein; ansonsten notwendige Zwischenlagerungen von Objekten entfallen dann.

- Jede Veränderung der Agentenmenge sollte Anlaß sein, die Verteilung der Aktionen auf die einzelnen Agenten einer erneuten Prüfung zu unterwerfen.

- Möglicherweise empfiehlt es sich, mit 'gut' bewertete Varianten als Alternativen im Plan zu belassen; unvorhergesehene Abweichungen des realen Systemstatus von der Planungsgrundlage könnten dann leichter kompensiert werden.

Unmittelbar vor Ausführung befindliche Aktionen δ werden von *dyncorr* durch Übertragung von der Liste E_a in die Liste D_a jedem weiteren Modifikationszugriff entzogen. Zur Vermeidung von Inkonsistenzen wird mit allen Aktionen, die zu δ in enger Beziehung stehen und zeitlich vor δ auszuführen sind, entsprechend verfahren.

method dyncorr(self)
for $(\delta, i, \omega(\delta)) \in E_{\text{self}}$ **do**
  if exectime(δ) near realtime
  then transfer$(\delta, i) \in E_{\text{self}}$ to $(\delta, i) \in D_{\text{self}}$;
    for $a' \in \mathcal{A}$ **do**
      sendmessage(a',Übergebe alle δ-bezogenen Aktionen nach D)
    endfor
  else
    Expandiere Ersatzteilpläne und bewerte sie;
    Ersetze eventuell alten Teilplan gegen neuen;
    Neuordnung der aktuellen Variante in zeitlicher Hinsicht;
    Überprüfung auf Verschmelzbarkeit von vor/nachbereitenden Aktionen;
    Überprüfung auf Vereinfachbarkeit lokaler Aktionspläne
  fi
endfor

4 Zusammenfassung und Ausblick

Es wurde ein verteilter Planungsalgorithmus für den Einsatz in Fertigungsumgebungen entwickelt, der den Aktionsfluß zur Steuerung der Agenten nutzt. Der vorgestellte Algorithmus ist in der Lage, einige Anomalien des Contract-Net Modells

zu vermeiden. Die durchgeführte Untersuchung liefert mit dem Verfahren, globale Informationen über Aktionsbewertungen zu transportieren, abhängig von der lokalen Planstruktur zu verarbeiten und so eine Auswahl hinsichtlich der Aufgabenverteilung zu treffen, einen auch außerhalb des Computer Integrated Manufacturing bei anderen Planungsproblemen vielseitig anwendbaren Ansatz. Es erscheint der Schluß möglich, daß sich die aktionsflußbasierte Planung insbesondere für komplexe Situationen eignet. Der Einsatz von Heuristiken eröffnet ein zusätzliches Entwicklungspotential.

Literatur

[1] J.Ayel: A Conceptual Supervision Model in Computer Integrated Manufacturing, Proceedings of the 8th European Conference of Artificial Intelligence, München 1988, S.427-432.

[2] J.Ayel: Decision Coordination in Production Management, in Preproceedings of the 4th European Workshop on 'Modelin Autonomous Agents in a Multi-Agent World' 1992.

[3] R.Davis, R.G.Smith: Negotiation as a Metaphor for Distributed Problem Solving, Artificial Intelligence, Vol.20(1983), S.63-109.

[4] B.L.Dietrich: A Taxonomy of Discrete Mabufacturing Systems, Operations Research, Vol.39(1991), S.886-902.

[5] G.Doumeingts: Methodology to Design CIM and Control of Manufacturing Units, in U.Rembold, R.Dillmann: Methods and Tools for CIM, Lecture Notes in Computer Science, Springer 1984, S.138-194.

[6] J.Draeger: Grundlegende Gedanken zur Formalisierung und Durchführung verteilten Planens, 1992, unveröffentlicht.

[7] J.Draeger: Ein Algorithmus zur verteilten Planung in Fertigungsumgebungen, 1993, unveröffentlicht.

[8] L.Gasser, C.Braganza, N.Herman: MACE: A Flexible Testbed for Distributed AI Research, in M.N.Huhns: Distributed Artificial Intelligence Vol.1, Pitman 1987, S.119-152.

[9] S.Hahndel, P.Levi: Kooperative Systeme in der Fertigung, in diesem Band.

[10] J.E.Hopcroft, J.D.Ullman: Introduction to Automata Theory, Languages and Computation, Addison-Wesley 1979.

[11] T.Kupec: Wissensbasiertes Leitsystem zur Steurung flexibler Fertigungsanlagen, Dissertation TU München, Springer 1991.

[12] U.Mahn: Atributierte Grammatiken und Attributierungsalgorithmen, Springer 1987.

[13] H.v.D. Parunak: Manufacturing Experience with the Contract Net, in M.N.Huhns: Distributed Artificial Intelligence Vol.1, Pitman 1987, S.285-310.

[14] H.v.D. Parunak: Distributed AI and Manufacturing Control: Some Issues and Insights, in Y.Demazeau, J.P.Müller: Decentraliced A.I. (1989), North Holland 1990, S.81-101.

[15] W.Reisig: Petri Nets, Springer 1985.

[16] E.Sandewall: The Pipelining Transformation on Plans for Manufacturing Cells with Robots, Proceedings of the 10th International Joint Conference of Artificial Intelligence, Milano 1987, S.1055-1062.

[17] E.Werner: The Design of Multi-Agent Systems, in E.Werner, Y.Demazeau: Decentraliced A.I.-3 (1991), North Holland 1992, S.3-28.

Ein hierarchischer Blackboard-Ansatz für Verteilte PPS-Systeme

Mathias Philipp[1] und Michael Weiß[2]

[1]Institut für Wirtschaftsinformatik
Universität Frankfurt
philipp@iwi.uni-franfurt.dbp.de

[2]Lehrstuhl für Praktische Informatik I
Universität Mannheim
weiss@pinfo100.informatik.uni-mannheim.de

Abstract

Mit unserem Beitrag wollen wir unsere Forschungsaktivitäten im Bereich Verbindung von Verteilter Künstlicher Intelligenz (VKI) mit Verteilten PPS-Systemen (VPPS) vorstellen, die im Rahmen der Entwicklung des Hierarchischen Blackboard-Systems HBBS durchgeführt werden. Zunächst wollen wir unseren hierarchischen Blackboard-Ansatz gegen andere Forschungsansätze abgrenzen bzw. einordnen. Nach einer Motivation unseres Ansatzes für die Verbindung von VKI mit VPPS wird die System- und Kontrollstruktur von HBBS vorgestellt. Es folgt die Anwendung von HBBS auf die Verbindung von VPPS und VKI sowie die Vorstellung eines Prototyps zur kurzfristigen Umdisposition im Störungsfall. Es zeigt sich, daß HBBS für die Anwendung im Bereich der VPPS über ausreichend mächtige Mechanismen verfügt. Insbesondere wird gezeigt, wie unter HBBS verschiedene Interaktionsstrategien (Kooperation und Konkurrenz) innerhalb derselben Multiagentenstruktur umgesetzt werden können.

1. Einordnung in den Stand der Forschung

VPPS-Forschungsansätze

In der Literatur gibt es grob drei Sichtweisen, wie an die Problemstellung der Verteilten PPS herangetreten wird (Abb. 1):

1. Beim technikorientierten Ansatz wird ausgehend von technischen Gegebenheiten versucht, die PPS-Verteilung so vorzunehmen, daß hard- und softwaretechnische Probleme wie beispielsweise Lastverteilung in einem Rechnernetz minimiert werden. In [1] wird beispielsweise auf die Problematik von verteilten Datenbeständen in der PPS eingegangen.

2. Beim anwendungsorientierten Ansatz wird die Verteilung rein aus Sicht des Anwenders betrieben, so daß betriebswirtschaftliche und organisatorische Aspekte im Vordergrund stehen. Hierbei kann sich eine Verteilung an bereits bestehenden betrieblichen Strukturen orientieren [2] oder sich von einer bereits existierenden organisatorischen Aufteilung ganz lösen [3]. Kemmner beispielsweise geht von 22

allgemeingültigen PPS-Teilfunktionen aus, die anhand von Abspaltungs- un Gruppierungskriterien zu Verteileinheiten (Subsystemen) zusammengefaßt werden.

3. PPS-Funktionen können auch in Form von isolierten, oft historisch gewachsene Insellösungen vorliegen [4]. Der Integrationsorientierte Ansatz verfolgt hier de Entwurf eines Verteilten PPS-Systems, so daß die oft sehr heterogenen Hard- un Softwareumgebungen der einzelnen Inseln integriert werden.

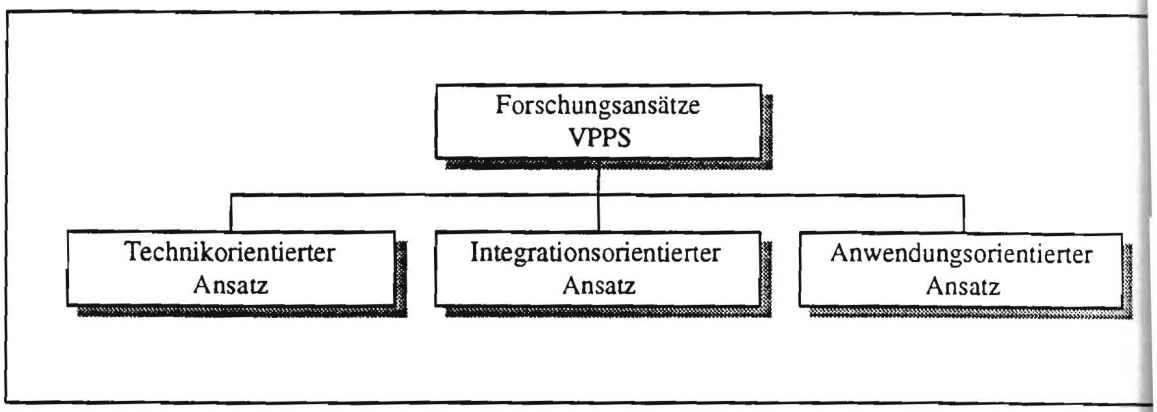


Abb. 1: Forschungsansätze der VPPS

Verbindung zwischen PPS und KI

Wie Abb. 2 zeigt, gibt es verschiedene Möglichkeiten, wissensbasierte Systeme in die Produktionsplanung und -steuerung einzubeziehen. Eine Zuordnung vorhandener Systeme sieht wie folgt aus: Punkt (1) - z.B. ISIS/OPIS, YAMS; Punkt (2) - z.B. XPS zur Unterst tzung der Erstinstallation des PPS-Systems RM von SAP; Punkt (3) - z.B SOJA/SONIA; Punkt (4) - z.B. DEPRODEX.

Unser Ansatz ist ebenfalls in Punkt (4) einzuordnen. Der Unterschied zu anderen Systemen besteht u.a. in dem hierarchischen Multi-Blackboard-Ansatz und damit in der Wissens- und Kommunikationsstruktur von HBBS.

2. Motivation

Bei der Entwicklung von ISIS/OPIS [6] und auch von SOJA/SONIA [7] hat sich gezeigt, daß Blackboard-Ansätze für die Verbindung von VKI mit VPPS besonders geeignet sind. Neben den allgemeinen Nachteilen wie Ausfallsicherheit haben die Erfahrungen (z.B. mit Hearsay-II [8]) jedoch auch gezeigt, das ein einzelnes (zentrales) Blackboard schnell zum Engpaß werden kann. Dieser Engpaß kann sowohl bezüglich des Kommunikationsaufkommens als auch der Strukturierbarkeit der (umfassenden) Wissensquellen bestehen.

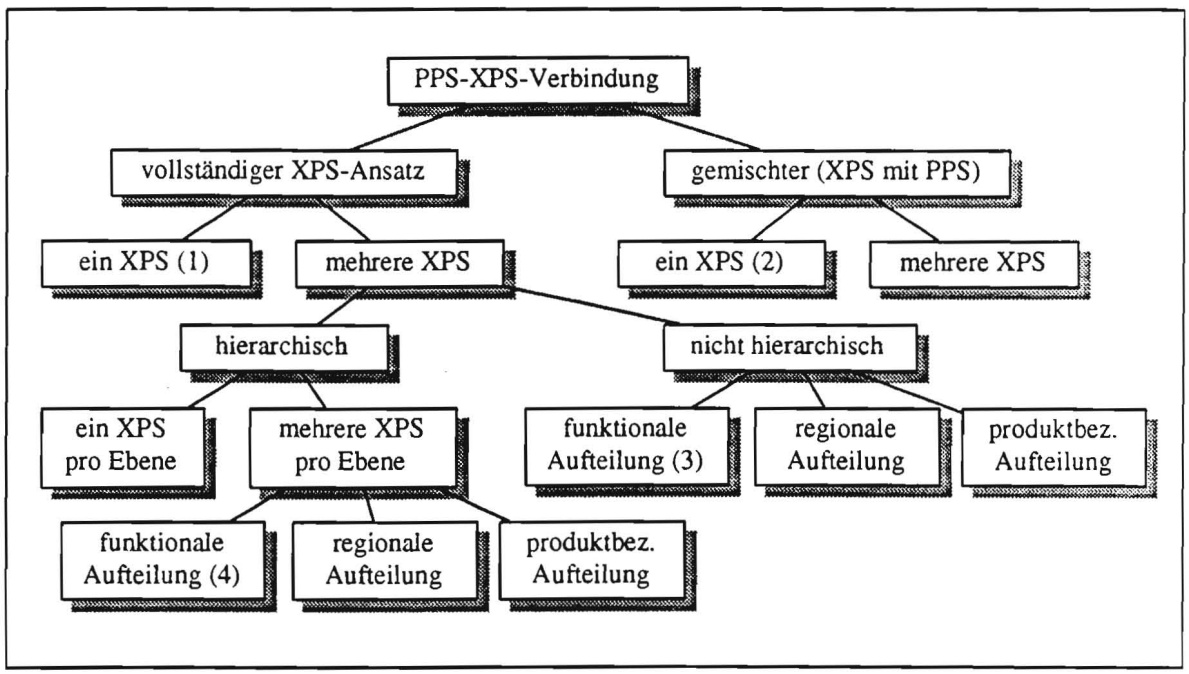


Abb. 2: Systematisierung wissensbasierter Ansätze in der PPS in Anlehnung an [5]

Dies motiviert zunächst eine Verteilung der Aufgaben des zentralen Blackboards auf mehrere verteilte Blackboards. Unseres Erachtens sollte für die Verbindung von VKI mit VPPS eine solche verteilte Blackboard-Struktur aus folgenden Gründen hierarchischer Art sein:

Intelligente Informationaggregation Betrachtet man das häufig benutzte Modell der Unternehmenspyramide, ist vertikale als auch horizontale Information zwar ein Produktionsfaktor, aber im Gegensatz zu früher kein knappes Gut mehr. Knapp dagegen ist die Ressource "Entscheider" bzw. "Experte". Daraus folgt, daß gerade die selektive Weiterleitung, Aggregation und Aufbereitung von entscheidungsrelevanten Informationen durch eine wissenbasierte Kommunikationsstruktur unterstützt werden muß.

Strukturierung der Prinzipien Kooperation und Konkurrenz Mertens [9] unterscheidet zwischen PPS-Systemen, deren einzelne Problemlöser in Kooperation oder in Konkurrenz zueinander stehen. In den meisten Prototypen wird entweder nur das Prinzip der Konkurrenz oder das der Kooperation verfolgt. Dabei ist es vergleichsweise schwierig, bei einer zentralen Blackboard-Architektur beide Prinzipien klar strukturiert zu implementieren. In einem hierarchischen Multi-Blackboard-System dagegen können die beiden Prinzipien sehr einfach nebeneinander realisiert werden. Ebenfalls ist eine klare hierarchisch-strukturelle Trennung der beiden Prinzipien möglich.

Wissensintegration über eine intelligente, hierarchische Kommunikationsstruktur Es muß eine horizontale als auch vertikale Integration von Daten, Funktionen und Wissen der verschiedenen betrieblichen Anwendungen mit deren Unterstützung durch wissens-

basierte Elemente angestrebt werden. Insbesondere die Wissensintegration erfordert abe eine Kommunikationsstruktur, die nicht wie bisher "tot", sondern "intelligent" ist, wi sie mit HBBS errichtet werden kann.

Ganzheitlicher Ansatz innerhalb der verteilten PPS Die drei VPPS-Forschungs ansätze werden in der aktuellen Forschung tendenziell isoliert bearbeitet. Die Sichtweisen ergänzen sich jedoch und sollten im Sinne einer ganzheitlichen Sichtweise de Problemstellung nicht isoliert betrachtet werden. Mit den Werkzeugen von HBBS könnte diese drei Sichtweisen zusammengeführt werden.

3. Das Hierarchische Blackboard-System HBBS

Das Hierarchische Blackboard-System HBBS [10,11] stellt die für die Interaktion, d.h. Kommunikation und Interaktion, von intelligenten Agenten benötigte Basisfunktionalität bereit und reduziert dadurch den Aufwand, der bei der Entwicklung einzelner Agenten hierfür entstehen würde, erheblich. Es wurde als Infrastruktur für die Konstruktion von Multiagentensystemen entworfen.

Die Bedeutung einer solchen Infrastruktur ist zum einen darin zu sehen, daß sie eine von der Komplexität und den Details der Implementierung der Agenteninteraktion getrennte Entwicklung der Agenten erm"oglicht und zum anderen darin, daß sie eine inkrementelle Entwicklung und Integration der lokalen Perspektiven von Agenten zuläßt. Die Infrastruktur wird als verteiltes Blackboard-System realisiert, dessen Struktur sich am Aufbau realer Organisationen orientiert, die i.d.R. einen hierarchischen Aufbau haben (Abb. 3). Hieraus leitet sich auch der Name des Hierarchischen Blackboard-Systems ab.

Die Elemente eines HBBS sind Metaagenten, die im Gegensatz zu den durch HBBS verknüpften Agenten kein Anwendungswissen, sondern allein Koordinationswissen enthalten. Die Verbindungen zwischen den Agenten werden durch "Botschafter" in den Metaagenten etabliert. Diese Botschafter vertreten die Interessen der Agenten an den Ergebnissen, die von anderen Agenten auf den Blackboards der Metaagenten veröffentlicht werden. Agenten, die auf ähnlichen Teilbereichen des gemeinsamen Problems arbeiten, können dabei zu Teams verbunden werden, die nach außen in derselben Weise wie "normale" Agenten mit den anderen Agenten über Botschafter interagieren.

Botschafter werden als Wissensquellen der Metaagenten-Blackboards realisieret und bestehen, wie allgemein üblich [8], aus Trigger, Vor- und Ausschlußbedingung und Aktionsteil. Sie werden durch Ziele auf dem Blackboard des Metaagenten aktivier ("getriggert") und können bei der Bearbeitung des Ziels auf alle Blackboard-Einträge zugreifen. Die Abarbeitung eines Botschafters führt zur Generierung eines Ziels in dem durch ihn repräsentierten Agenten. Es können aber auch lokale Ziele erzeugt werden, mit deren Hilfe ein komplexes Ziel in Teilziele zerlegt und auf verschiedene Botschafter verteilt werden kann. Dabei nehmen die Botschafter auch eine Umsetzung der gelesenen Informationen in die interne Sprache der Agenten vor.

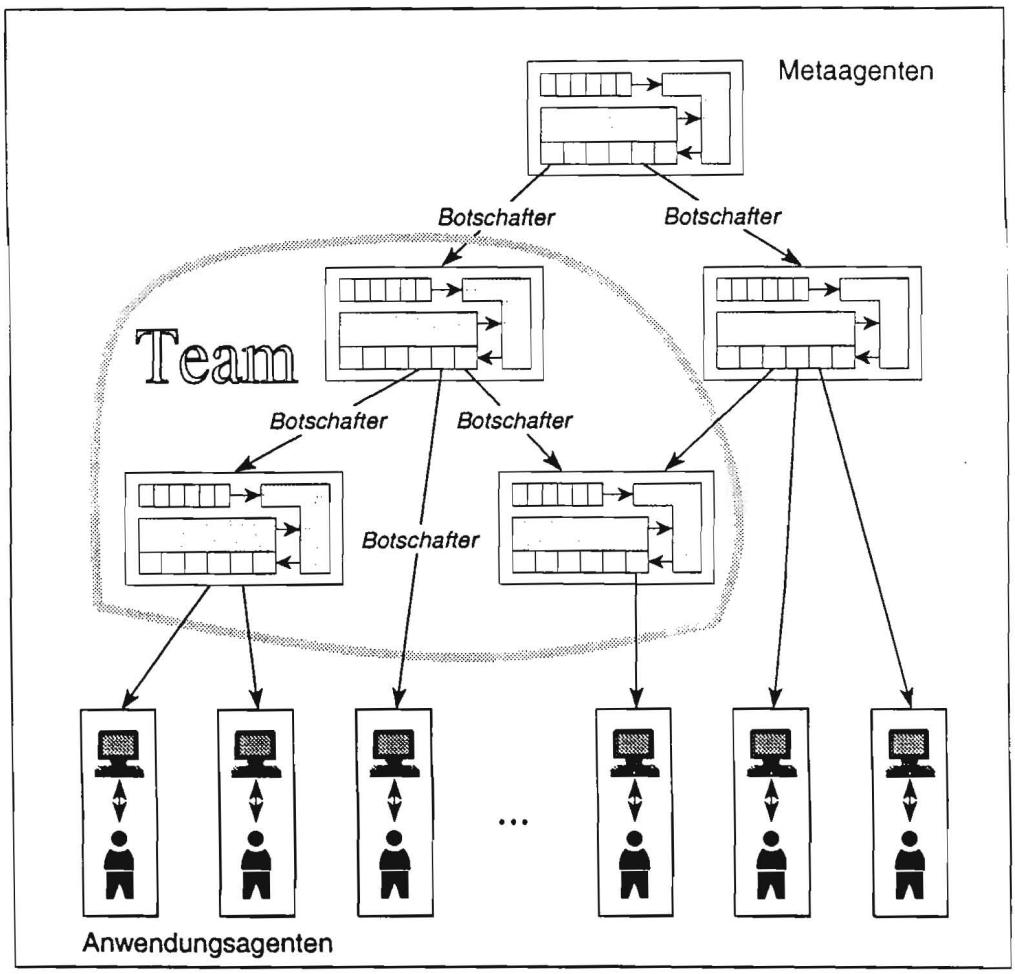


Abb. 3: Hierarchisches Blackboard-Modell

Das Hierarchische Blackboard-System wurde bereits auf den verteilten Entwurf, d.h. die Integration menschlicher und maschineller Agenten (Entwerfer und Werkzeuge) im Entwurf angewandt [11]. Es genügt außerdem den Kriterien einer adaptiven Architektur und erlaubt die Untersuchung von Strategien zur dynamischen Reorganisation von Blackboard-Hierarchien.

4. Anwendung von HBBS auf die Verbindung von VPPS mit VKI

Es werden hochautomatisierte Maschinen und Maschinengruppen eingesetzt, die universell für verschiedene Aufträge verwendbar sind. Ein Fertigungsauftrag ist die Planungseinheit der Produktionsplanung. Ein Maschinenauftrag ist die Planungseinheit auf Maschinenebene und umfaßt alle Arbeitsgänge, die eine Maschine hintereinander ausführen kann. Maschinen sind zu Maschinengruppen zusammengefaßt, die alle denselben Arbeitsplan zur Herstellung eines Maschinenauftrags verwenden. Die

Produktionsplanung dient der Grobterminierung, eine Feinterminierung erfolgt in d Produktionssteuerung.

Die Grenzen des Entwurfs sind darin zu sehen, daß kein neuer Algorithmus zur Lösun des Problems der Produktionsplanung und -steuerung entworfen wird. Vielmehr soll ei Ansatz entworfen werden, der besonders die Kooperation und die Kommunikation vo verteilten PPS-Funktionen mit verteilten Expertensystemen unterstützt. Der Entwu findet auf der Ebene von Ereignissen statt, die entsprechende Botschafter in d Metaagenten des Hierarchischen Blackboard-Systems ansprechen ('triggern').

4.1 Produktionssteuerung

Auf unterster Ebene befinden sich die *Maschinen.* Zu jeder Maschine existiert ei *Maschinenagent* (MaA), der ber die aktuelle Situation (Auslastung, Wartung, ...) seine Maschine informiert ist. Er erhält von dem übergeordneten Maschinengruppen-Black board Aufträge. Falls irgendwelche Störungen auftreten, gibt er den Auftrag mittels eine Fehlermeldung zurück auf das Blackboard, so daß eine andere Maschine der Maschinen gruppe den Auftrag übernehmen kann. Die Fehlermeldung löst eine entsprechend Aktion in Form einer Meldung an übergeordnete Ebenen aus.

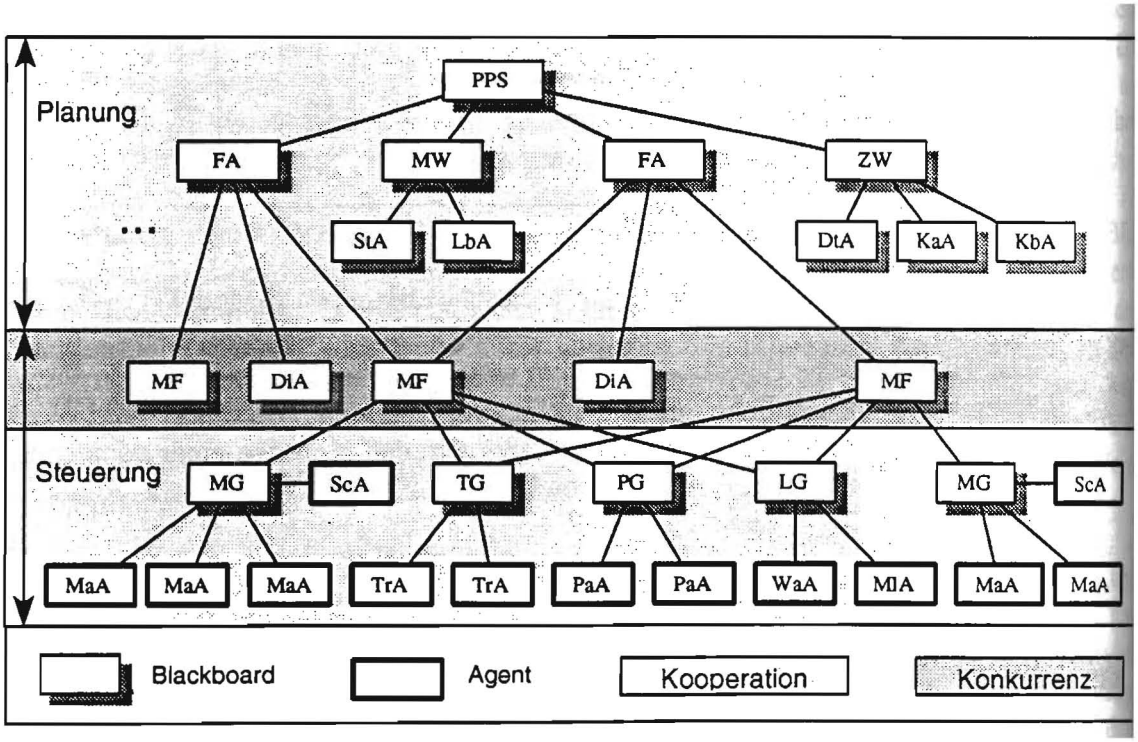


Abb. 3 Verteilte PPS auf Basis von HBBS

Das *Maschinengruppen-Blackboard* (MG) dient dazu, Aufträge für die Maschinengrupp entgegenzunehmen. Es koordiniert die kooperative Zusammenarbeit der einzelne

Maschinenagenten. Dieses Blackboard erhält Maschinenaufträge von einem übergeordneten Maschinenauftrags-Blackboard (MF) oder im Störungsfall auch von einem der Maschinenagenten.

Der *Schedulingagent* (ScA) bernimmt die Reihenfolgeplanung der Fertigungsaufträge einer Maschinengruppe. Das *Maschinenauftrags-Blackboard* (MF) enthält Maschinenaufträge, für deren Bearbeitung sind Lager-, Transport- und Personalkapazitäten nötig sind. Diese werden von einem Lagergruppen-Blackboard (LG), Transportgruppen-Blackboard (TG) und Personalgruppen-Blackboard (PG) zur Verfügung gestellt. Um diese Kapazitäten zu erhalten, formuliert das Maschinenauftrags-Blackboard entsprechende Aufträge. Diese triggern die jeweiligen Botschafter des entsprechenden Blackboards. Die verschiedenen Maschinenauftrags-Blackboards stehen dabei untereinander in Konkurrenz um diese Ressourcen.

Kann eine bestimmte Maschinengruppe einen Maschinenauftrag nicht erfüllen, erfolgt eine Meldung auf dem *Fertigungsauftrags-Blackboard* (FA). Mit Hilfe eines Dispositionsagenten (DiA) werden dann eine oder mehrere Maschinengruppen gesucht, die den Auftrag übernehmen können. Das Fertigungsauftrags-Blackboard ist nach dem Prinzip der Kooperation organisiert, damit sich die Maschinengruppen gegenseitig unterstützen können.

Wird ein Maschinenauftrag aufgrund einer Störungssituation zurückgegeben, aktiviert dies den Botschafter des *Dispositionsagenten* (DiA), der den zurückgegebenen Maschinenauftrag umdisponiert. Dazu muß er eine oder mehrere Maschinengruppen finden, die die Arbeitsgänge des Maschinenauftrags erledigen können.

4.2 Produktionsplanung

Als erstes erscheint der Primärbedarf eines Erzeugnisses auf dem *PPS-Blackboard* (PPS). Dieses Blackboard koordiniert die Materialwirtschaft und die Zeitwirtschaft nach dem Prinzip der *Kooperation*. Ist die Mengenplanung als auch die Kapazitätsplanung abgeschlossen, wird der Botschafter des Fertigungsauftrags-Blackboard aktiviert. Die entsprechenden Fertigungsaufträge erscheinen auf den *Fertigungsauftrags-Blackboard* (FA).

Der *Dispositionsagent* (DiA) sucht freie Maschinengruppen und zerlegt die Fertigungsaufträge anhand der entsprechenden Arbeitspläne in Maschinenaufträge, die dem Maschinenauftrags-Blackboard (MF) übergeben werden. Die hierfür notwendigen Informationen, wie beispielsweise freie Kapazitäten der Maschinengruppen, erhält der Dispositionsagent aus den Rück- und Kontrollmeldungen der Maschinenauftrags-Blackboards.

4.3 Umsetzung der Prinzipien Konkurrenz und Kooperation

Die Zusammenarbeit der beteiligten Systemkomponenten kann dem Prinzip Kooperation oder dem Prinzip der Konkurrenz folgen.

Prinzip der Konkurrenz

Konkurrenz kann prinzipiell um Aufgaben (Aufträge) oder um Ressourcen bestehen. gleiche Aufgaben beziehungsweise Ressourcen in einer hierarchischen Unternehme gliederung auf derselben Hierarchiestufe bestehen, kann somit das Prinzip der K kurrenz nur zwischen Systemelementen herrschen, die sich auf gleicher horizonta Ebene befinden.

Zur Umsetzung des Konkurrenzprinzips gibt es somit zwei Möglichkeiten:

1. Ein Agent hat die Möglichkeit zur Ablehnung von Aufträgen. Erhält beispielswe das Transportgruppen-Blackboard zwei konkurrierende Aufträge von verschieden Maschinenauftrags-Blackboards, beurteilt er deren Attraktivität und lehnt den Auft mit der für ihn geringeren Attraktivität ab.

2. Der Botschafter erhält von seinem Agenten einen Schwellwert, ab welcher Attra tivität er noch bereit ist, neue Aufträge anzunehmen. Dieser Schwellwert geht in Ausschlußbedingung des Bedingungsteils des Botschafters ein.

Es hat sich gezeigt, daa die zweite Variante zur Umsetzung des Prinzips der Konkurre besser geeignet ist.

Prinzip der Kooperation

Folgt man dem Prinzip der Kooperation, so sind zur Erledigung einer bestimmt Aufgabe der Ebene *n* verschiedene Systemelemente der darunterliegenden Ebene *n* notwendig. Wegen der lokalen Sichtweise der Elemente der Ebene *n-1* kann ei Kooperation der Elemente nur über ein Element der Ebene *n* erfolgen.

Das Prinzip der Kooperation kann dadurch realisiert werden, daß die Prioritätensteueru der Agenda des Blackboards der Ebene *n* "gerecht" ist. Gerecht bedeutet hier, daß si die Priorität eines Warteschlangeneintrags an dem Gruppenziel der an diesem Blackboa angeschlossenen Agenten der Ebene *n-1* orientiert.

5 Prototyp

5.1 Aufbau und Ablauf einer Störungsbehandlung

An dieser Stelle wird gezeigt, wie das System auf Störungen reagiert und wie diese so lokal wie möglich behandelt werden (Abb. 4).

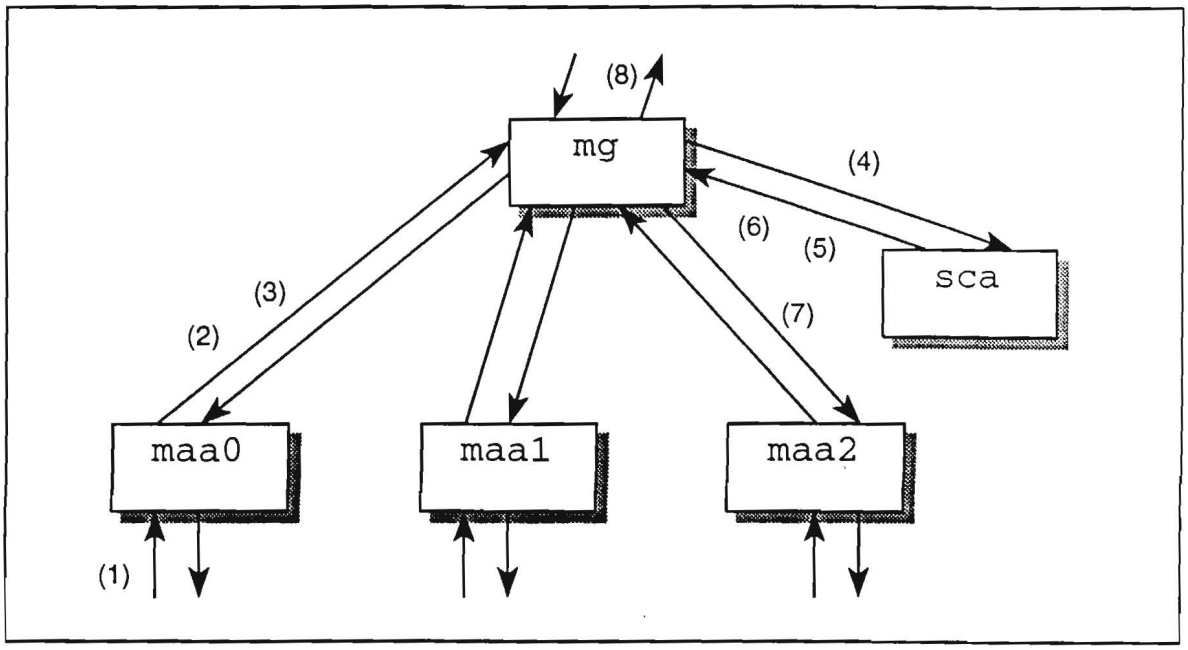


Abb 4. Aufbau des Prototyps

Angenommen, Maschine 0 der betrachteten Maschinengruppe fällt aus, hat aber einen übernommenen Maschinenauftrag noch nicht begonnen. Der Maschinenagent (MaA) informiert sein Maschinengruppen-Blackboard über den Störungsfall, indem er den Auftrag zurückgibt. (Die folgenden Nummern beziehen sich auf Abb. 4 und sind nicht im Programmtext enthalten).

```
(1) defekt --> (2) inform MG to Auftrag ?anr
    defekt --> (3) inform MG to nicht bereit Maschine 0
```

Es kann zunächst probiert werden, den Auftrag von einer anderen Maschine der Gruppe übernehmen zu lassen. Dazu muß der Schedulingagent eine neue Maschine bestimmen. Aufgrund der Fehlermeldung hat er die notwendigen Informationen, um die neue Situation entsprechend zu berücksichtigen.

```
(4) nicht bereit Maschine 0 --> inform ScA
```

Kann eine der Maschinen den Auftrag übernehmen, ist die Störung bereits auf dieser

Ebene erfolgreich behandelt.

```
(5) inform MG to Maschine 2 erhält Auftrag ?anr
(6) inform MAA2 to Auftrag ?anr
```

Kann der Schedulingagent keine neue Maschine bestimmen, gibt er eine entsprech
Meldung zurück.

```
(7) inform MG to Maschine erfolglos erhält Auftrag ?anr
```

Anschließend erfolgt der Versuch, die Störung auf höherer Ebene zu bewältigen. I
wird der Maschinenauftrag ganz aus der Maschinengruppe herausgenommen, und
Maschinenauftrags-Blackboard wie folgt benachrichtigt.

```
(8) inform MF to Auftragsstorno ?anr
```

Das Maschinenauftrags-Blackboard storniert die entsprechenden Transport-, Perso
und Lageraufträge bei den entsprechenden Gruppen-Blackboards, über die auch
entsprechenden Agenten informiert werden.

Anschließend wird versucht, eine andere Maschinengruppe den Maschinenauf
übernehmen zu lassen. Dazu muß der Auftrag dem Fertigungsauftrags-Blackboard (F
übergeben werden.

5.2 Beispielhafter Aufbau eines Maschinenagenten

Die Auftragsbearbeitung findet auf den an die Maschinenagenten angeschlossenen M
schinen statt. Ein Maschinenagent (siehe Abb. 5) dient primär dazu, die Maschine
Aufträgen zu versorgen und Statusmeldungen der Maschine zu behandeln.

Für die Versorgung der Maschine mit Aufträgen ist die Wissensquelle `auftr`
zuständig. Ihr Aktionsteil informiert das Maschinengruppen-Blackboard, daß der Aufti
angenommen wurde, indem es die Nachricht `nicht bereit Maschine` verschickt. U
die Auftragsbearbeitung auf der Ebene des Maschinenagenten simulieren zu k nnen, wi
im Aktionsteil zusätzlich das lokale Teilproblem `bearbeite` erzeugt, welches d
Wissensquelle `auftragsbearbeitung` aktiviert.

Die Wissensquelle erzeugt ebenfalls eine Bereitmeldung an das Maschinengruppe
Blackboard. Desweiteren wird hier das Blackboard-Objekt

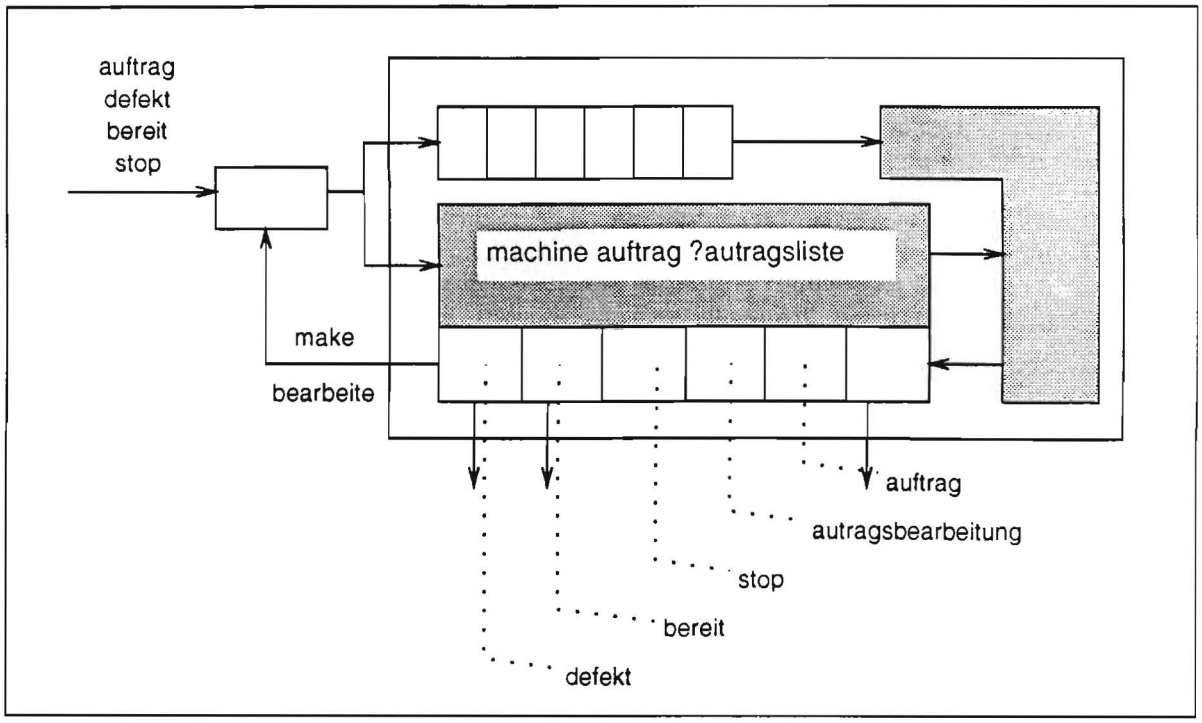


Abb.5 Maschinenagent

```
maschine auftraege ?auftragsliste
```

erzeugt. Es wird zu Dokumentationszwecken und für die Umdisposition im Störungsfall benutzt. Das Objekt enthält alle bearbeiteten Aufträge seit Start des Maschinenagenten sowie den gerade in Bearbeitung befindlichen Auftrag.

Für die Behandlung der Statusmeldungen sind drei Wissensquellen zuständig:

1. Die Wissensquelle `status-stop` dient dazu, ein Ausschalten der Maschine zu verarbeiten.
2. Die Wissensquelle `status-bereit` hat zweierlei Bedeutungen. Erstens wird sie beim Start der Maschine benutzt, zweitens nach Bearbeitung eines Auftrags, so daß die Maschine weitere Aufträge erhalten kann.
3. Die Wissensquelle `status-defekt` reagiert im Störungsfall der Maschine. Der gerade in Bearbeitung befindliche Auftrag muß nun umdisponiert werden.

5. Ergebnisse

Unsere bisherigen Forschungen haben folgendes gezeigt:

- Das hierarchische Blackboard-System HBBS eignet sich gut für die Verbindung vo VKI und VPPS und es können damit in eleganter Art und Weise einige Proble bisheriger Systeme gelöst oder vermieden werden.

- Der neue Ansatz zeichnet sich vor allem durch seinen integrativen Charakter (Int gration verteilter wissenbasierter Systeme und historisch gewachsener Insellösunge und seine Flexibilität aus.

- Neu gegenüber früheren Ansätzen ist ebenfalls, daa mit Hilfe von HBBS die beide Prinzipien Kooperation und Konkurrenz in einem System sauber entworfen un implementiert werden k nnen.

- Wie man an Abb. 4 sehen kann, wird hier in natürlicher Weise eine anwenderorie tierte Organisationsstruktur eines Unternehmens abgebildet. Es wurde ebenfalls ge zeigt, wie sich eine hierarchische Blackboard-Architektur für die Integration und En wicklung von PPS-Teilsystemen eignet.

- Der entwickelte Prototyp eignet sich noch nicht für einen realen Einsatz. Es zeig sich aber, daß die Werkzeuge, die HBBS zur Verfügung stellt, für die gegeben Aufgabenstellung geeignet und ausreichend sind.

Die Untersuchung in [12] zeigt außerdem, daß auch eine dynamische Anpassung de hierarchischen Blackboard-Struktur von HBBS möglich ist. Die Hierarchie-Struktur kan im Falle der Überlastung bestimmter Kommunikationswege so geändert werden, da dieser Engpaß vermieden wird. Ebenfalls kann bei einer Überlastung eines Blackboard durch Aufteilung des Blackboards auf zwei Blackboards dieser Engpaß überwunde werden. Somit wird auch der technikorientierten Sichtweise Genüge geleistet. Wir sehe somit unsere Forschungsbestrebungen als einen ersten Schritt an, die oben erwähnte VPPS-Forschungsbestrebungen zusammenzuführen.

Literatur

[1] Burgard, E., Nissing, T.: Dezentrale Produktionsplanung- und -steuerung, in: CIM-Management, Heft 1 1987, S.26-38; Nissing, T.: Beitrag zur Entwicklung eine Dezentralen Produktionsplanung und -steuerungsystem auf der Basis von verteilte Datenbeständen, Dissertation, Aachen, 1982

[2] Kurbel, K., Rautenstrauch, C.: Ein verteiltes PPS-System auf Arbeitsplatzbasis, in: M.Paul (Hrsg.): Computergestützter Arbeitsplatz, Informatik Fachberichte 223, Berlin et al.: Springer, 1989, S.476-490; Bolte, C., et al.: Integration of Knowledge-Based Modules into a Distributed Production Planning and Control System, in: A M. Tjoa, R. Wagner (Hrsg.): Database and Expert Systems Applications, Springer-Verlag, 1990, S.101-104; Rautenstrauch, C.: Entwicklung und Evaluierung eines verteilten PPS-Systems auf Basis von ORACLE, in: HMD, 157, 1991, S.45-56

[3] Hackstein, R., Kemmner, G.-A.: Dezentralisierung macht die PPS leistungsfähiger, in: Management Zeitschrift 60 (1991) Nr.4, S.69-72; Kemmner, G.-A.: Anwenderorientierte Dezentralisierung von PPS-Systemen, R. Hackstein (Hrsg.): Forschung für die Praxis Band 39, Berichte aus FIR und IAW der TH Aachen, Berlin, Springer, 1991

[4] Philipp, M.: Erfassung und Analyse der Netzanwendungen bei HDM, Arbeitsbericht, Projekt-Nr. 09.34.40, Abteilung P 41.2, Heidelberger Druckmaschinen AG, Wiesloch, September 1990

[5] Mertens, P. et al.: Ein Ansatz zu kooperierenden Expertensystemen bei der Produktionsplanung und -steuerung, in: K. Kurbel et al. (Hrsg.): Interaktive betriebswirtschaftliche Informations- und Steuerungssysteme, Berlin, de Gruyter, 1989, S.13-40

[6] Fox, M.S., Smith, S.F.: ISIS - A Knowledge-Based System for Factory Scheduling, in: Expert Systems, The International Journal of Knowledge Engineering, Vol. 1, Juli 1984, Learning Information Inc., Medford, NJ, S.25-49; Smith, S.F.: A Constraint-Based Framework for Reactive Management of Factory Schedules, in: M.D. Oliff (Hrsg.), Intelligent Manufacturing, Proc. from the First Intnl. Conf. on Expert Systems and the Leading Egde in Production Planning and Control, Menlo Park, Benjamin/Cummings, 1988, S.113-130

[7] Sauve, B., Collinot, A.: An expert system for scheduling in a flexible manufacturing system, in: Robotics & Computer-Integrated Manufacturing, 1987, Vol. 3, Nr. 2, S.229-233; Collinot, A., Le Pape, C., Pinoteau, G.: SONIA: A knowledge-based scheduling system, in: Artificial Intelligence in Engineering, 1988, Vol. 3, Nr. 2, S.87-95

[8] Erman, L.D., Hayes-Roth, F. et al. The Hearsay-II speech-understanding system: Integrating knowledge to resolve uncertainty, in: ACM Comput. Surv., Vol. 12, Nr. 2, 1980, S. 213--353

[9] Mertens, P.: Ausgewählte Ans tze zur Expertensystem-Unterstützung der Produktionsplanung und -steuerung, in: CIM-Management 5, 1991, S.74-77

[10] Weiss, M., Stetter, F.: A Hierarchical Blackboard Architecture for Distributed AI Systems, in: G. Tortora (Hrsg.), Proc. 4th Int. Conf. Softw. Eng. Knowl. Eng., IEEE Press, Los Alamitos, CA, 1992, S. 349-355

[11] Weiss, M.: A Hierarchical Blackboard Architecture for Control, in: K. Sundermeyer, B. Burmeister (Hrsg.), GWAI-92 Workshop Distributed Artificial Intelligence, Bonn, 1992

[12] Hirth, I.: Dynamische Reorganisation Hierarchischer Blackboard-Strukturen, Diplomarbeit am Lehrstuhl für Praktische Informatik I, Universität Mannheim, 1993

DASEDIS –
Eine Entwicklungsumgebung zum Agenten-Orientierten Programmieren

Birgit Burmeister
Daimler-Benz AG, Forschung Systemtechnik
Alt-Moabit 91 b, W–1000 Berlin 21
bur@DBresearch–berlin.de

1 Einleitung

Der Begriff "agenten-orientiertes Programmieren" wurde von Shoham in [Shoham 90] eingeführt und bezeichnet die Weiterentwicklung oder Verfeinerung des objekt-orientierten Programmierens: Objekte, jetzt Agenten genannt, werden mit "mentalen" Fähigkeiten (wie Wissen, Zielen, Absichten) ausgestattet, so daß sie eine größere Autonomie als Objekte erlangen. Die Kommunikation zwischen den Agenten ist stark typisiert, basiert auf der Spechakttheorie von Searle [Searle 69], und kann so auf der Ebene der Kooperation (mit Formen wie "anbieten", "vorschlagen") betrachtet werden. Die agenten-orientierte Programmierung erlaubt damit eine Programmierung auf einer höheren Abstraktionsebene, komplexe Problemstellungen werden durch die Modellierung als "Multi-Agenten-System" auf einfachere Art und Weise strukturiert.

Zur schnellen und effizienten Entwicklung eines Multi-Agenten-Systems reicht allerdings allein die Verwendung einer agenten-orientierten Programmiersprache, wie sie beispielsweise von Shoham mit AGENT0 [Shoham 91] vorgeschlagen wurde, nicht aus. Es muß eine umfassende Entwicklungsumgebung bereitgestellt werden, die nicht nur die Programmierung mit Werkzeugen unterstützt, sondern darüberhinaus eine methodische Unterstützung bietet. Für solche Entwicklungsumgebungen wurde der Begriff *agent factory* vorgeschlagen [O'Hare/Wooldridge 92].

Die vorgestellte Entwicklungsumgebung DASEDIS ("Development And Simulation Environment for Distributed Intelligent Systems") ist in diesem Sinne ein Schritt in Richtung einer *agent factory*. DASEDIS bietet einen konzeptionellen und architektonischen Rahmen für die Entwicklung von Multi-Agenten Systemen und beinhaltet

- Richtlinien, wie eine Domäne als Multi-Agenten-Szenario zu konzeptualisieren ist,
- ein allgemeines Agentenmodell und generische Modelle der Interaktionen zwischen Agenten sowie Agent und Umwelt,
- eine Anleitung, wie die Fähigkeiten von Agenten auf das Agentenmodell abgebildet werden können,
- wohldefinierte, reproduzierbare Kooperationsmuster und
- entsprechende Entwicklungswerkzeuge.

DASEDIS wurde und wird innerhalb des Projektes COSY ("COoperating SYstems") entwickelt. In COSY sollen die Grundlagen für das agenten-orientierte Programmieren gelegt werden. Die Arbeiten in COSY betreffen zum einen die Untersuchung von Methoden und Verfahren, die auf ihre allgemeine Verwendbarkeit geprüft und als Teil von DASEDIS implementiert werden. Zum anderen werden diese Methoden und Verfahren anhand unterschiedlicher Anwendungen getestet.

Im folgenden Abschnitt wird zunächst ein allgemeiner Überblick über DASEDIS gegeben. Es wird dargestellt, welches Agentenmodell hinter DASEDIS steht und wie sich daraus die Architektur der Entwicklungsumgebung ableitet. Im Abschnitt 3 wird die Agentenarchitektur von DASEDIS im Detail beschrieben und eine Sprache zur Agenten- bzw. Wissensdefinition vorgestellt, mit der die eigentliche agenten-orientierte Programmierung erfolgt. Im Abschnitt 4 werden Werkzeuge beschrieben, die DASEDIS zur Unterstützung des agenten-orientierten Programmierens bereitstellt. Eine Zusammenfassung und ein Ausblick auf den weiteren Ausbau von DASEDIS bilden den Abschnitt 5.

2 DASEDIS im Überblick

Das allgemeine Agentenmodell des COSY-Projektes [Sundermeyer 90] beschreibt einen Agenten durch das Tupel (Wahrnehmung, Handlungen, Absichten, Ressourcen): Jeder Agent nimmt gewisse andere Agenten und einen Ausschnitt des Umfeldes wahr, handelt gemäß Absichten und benötigt Ressourcen zur Verwirklichung von Absichten, d.h. zum Handeln. Aus diesem Agentenmodell wurde in COSY eine modulare Agenten-Architektur abgeleitet, bei der die unterschiedlichen Verhaltensformen (Verhalten = Wahrnehmung + Handlungen) unterschiedlichen Modulen zugeordnet wurden:

- Das Modul COGNITION übernimmt die "kognitiven" Handlungen des Agenten, d.h. solche Handlungen, die von anderen Agenten nicht wahrgenommen werden.
- ACTUATORS beinhaltet alle effektorischen Handlungen, d.h. solche die von anderen Agenten wahrgenommen werden können und unmittelbar Auswirkungen in der Umgebung des Agenten haben.
- SENSORS bildet das Wahrnehmungsverhalten ab, bei dem andere Agenten nicht explizit beteiligt sind.
- Das Modul COMMUNICATION ist zuständig für die explizite Kommunikation mit anderen Agenten, d.h. das Empfangen und Senden von Nachrichten.
- INTENTION stellt die langfristigen Ziele, Grundeinstellungen, Rollen usw. eines Agenten dar.

Das Kernstück jedes Agenten bildet das Modul COGNITION, welches in DASEDIS als wissensbasiertes System realisiert ist. Alle anderen Module der Agenten-Architektur sowie das Umfeld der Agenten (ENVIRONMENT) werden anwendungsabhängig simuliert.

Aus dieser Sichtweise ergibt sich der grobe Aufbau von DASEDIS, den Abbildung 1 zeigt: DASEDIS enthält eine Simulationskomponente, die die Modelle zur Simulation von INTENTION, COMMUNICATION, SENSORS, ACTUATORS und ENVIRONMENT enthält. Eine Entwicklungskomponente dient vor allem der Entwicklung der einzelnen wissensbasierten Systeme. Diese beiden Komponenten liegen unter einer gemeinsamen graphischen Benutzeroberfläche. Darin eingebettet sind die wissensbasierten Systeme, d.h. die COGNITION-Module der Agenten.

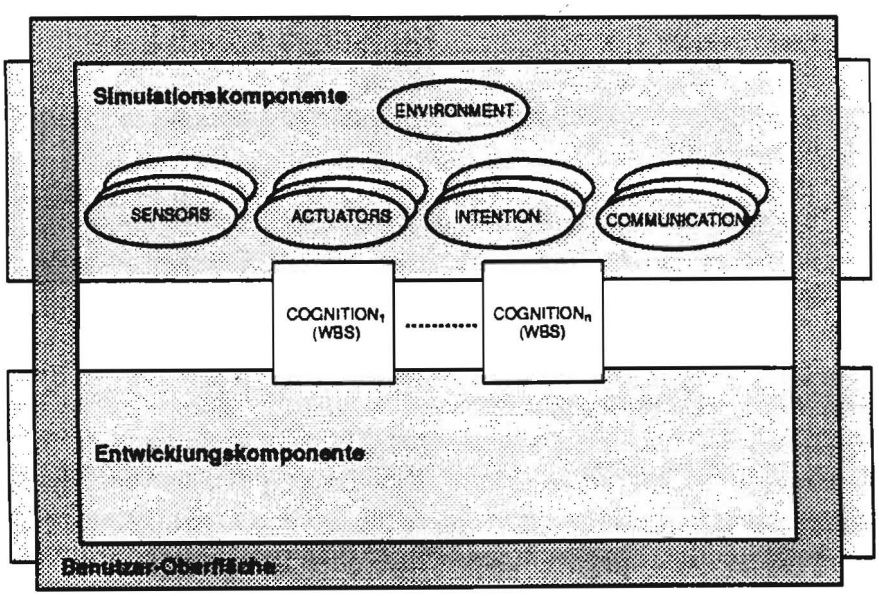


Abb. 1: DASEDIS Architektur

DASEDIS enthält sowohl anwendungsunabhängige Teile, DASEDIS-Kern genannt, als auch eine "DASEDIS-Hülle" für jede spezifische Anwendung COx oder COy (siehe Abbildung 2).

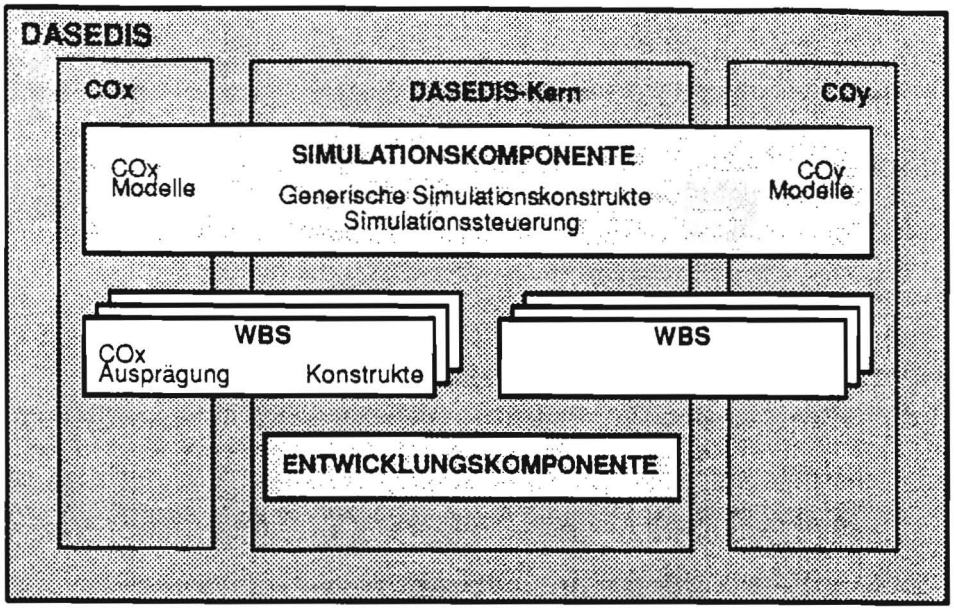


Abb. 2: DASEDIS-Kern und -Hülle

Die Entwicklungskomponente liegt vollständig im DASEDIS-Kern.

Der Bestandteil der Simulationskomponente im Kern enthält generische Konstrukte zu de Simulationsmodellen, die in den konkreten Simulationsmodellen der jeweiligen Anwendunge verwendet werden. Weiterhin ist im Kern die Simulationssteuerung realisiert.

Auch die Benutzeroberfläche hat Bestandteile im DASEDIS-Kern und in der DASEDIS-Hülle. I der Hülle sind die anwendungsspezifischen Funktionen für die Visualisierung und Animatio enthalten.

Die Strukturierung der wissensbasierten Systeme, d.h ihre Aufteilung in Wissensbasis sowi Problemlösungs- und Kooperationskomponente (s. Abschnitt 3), ist Teil des DASEDIS-Kerns Ebenso das Verfahren zum anwendungsunabhängigen Problemlösen, das in der Problem lösungs- und Kooperationskomponente abläuft. Es greift auf die ebenfalls im DASEDIS-Ker festgelegte Strukturierung des Wissens zu. Das anwendungsspezifische Wissen muß in dies Struktur eingepaßt werden.

Bei der Implementierung und Untersuchung einer Anwendung in DASEDIS sind drei Arbeits schritte zu unterscheiden.

- Nach der Konzeptualisierung der Anwendung als Multi-Agenten-Szenario mit eine Abbildung auf das COSY-Agentenmodell erfolgt die 'Programmierung' der Agenten. Daz muß das Wissen der Agenten definiert, die Simulationsmodelle für die simulierten Agenten module (und ENVIRONMENT) müssen formuliert und anwendungsabhängige Animations funktionen bereitgestellt werden.
- Nachdem die Agenten programmiert wurden, werden Agententypen definiert und Simula tionsszenarien und -fälle konfiguriert.
- Schließlich können dann Simulationsfälle 'abgearbeitet' werden, d.h. die kooperativ Erarbeitung einer Problemlösung in der Anwendung simuliert und visualisiert werden.

Die in DASEDIS bereitgestellten Werkzeuge, die sowohl in der Simulations- als auch in de Entwicklungskomponente liegen, unterstützen das oben genannte Vorgehen in dreierle Hinsicht: Zur Wissensdefinition steht eine "Agenten-/Wissensdefinitionssprache" zur Ver fügung, in der das Verhalten der Agenten in Form von Skripten und Protokollen, die zugehörige Absichten und benötigte Ressourcen beschrieben werden. Standard-Kooperationsprotokoll werden im DASEDIS-Kern bereitgestellt und können für jede Anwendung verwendet werden. Zu Definition und Konfigurierung von Szenarien sind "Entwicklungswerkzeuge" vorhanden, di eine graphische Definition von Agententypen, und darauf aufbauend die Konfigurierung vo Simulationsszenarien und Standard-Simulationsfällen erlauben.

Mit Hilfe der Werkzeuge in der Simulationskomponente von DASEDIS können Simulationsläufe kontrolliert werden.

Schließlich dienen die "Experimentierwerkzeuge" Inspector und Observer der Untersuchung und Beobachtung von Agenten während einer Simulation.

3 Agenten in DASEDIS

3.1 Architektur

Agenten in DASEDIS haben die im vorigen Abschnitt erläuterte modulare Architektur, bestehend aus dem als wissensbasiertem System realisiertem Modul COGNITION und den simulierten Modulen SENSORS, ACTUATORS, COMMUNICATION und INTENTION. Abbildung 3 zeigt zwei Agenten dieser Architektur in einem Umfeld ENVIRONMENT.

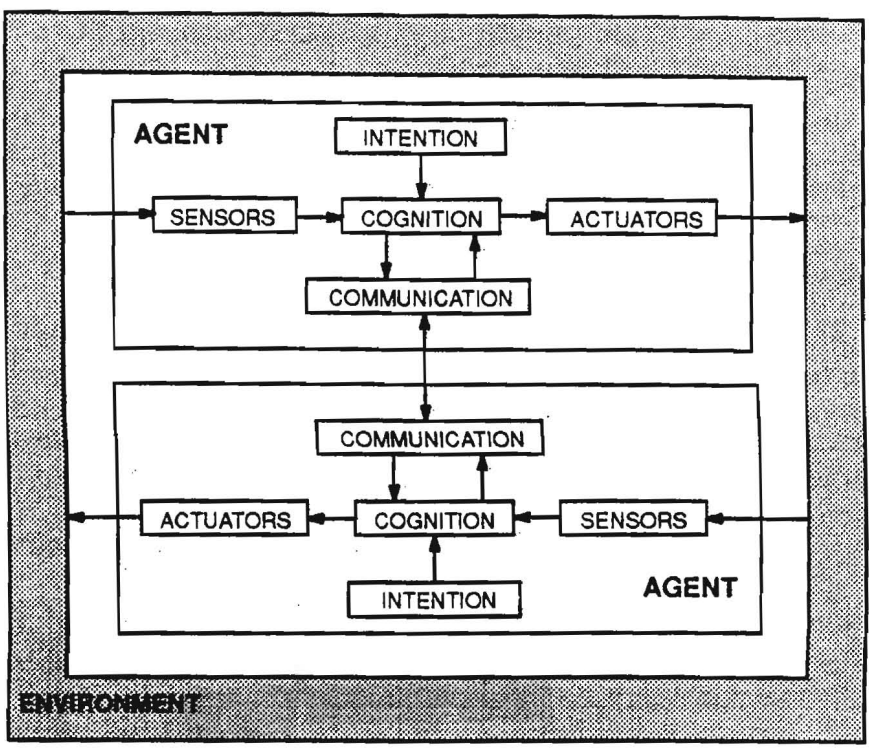


Abb. 3: Agenten-Architektur von DASEDIS

Die Module eines Agenten, mit Ausnahme von COGNITION, werden anwendungsabhängig simuliert und haben damit keine im DASEDIS-Kern festgelegte Architektur.

Das Modul COGNITION hat eine gegenüber herkömmlichen wissensbasierten Systemen erweiterte Architektur (s. Abbildung 4). Neben einer Wissensbasis und einer Problemlösungskomponente (PC) ist zusätzlich eine Kooperationskomponente (CC) vorhanden.

Die Wissensbasis ist in drei Teile eingeteilt: Wissen des Agenten über sich selbst, über andere Agenten und über das Umfeld. In jedem dieser Anteile ist sowohl generisches als auch aktuelles Wissen vorhanden. Generisches und aktuelles Wissen über den Agenten selbst und andere Agenten betrifft Absichten, Verhalten und Ressourcen. Wissen über das Umfeld betrifft nur die Ressourcen, die das Umfeld grundsätzlich bzw. aktuell bereitstellt.

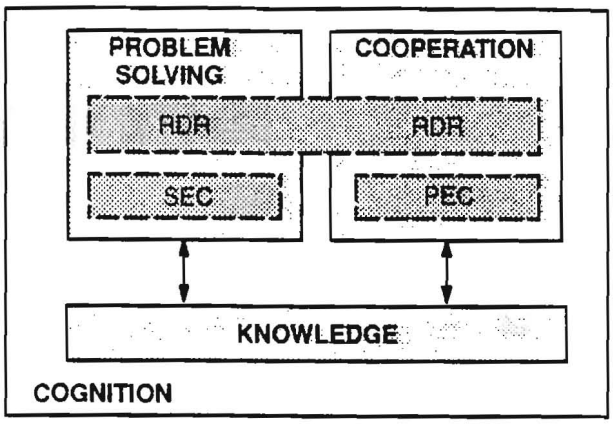


Abb. 4: COGNITION Architektur

Die Problemlösungskomponente PC enthält zum einen eine Skriptabarbeitungskomponente (SEC, *script execution component*) und den Teil der Entscheidungskomponente (RDR, *Reasoning, Deciding, Reacting*), der für die lokale Problemlösung des Agenten benötigt wird.

Die Kooperationskomponente CC beinhaltet eine Protokollabarbeitungskomponente (PEC, *protocol execution component*) und die Teile von RDR, die bei der Interaktion/Kooperation mit anderen Agenten benötigt werden.

Die in PC und CC verwendeten Algorithmen sind anwendungsunabhängig und Teil des DASEDIS-Kerns [Burmeister/Sundermeyer 92]. Wie das anwendungsabhängige Wissen definiert und in die Wissensbasis eingetragen wird, wird im folgenden Abschnitt erläutert.

3.2 Wissensdefinition

Der Teil des Wissens, durch den die eigentliche Programmierung eines Agenten erfolgt, ist der generische Teil des Wissens des Agenten über sich selbst. In ihm wird das Verhaltensrepertoire des Agenten, seine Absichten und die zur Ausführung des Verhaltens benötigten Ressourcen beschrieben. Die anderen generischen Anteile der Wissensbasis, das generische Wissen über andere Agenten (d.h. über Agententypen) und das generische Wissen über das Umfeld werden in der derzeitigen Ausbaustufe von DASEDIS noch nicht verwendet. Die aktuellen Anteile der Wissensbasis werden dynamisch während des kooperativen Problemlösens aufgebaut.

Das Verhaltensrepertoire eines Agenten wird in Form von Skripten und Protokollen festgelegt. Skripte beschreiben die Einzelschritte eines Verhaltens des Agenten; Protokolle beschreiben die einzelnen Kommunikationsschritte einer Kooperation zwischen zwei Agenten.

Skripte

Skripte beschreiben die typischen einzelnen Schritte eines Verhaltens eines Agenten und deren zeitliche Abfolge. Einzelschritte eines Skriptes können Sub-Skripte sein, Protokollaufrufe oder primitive Aktionen/Handlungen. Primitive Aktionen sind entweder primitive kognitive Aktionen, d.h. Aufrufe von benutzerdefinierten Funktionen, die in COGNITION definiert sind (z.B. zur Berechnung eines Wertes). Oder es handelt sich um primitive aktorische Aktionen, d.h. Aufrufe von Schnittstellenfunktionen, die im simulierten Modul ACTUATORS ausgeführt (d.h. simuliert) werden (z.B. fahren eines Fahrzeuges). Bedingungen für Verzweigungen innerhalb eines Skriptes sind das Vorhandensein von Ressourcen oder Prioritäten von nachfolgenden Schritten. Skripte können Parameter haben.

Zur Definition von Skripten eignet sich eine graphische Notation. Diese Notation besteht aus einem Wurzelknoten, der den Namen und die Parameter des Skriptes angibt, und Folgeknoten, die die einzelnen Schritte des Skriptes darstellen (s. Abbildung 5).

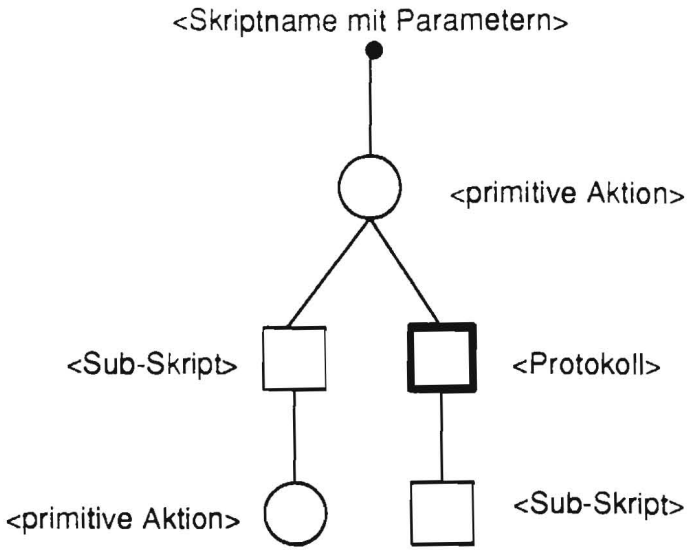


Abb. 5: Graphische Notation für Skripte

Manche Skripte (sogenannte *top-level*-Skripte) sind nur in bestimmten Situationen anwendbar. Dies kann eine Situation in der Umgebung des Agenten sein, die über SENSORS wahrgenommen wird, oder ein interner Zustand der Wissensbasis. Die Situationen und ihre Zuordnung zu Skripten werden getrennt von den Skripten beschrieben. Diese Trennung, der oft "Eingangs- oder Initialbedingungen" genannten Situationen von den Skripten, erlaubt die Verwendung eines Skriptes in unterschiedlichen Situationen.

In manchen Situationen sind mehrere Skripte anwendbar. In diesem Fall muß eine Entscheidung getroffen werden, welches der Skripte zur Ausführung gelangen soll. Diese Entscheidung erfolgt mit Hilfe des INTENTION-Moduls des Agenten. Die zu diesem Zweck vorhandene Modellierung von INTENTION in der Wissensbasis, favorisiert jeweils ein Skript, welches dann gewählt wird. An einer Verbesserung der Einbindung von INTENTION wird derzeit im COSY-Projekt gearbeitet, s.a. [Haddadi 93].

Zur Ausführung eines Skriptes werden Ressourcen benötigt. Bei den Ressourcen wird unterschieden zwischen Ressourcen über die ein Agent selbst verfügt oder die er allein beschaffen kann, und solchen Ressourcen, zu deren Beschaffung mit anderen Agenten kooperiert werden muß. Ressourcen werden durch die Art ihrer Beschaffung beschrieben. Verfügt der Agent selbst über die Ressource, so kann sie aus einem simulierten Agentenmodul oder durch die Ausführung eines Skriptes bzw. einer primitiven Aktion besorgt werden. Muß zur Ressourcenbeschaffung mit anderen Agenten kooperiert werden, so wird bei der Ressource ein Kooperationsprotokoll angegeben, mit dem die Beschaffung erfolgen kann.

Alle Wissensarten eines Agenten, d.h. Verhalten, Absichten und Ressourcen, werden als *frame*-artige Strukturen repräsentiert. Die DASEDIS-Wissensdefinitionssprache stellt spezielle Definitionsmakros, wie `def-script`, `def-strategic-intention`, `def-own-resource`, zur Verfügung, mit denen festgelegte Attribute beschrieben werden können. Die Definitionsmakros sorgen auch für die Einordnung eines definierten Skriptes, einer Ressource usw. in den richtigen Teil der Wissensbasis.

Protokolle

Kooperationsprotokolle beschreiben den Ablauf der Kommunikation zwischen zwei Agenten, d.h. die Abfolge der zwischen den beiden Agenten ausgetauschten Nachrichten.

Zur Vereinfachung der Kommunikation und Interpretation der Nachrichten werden in DASEDIS, aufbauend auf der Sprechakttheorie [Searle 69] Nachrichten mit einem Typ versehen.

Zu jedem Protokoll gehört ein Nachrichtentyp durch den das Protokoll eindeutig identifiziert wird. Dieser Nachrichtentyp kennzeichnet den ersten Schritt in einem Protokoll, zu dem auch die primitiven kommunikativen Verhalten gehören, die der Sender/Empfänger bei Abarbeitung dieses Knotens, d.h. zum Senden/Verarbeiten der ersten Nachricht, ausführen. Weitere Schritte in einem Protokoll sind die Aufrufe von (Sub-)Protokollen. Hier muß zusätzlich angegeben werden, welcher der zwei beteiligten Agenten in diesem Knoten der aktive Agent, d.h. der Sender, ist.

Eine graphische Notation für Protokolle zeigt Abbildung 6. Der erste Knoten zeigt stets den das Protokoll identifizierenden Nachrichtentypen mit den durch Sender bzw. Empfänger ausgeführten primitiven kommunikativen Verhaltensweisen. Die Knotenfarbe bei Subprotokoll-Aufrufen zeigt an, welcher der zwei beteiligten Agenten in diesem Knoten der aktive Agent ist: Weiß steht für den Sender der ersten Nachricht, also den Initiator des Protokolls, grau für den Empfänger der ersten Nachricht.

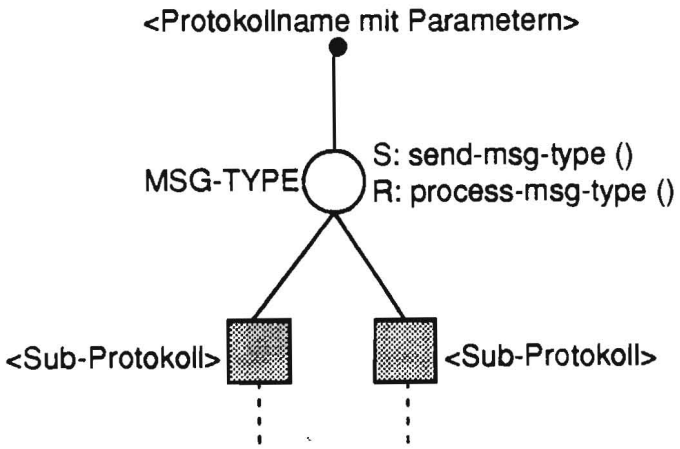


Abb. 6: Graphische Notation für Protokolle

Die zur Ausführung von Protokollen benötigten Ressourcen müssen nicht explizit definiert werden, da es sich nur um interne Ressourcen des Agenten handelt.

Auch Protokolle werden als *frame*-artigen Stukturen repräsentiert. Die Definition und Einordnung in die Wissensbasis erfolgt mit dem Definitionsmakro `def-protocol`.

Der DASEDIS-Kern stellt allerdings verschiedene Basisprotokolle zur Verfügung, die in jeder Anwendung verwendet oder erweitert werden können [Burmeister/Haddadi/Sundermeyer 93]. Dies sind die Protokolle *Informing* (über Inhalte der Wissensbasis informieren), *Querying* (Inhalte der Wissensbasis anfragen), *Commanding* (Ausführen einer Handlung befehlen), *Offering* (Ausführen einer Handlung oder Ressourcen anbieten), *Demanding* (Ausführen einer Handlung verlangen), *Proposing* (etwas vorschlagen) und *Requesting* (etwas ersuchen).

Neben der Wissenseingabe müssen zur vollständigen Beschreibung eines Agenten die anwendungsabhängigen Simulationsmodelle für SENSORS, ACTUATORS, COMMUNICATION und INTENTION bereitgestellt werden. Hierfür wird jedoch von DASEDIS keine weitere Definitions- oder Werkzeugunterstützung angeboten.

4 Werkzeuge in DASEDIS

4.1 Entwicklungswerkzeuge

Nachdem Agenten und Umfeld in einer Anwendung vollständig programmiert wurden, erlauben die Entwicklungswerkzeuge von DASEDIS die Definition und Konfiguration von Agententypen, Simulationsszenarien und Standardsimulationsfällen (kurz Standards).

Unter einem Agententyp wird in DASEDIS eine spezielle Kombination von Typen der simulierten Agentenmodule (ACTUATORS usw.), einer generischen Wissensbasis und einem graphischen Symbol verstanden. Agententypen werden innerhalb einer Anwendung definiert. Für die einzelnen simulierten Agentenmodule kann die Klasse eines Simulationsmodells festgelegt, sowie ggf. Parameter des Modells mit Werten vorbelegt werden. Für das Modul COGNITION kann der Name einer Wissensbasis eingegeben werden.

Neben der Erzeugung von vollständig neuen Agententypen können Untertypen oder Typkopien gebildet werden. Die Eingaben erfolgen über die graphische Darstellung der Agentenarchitektur (s. Abbildung 7).

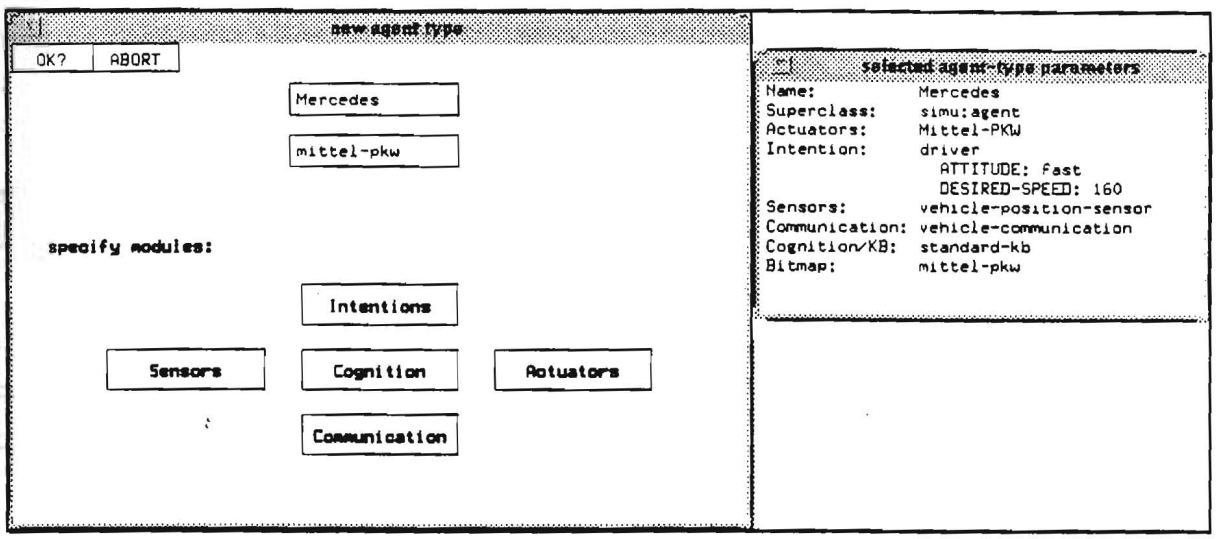


Abb. 7: Definition von Agententypen

Ein Simulationsszenario ist die Kombination von Agententypen mit einem Umfeldtyp. Ein Szenario wird durch Auswahl von Agententypen und Umfeldtyp in Menüs zusammengestellt und ist dann in der DASEDIS-Simulationskomponente ebenfalls über ein Menü auswählbar.

Ein Simulationsfall in einem Szenario enthält instantiierte Umfeld- und Agententypen, d.h. eine Menge von Agenten "existiert" in einem konkreten Umfeld. Ein beliebiger Simulationsfall kann zum Standard erklärt werden. Ein Standard kann jederzeit abgerufen werden und muß nicht vor jedem Simulationslauf neu erzeugt werden.

4.2 Simulations- und Experimentierwerkzeuge

In der Simulationskomponente von DASEDIS kann nach der Auswahl eines Szenarios und der Zusammenstellung eines Simulationsfalles oder Auswahl eines Standards die Simulation erfolgen: sie kann gestartet und angehalten werden, ein Rücksetzen auf einen definierten Ausgangszustand ist möglich oder das Re-Initialisieren durch Löschen aller Agenten.

DASEDIS bietet zwei Experimentierwerkzeuge zur Überwachung der Agenten, der Agenten-Module und speziell des Agenten-Moduls COGNITION an.

Inspector

Ein Inspektionswerkzeug dient zum Inspizieren/Untersuchen der Agenten und ihrer Module. Über die graphische Darstellung der Agenten-Architektur können während der Simulation die simulierten Agentenmodule und das Modul COGNITION untersucht werden. Als Teil von COGNITION kann die Wissensbasis in ihrer hierarchischen Struktur bis zu einzelnen Elementen, wie Skripten, Ressourcen usw. angesehen werden. Die Abbildungen 8 und 9 zeigen die Fenster des *Inspectors.*

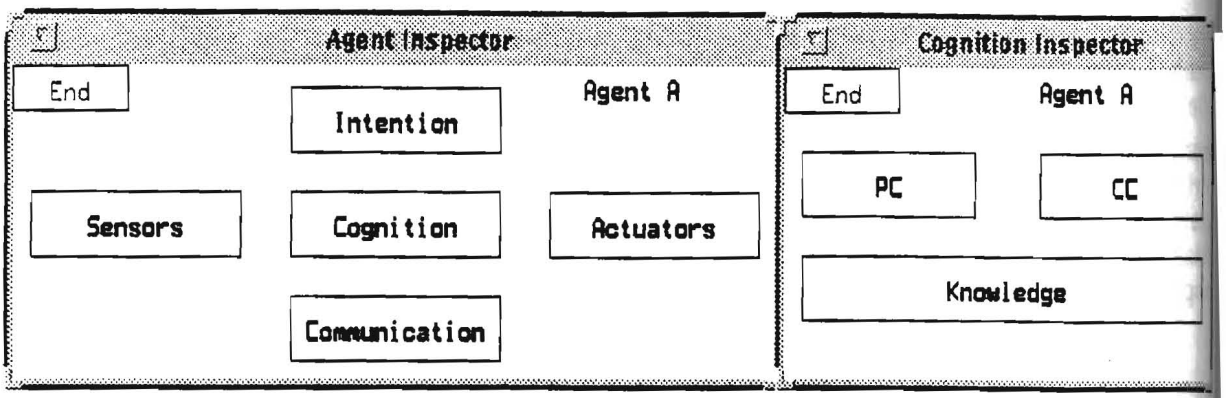


Abb. 8: DASEDIS *Inspector* für Agent und COGNITION

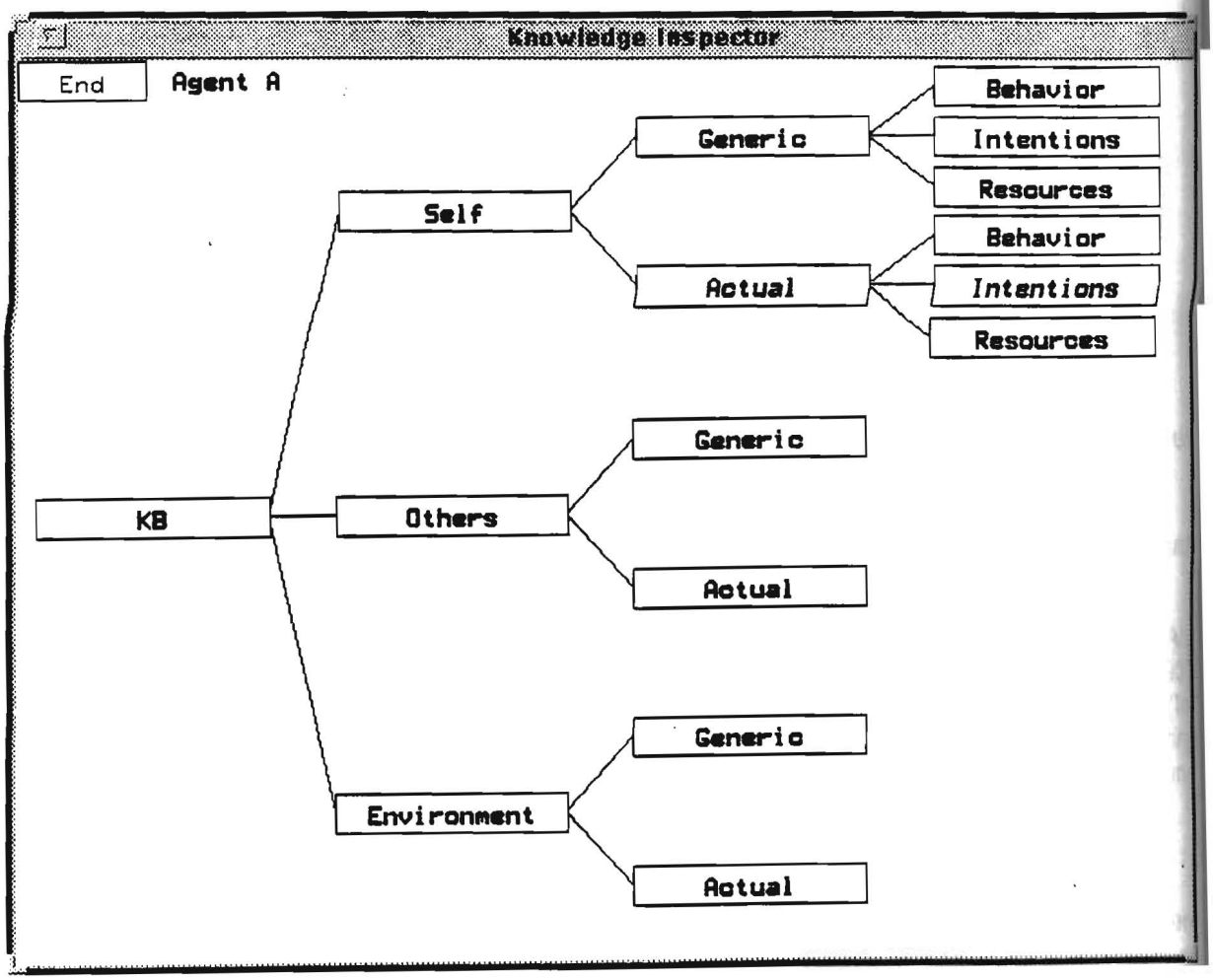


Abb. 9: DASEDIS *Inspector* für die Wissensbasis

Observer

Mit dem Beobachtungswerkzeug kann das kooperative Problemlösen eines Agenten sowie die Kommunikation eines Agenten bzw. des gesamten Systems beobachtet werden.

Bei der Beobachtung des kooperativen Problemlösens eines Agenten wird der Beginn der Ausführung eines Verhaltens (Skriptes, Protokolls oder primitiven Verhaltens) protokolliert. Abbildung 10 zeigt ein Beobachtungsfenster.

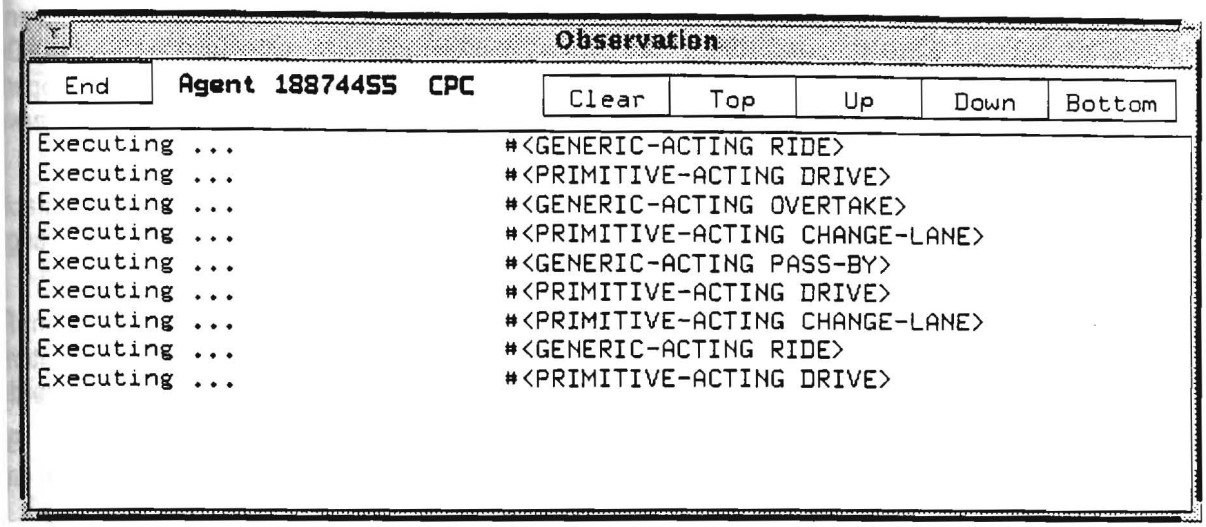


Abb. 10: Beobachtung des kooperativen Problemlösens

Wird die Kommunikation eines einzelnen Agenten beobachtet, so werden die Nachrichten, die der ausgewählte Agent versendet und empfängt protokolliert. Bei der Beobachtung der globalen Kommunikation wird jeweils das Senden der Nachrichten angezeigt. Ausgegeben werden die Nachrichtenidentifikation und der Nachrichtentyp, Absender und Empfänger (1. Zeile), der eigentliche Inhalt der Nachricht und die Identifikation einer Referenznachricht (2. Zeile). Abbildung 11 zeigt die Beobachtung der Kommunikation eines einzelnen Agenten.

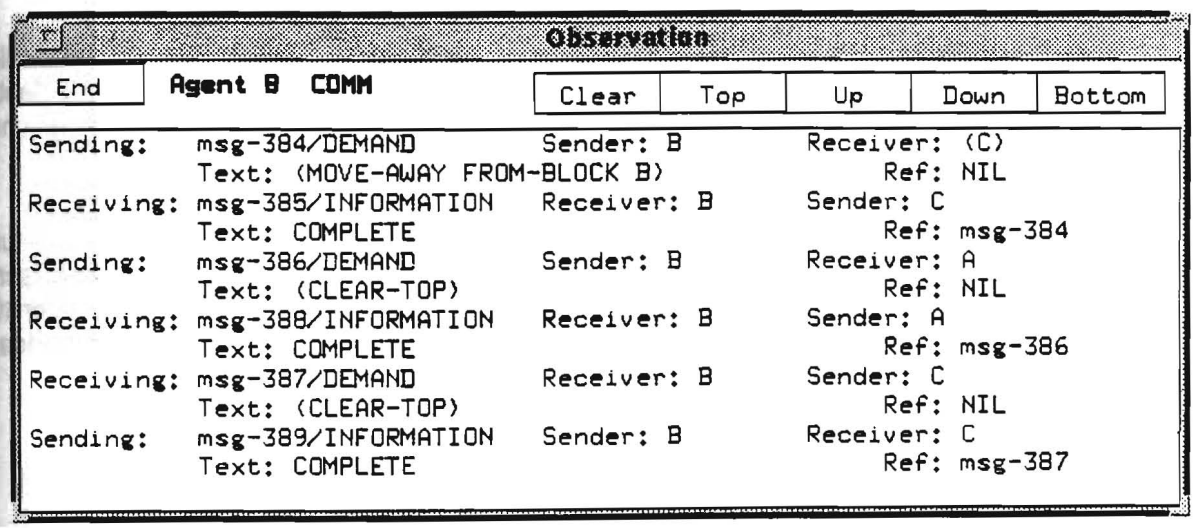


Abb. 11: Beobachtung der Kommunikation eines Agenten

5 Zusammenfassung und Ausblick

Vorgestellt wurde das Werkzeug DASEDIS, das im Rahmen des COSY-Projektes entwickelt wird. DASEDIS, dessen Architektur aus der Agenten-Architektur von COSY abgeleitet wurde, enthält unter einer gemeinsamen graphischen Benutzeroberfläche eine Simulations- und eine Entwicklungskomponente, die die Programmierung der Agenten, genauer der COGNITION-Module der Agenten unterstützen. DASEDIS besteht aus einem anwendungsunabhängigen Kern und der jeweils anwendungsspezifischen Hülle.

Die eigentliche agenten-orientierte Programmierung erfolgt in DASEDIS durch die Wissensdefinition, d.h. durch eine Beschreibung des Agentenverhaltens in Form von Skripten und Protokollen und mit ihren zugehörigen Absichten und Ressourcen. DASEDIS stellt hierfür Definitionsmakros bereit sowie eine graphische Notation für Skripte und Protokolle. Zur Kooperation können die im DASEDIS-Kern vorhandenen Basis-Protokolle direkt verwendet oder erweitert werden.

Um die Durchführung von Simulationsläufen zu vereinfachen, erlaubt DASEDIS die (graphisch unterstützte) Definition von Agententypen und darauf aufbauend die Konfigurierung von Simulationsszenarien und Standardsimulationsfällen.

Neben der eigentlichen Kontrolle eines Simulationslaufs durch Starten, Unterbrechen, Rücksetzen usw. können die Agenten während des Simulationslaufs in ihren Interna untersucht sowie das kooperative Problemlösen und die Kommunikation im Detail beobachtet werden.

DASEDIS läuft auf SUN-Sparcstations unter Allegro CommonLisp und X-Windows. Bisher wurden in DASEDIS drei verschiedene Anwendungen implementiert und getestet und so insbesondere die Tragfähigkeit der im DASEDIS-Kern verwendeten Konzepte gezeigt.

Eine Untersuchung vorhandener Werkzeuge und Test-Umgebungen aus dem Bereich DAI (*Distributed Artificial Intelligence*) zeigte, daß Werkzeuge wie MACE [Gasser/Braganza/Hermann 87] oder MICE [Durfee/Montgomery 89] meist experimentellen Charakter haben und eine Programmierung auf höherer Ebene nicht gestatten. Test-Umgebungen, wie z.B. DVMT [Lesser/Corkill 88] sind jeweils auf eine spezielle Anwendung zugeschnitten und daher nicht allgemein verwendbar, wie dies für DASEDIS der Fall ist.

DASEDIS ist (noch) kein fertiges Werkzeug. Im Rahmen des COSY-Projektes werden ständig neue Konzepte in den DASEDIS-Kern aufgenommen. Derzeitige Arbeiten betreffen insbesondere die RDR-(*Reasoning, Deciding, Reacting*)-Komponente in COGNITION (s.a. [Haddadi 93]) und die Ausarbeitung weiterer Kooperationsprotokolle.

Auf der eigentlichen Werkzeugseite sind graphische Editoren für Skripte und Protokolle zur vereinfachten Wissenseingabe angedacht. In diesem Zusammenhang soll auch die Beobachtung der kooperativen Problemlösung eines Agenten, d.h. die Skriptabarbeitung, und die Kommunikation/Kooperation, d.h. die Protokollabarbeitung graphisch erfolgen.

Die durch DASEDIS unterstützte Vorgehensweise der agenten-orientierten Programmierung ist eine konsequente Weiterentwicklung der objekt-orientierten Programmierung. Die Wissensdefinitionsprache von DASEDIS, die Entwicklungs- und Experimentierwerkzeuge unterstützen diese Vorgehensweise nicht durch eine agenten-orientierte Programmiersprache, sondern im Sinne einer echten *agent factory*.

6 Literatur

[Burmeister/Haddadi/Sundermeyer 93] B.Burmeister, A.Haddadi, K.Sundermeyer, "Generic Configurable Cooperation Protocols for Multi–Agent Systems", to be published

[Burmeister/Sundermeyer 92] B.Burmeister, K.Sundermeyer: "Cooperative Problem-Solving Guided by Intentions and Perception", in: E.Werner, Y.Demazeau (eds.), Decentralized A.I.3, North-Holland, 1992

[Durfee/Montgomery 89] E.H.Durfee, T.A.Montgomery "MICE: A Flexible Testbed for Intelligent Coordination Experiments" in: M.Benda (ed.) Proc. Ninth Workshop on Distributed Artificial Intelligence, 1989

[Gasser/Braganza/Herman 87] L.Gasser, C.Braganza, N.Herman, "MACE: A Flexible Testbed for Distributed AI Research" in: M.N.Huhns (ed.), Distributed Artificial Intelligence, Research Notes in Artificial Intelligence, Pitman, Morgan Kaufman, 1987

[Haddadi 93] A. Haddadi, "Eine Hybride Architektur für Mehragenten-Systeme", Gründungs-workshop Fachgruppe 1.1.6 "Verteilte Künstliche Intelligenz", Saarbrücken, 1993

[Lesser/Corkill 88] V.R.Lesser, D.D.Corkill, "The Distributed Vehicle Monitoring Testbed: A Tool for Investigating Distributed Problem Solving Networks" in: R.Engelmore, T.Morgan (eds.), Blackboard Systems, Addison-Wesley, 1988

[O'Hare/Woolridge 92] G.M.P.O'Hare, M.J.Wooldridge, "A Software Engineering Perspective on Multi-Agent System Design: Experience in the Development of MADE" in: N.M. Avouris, L.Gasser (eds.), Distributed Artificial Intelligence: Theory and Praxis, Kluwer, 1992

[Searle 69] J.R.Searle, Speech Acts, Cambridge Univ.Press, 1969

[Shoham 90] Y.Shoham, "Agent Oriented Programming", Technical Report STAN-CS-90-1335, 1990

[Shoham 91] Y.Shoham, "AGENT0: A simple language and its interpretation", Proc. AAAI-91

[Sundermeyer 90] K.Sundermeyer, "Modellierung von Szenarien kooperierender Akteure" in: H.Marburger (ed.), GWAI-90, Springer, Informatik Fachberichte, 1990

PEDE–Lab — Eine Experimentierumgebung für die Verteilte Künstliche Intelligenz

Rolf Reinema, Gerhard Kraetzschmar
Bayerisches Forschungszentrum für Wissensbasierte Systeme (FORWISS)
Forschungsgruppe Wissenserwerb
Am Weichselgarten 7, W-8520 Erlangen
Email: {rfreinem, gkk}@forwiss.uni-erlangen.de

Zusammenfassung

In der Verteilten Künstlichen Intelligenz (VKI) werden große und komplexe Softwaresysteme entwickelt, die aus einer Vielzahl von (teil-)automen Subsystemen bestehen und verteilt an der Lösung einer komplexen Problemstellung arbeiten. Dabei steht man immer wieder vor dem Problem, das Verhalten der entwickelten Softwaresysteme unter möglichst realistischen Einsatzbedingungen zu testen und zu evaluieren, vorhandene Fehler zu finden, Verhaltensanomalien aufzudecken und Aussagen über die Leistungsfähigkeit eines Softwaresystems zu machen.

Konventionelle Verfahren und formale Methoden, wie sie aus dem Bereich des Software Engineering bekannt sind, sowie verfügbare Experimentierumgebungen besitzen eine Reihe von Defiziten, wodurch sie für das Testen und Evaluieren komplexer Softwaresysteme, insbesondere von VKI–Systemen, nicht ausreichen oder nur bedingt eingesetzt werden können.

Im PEDE-Projekt wird mit der Experimentierumgebung PEDE–Lab ein neuer Ansatz für das Testen und Evaluieren komplexer Softwaresysteme verfolgt. Für die Experimentierumgebung PEDE–Lab wurde ein konventionelles Simulationssystem als Basissystem verwendet. Darauf aufbauend wurden Konzepte und Methoden entwickelt, die es ermöglichen, Softwaresysteme in Simulationsmodelle innerhalb des Simulationssystems zu integrieren und eine gemeinsame Simulation durchzuführen.

1 Einleitung und Motivation

In der Verteilten Künstlichen Intelligenz (VKI) werden sehr große und komplexe Softwaresysteme entwickelt, die meist aus einer Vielzahl von (teil-)automen Subsystemen bestehen und verteilt an der Lösung einer komplexen Problemstellung arbeiten. Dabei steht man immer wieder vor dem Problem, das Verhalten der entwickelten Softwaresysteme unter

möglichst realistischen Einsatzbedingungen zu testen und zu evaluieren, vorhandene Fehler zu finden, Verhaltensanomalien aufzudecken und Aussagen über die Leistungsfähigkeit eines Softwaresystems machen zu wollen.

Stehen für die Lösung einer Problemstellung verschiedene alternative Ansätze, Architekturen oder Softwaresysteme zur Verfügung, so will man deren Verhalten in charakteristischen Problemsituationen untersuchen, die ermittelten Ergebnisse analysieren und miteinander vergleichen können, um Aufschlüsse über die Verwendbarkeit in Frage kommender Ansätze und Systeme zu erhalten und die bestmögliche Problemlösung zu finden.

Konventionelle Verfahren und formale Methoden, wie sie aus dem Bereich des Software Engineering bekannt sind, reichen nicht aus oder können nur bedingt eingesetzt werden. Beispielsweise erfordert die Anwendbarkeit formaler Methoden zur Komplexitätsanalyse gerade bei komplexen Softwaresystemen so viele idealisierende Annahmen, daß die entstehenden Modelle nur noch sehr wenig Ähnlichkeit mit den tatsächlichen Systemen haben. Weiterhin lassen sich realistische Einsatzszenarien eines Softwaresystems mit Hilfe mathematischer Modelle nur unbefriedigend nachbilden. Ebenso scheiden häufig auch Untersuchungen eines Softwaresystems in seiner realen Einsatzumgebung aus, da diese meist noch gar nicht existiert oder die Untersuchungen einen unzumutbaren Eingriff auf das Systemverhalten zur Folge haben.

Experimentierumgebungen sind also unverzichtbare Werkzeuge, wenn es darum geht, das Verhalten komplexer Softwaresysteme, insbesondere VKI–Systeme, zu testen und zu evaluieren.

Im folgenden Abschnitt soll dargestellt werden, welche Defizite verfügbare Experimentierumgebungen besitzen und welche Anforderungen an eine Experimentierumgebung für komplexe Softwaresysteme zu stellen sind. Anschließend wird gezeigt, welche Unterstützung die Experimentierumgebung PEDE–Lab dem Anwender bieten kann, das Verhalten eines komplexen Softwaresystems in einem Modell seiner Einsatzumgebung zu untersuchen.

2 Verfügbare Werkzeuge und deren Defizite

2.1 Debugging–Systeme

Im Bereich der konventionellen Softwaretechnik gibt es verschiedene Verfahren für das Testen und Evaluieren von Softwaresystemen. Dazu gehören neben formaler Programmverifikation, symbolischen Tests und statischer Analyse auch dynamische Tests [Reinema 93]. Bei dynamischen Testverfahren wird ein zu untersuchendes Softwaresystem mit Hilfe von Debugging–Systemen direkt in seiner realen Systemumgebung getestet.

Mit den derzeit vorhandenen Debugging–Systemen lassen sich fast ausschließlich interaktive Tests und Experimente durchführen. Bei großen und komplexen Softwaresystemen müssen umfangreiche Testdatensätze interaktiv eingegeben werden. Die Generierung von Ereignissen und Zufallszahlenfolgen, die bestimmten Verteilungsfunktionen angehören sollen, bleibt dem Anwender überlassen.

Auch die Steuerung der Experimentabläufe sowie die Auslösung von asynchronen Er-

eignissen obliegt dem Anwender. Da nicht gewährleistet werden kann, daß für verschiedene Testsituationen die gleichen Rahmenbedingungen reproduzierbar sind, sind die ermittelten Untersuchungsergebnisse im allgemeinen nicht miteinander vergleichbar. Darüber hinaus führen die mangelnde automatische Aufbereitung und Auswertung von umfangreichen Testergebnissen und deren graphische Präsentation dazu, daß eine Verwendung von Debugging-Werkzeugen für das Testen und Evaluieren von komplexen Softwaresystemen wenig praktikabel ist.

Besonders problematisch ist jedoch die Tatsache, daß die meisten der verfügbaren Debugging-Systeme für das Testen verteilter Softwaresysteme ungeeignet sind (siehe hierzu [McDowell 91]).

2.2 Simulationssysteme

Simulationssysteme verfügen über eine Vielzahl von Methoden und Verfahren zur Durchführung von Untersuchungen des Verhaltens komplexer und verteilter Systeme unter vorgegebenen Bedingungen. Dazu gehören unter anderem die Erzeugung von Zufallszahlenfolgen, die bestimmten Verteilungsfunktionen genügen, die Beschreibung von Experimentabläufen und hervorragende Verfahren zur Ergebnisauswertung und -präsentation [SIMPLEX 92].

Simulationsuntersuchungen zum Verhalten eines komplexen Softwaresystems lassen sich jedoch nicht direkt am System selbst, sondern immer nur an einem Modell des zu untersuchenden Systems durchführen. Für die Erstellung solcher Modelle werden dem Anwender von konventionellen Simulationssystemen wie SIMPLEX-II [SIMPLEX 92], unterschiedliche Methoden zur Verfügung gestellt. Dabei lassen sich im wesentlichen die Auswahl bereits vorgefertigter Bausteine aus Bibliotheken und die Beschreibung des dynamischen Verhaltens mit Hilfe spezieller Modellbeschreibungssprachen unterscheiden.

Die Tatsache, daß das Verhalten eines komplexen Softwaresystems nur an einem Modell untersucht werden kann, ist in der Praxis häufig ein K.O.-Kriterium für den Einsatz eines konventionellen Simulationssystems. Dies liegt vor allem daran, daß die Modellierung komplexer Softwaresysteme sowie die Modellvalidierung im allgemeinen schwierig und einen hohen Aufwand an Zeit und damit auch Kosten erfordert. Es gibt komplexe Sachverhalte und Problemstellungen (beispielsweise aus der Verteilten KI), die sich mit den zur Verfügung gestellten Mitteln nur schwer oder gar nicht beschreiben lassen.

2.3 Experimentierumgebungen der VKI

Die im Bereich der Verteilten Künstlichen Intelligenz derzeit existierenden Experimentierumgebungen (siehe u. a. [Bond 88], [Brauer 91], [Huhns 87]) besitzen ihrerseits ebenfalls eine Reihe von Defiziten, die deren Verwendbarkeit einschränkt.

Dazu gehört unter anderem die Beschränkung auf eine eng begrenzte Klasse von Softwaresystemen und Anwendungsszenarien (oft mit relativ geringem Praxisbezug) sowie die Einschränkung auf bestimmte Modellierungsansätze der VKI oder Programmiersprachen. Die Hauptursache hierfür liegt darin, daß die existierenden Experimentierumgebungen aus Ermangelung geeigneter Werkzeuge meist um ein existierendes Softwaresystem, dessen

Verhalten getestet und evaluiert werden sollte, herumgebaut wurden. Aus diesem Grund enthalten viele der existierenden Experimentierumgebungen nur sehr rudimentäre Verfahren zur Ergebnisauswertung und -präsenation, die eine Vielzahl von Wünschen zur Unterstützung eines Anwenders offen lassen.

2.4 Anforderungen an eine Experimentierumgebung

Untersuchungen des Verhaltens komplexer Softwaresysteme in Mini-Welten, wie beispielsweise die Untersuchung des Verhaltens von Planungssystemen in der Klötzen- oder Raumwelt, sind für die praktische Einsatzfähigkeit solcher Systeme nur von geringer Relevanz. Sie können zwar erste Aufschlüsse über mögliche Schwächen der untersuchten Softwaresysteme in solchen Mini-Welten geben, es sind jedoch keine genauen Aussagen über das Verhalten in realen Systemumgebungen möglich, die durch eine Vielzahl komplexer äußerer Einwirkungen und sich dynamisch ändernden Problemsituationen gekennzeichnet sind.

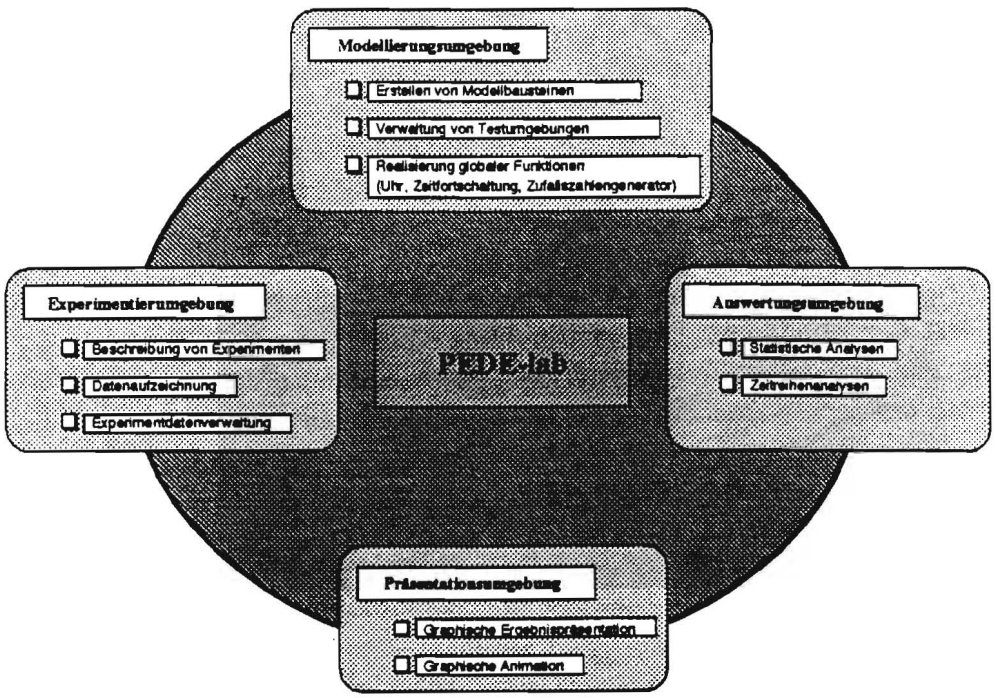


Abbildung 1: Anforderungen an eine Experimentierumgebung

Eine Experimentierumgebung sollte daher eine Modellierungsumgebung zur Verfügung stellen, die es erlaubt, beliebige Modellbausteine für die Komponenten einer künstlichen Systemumgebung zu erstellen und ein zu testendes Softwaresystem darin zu integrieren. Beispielsweise sollte es möglich sein, ein Modell eines Flexiblen Fertigungssystems aufzubauen und ein zu testendes Produktionsplanungssystem in dieses Modell zu integrieren.

Einmal erstellte Testumgebungen sollten in Form sogenannter Modellbanken verwaltet werden, aus denen der Anwender dann für ein bestimmtes Szenario eine bereits existieren-

de Testumgebung auswählen beziehungsweise eine neue Testumgebung aus vorhandenen Bausteinen aufbauen und von ihm erstellte Bausteine und Testumgebungen einfügen kann.

Um Experimente an einem heterogenen Modell, das aus den zu testenden Softwaresystemen und deren Testumgebung besteht, durchführen zu können, sind geeignete Methoden zur Experimentsteuerung erforderlich. Dazu gehören die Initialisierung vom Anwender vorgebbarer Anfangsbedingungen, das Starten, Beenden und gegebenenfalls Unterbrechen von Experimenten sowie die zeitliche Koordination von Abläufen.

Sollen ausgehend von einem einmal erreichten Zustand weitere Untersuchungen durchgeführt werden (z. B. detaillierte Untersuchung der Ursachen einer aufgetretenen Fehlersituation), so sollten derartige (konsistente) Zustände jederzeit wiederherstellbar sein. Hierfür sind geeignete Verfahren zur Verfügung zu stellen, durch die einerseits vom Anwender spezifizierbare Zwischenzustände vollständig aufgezeichnet werden können und die andererseits bei Bedarf ein Wiederherstellen eines aufgezeichneten Zustands ermöglichen.

Eine Experimentierumgebung sollte Verfahren zur selektiven Aufzeichnung von repräsentativen Daten während eines Experiments sowie deren Analyse zur Verfügung stellen. Dabei müssen die aufgezeichneten Daten und ermittelte Ergebnisse eindeutig einem bestimmten Experiment zugeordnet werden können und jederzeit wieder auffindbar sein.

In einer Präsentationsumgebung sollten aufgezeichnete Daten und Auswertungsergebnisse übersichtlich in graphischer Form dargestellt werden können. Dazu gehören beispielsweise Funktionsdiagramme, Histogramme und Kreisdiagramme. Für die Veranschaulichung dynamischer Vorgänge und Zusammenhänge innerhalb eines betrachteten Systems ist eine graphische Animation von großem Nutzen.

Durchzuführende Experimente sollten mit Hilfe einer Experimentbeschreibungssprache spezifiziert werden können. Dies ist besonders dann von großem Nutzen, wenn eine Vielzahl ähnlicher Experimente durchgeführt werden sollen oder die Experimente sich über längere Zeiträume erstrecken und eine ständige Interaktion zwischen Anwender und Experimentierumgebung überflüssig ist. Durch die Verwaltung der durchgeführten Experimente in Form von Experimentbeschreibungen, kann die Reproduzierbarkeit eines Experiments auf elegante Art und Weise erreicht werden.

3 PEDE-Lab

Im Rahmen des PEDE–Projekts [Stoyan 92] sollte für die zu realisierende Experimentierumgebung PEDE–Lab kein von Grund auf neues Simulationssystem mitsamt einer eigenen Modellbeschreibungssprache erstellt werden. Vielmehr wurde auf ein konventionelles Simulationssystem zurückgegriffen, um den nicht unerheblichen Aufwand für die Realisierung von Verfahren beispielsweise zur Erzeugung von Zufallszahlen, Ergebnisauswertung und -präsentation zu reduzieren.

Als Basisystem der Experimentierumgebung PEDE-Lab wurde das Simulationssystem SIMPLEX–II verwendet [SIMPLEX 92]. SIMPLEX–II und die zugehörige Modellbeschreibungssprache SIMPLEX–MDL sind gut geeignete Werkzeuge zur formalen Beschreibung abstrakter Modelle, deren automatische Übersetzung in ablauffähige Simulationsprogram-

me und die anschließende Durchführung von Simulationsexperimenten. Die Modellbeschreibungssprache SIMPLEX-MDL basiert auf der allgemeinen Systemtheorie. SIMPLEX-MDL ist eine deklarative Modellbeschreibungssprache, die einen hierarchischen, modularen Modellaufbau ermöglicht und über ein Klassenkonzept verfügt. Mit ihr lassen sich Modelle der Klassen zeitkontinuierlich, zeitdiskret und transaktionsorientiert aufbauen.

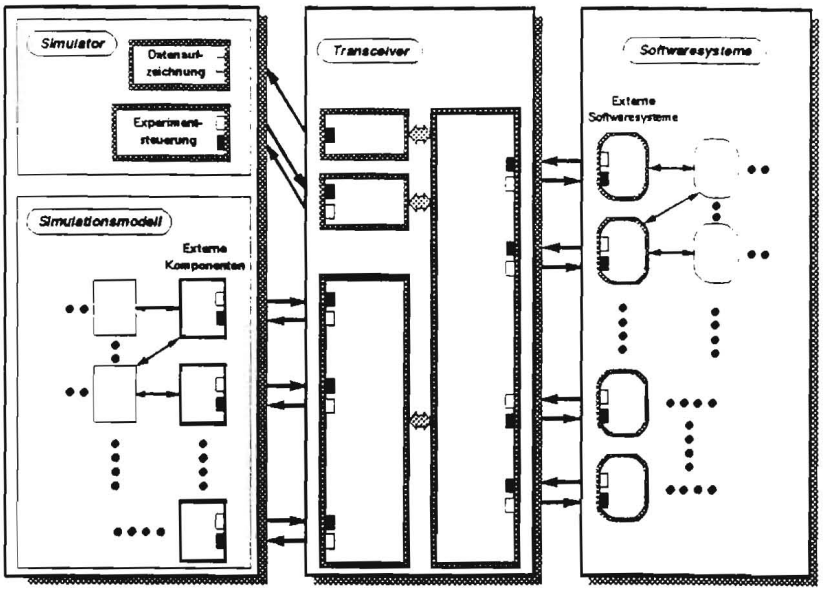


Abbildung 2: PEDE–Lab Systemarchitektur

3.1 Aufbau von Testumgebungen

Für die Experimentierumgebung PEDE–Lab wurde das Simulationssystem SIMPLEX–II so erweitert, daß Softwaresysteme, deren Verhalten in einer ganz bestimmten Systemumgebung untersucht werden soll, in ein Simulationsmodell innerhalb des Simulationsystems integriert werden können und eine gemeinsame Simulation durchgeführt werden kann.

Damit externe Softwaresysteme überhaupt in ein Simulationsmodell integriert werden können, mußte eine Klasse von Modellkomponenten (sogenannte externe Modellkomponenten) geschaffen werden, die nicht nur in der Lage ist, mit anderen Modellkomponenten durch den Austausch von Daten und Informationen zu interagieren, sondern darüber hinaus auch mit externen Softwaresystemen.

Der Austausch von Daten und Informationen zwischen externen Modellkomponenten des Simulationsmodells und den zu integrierenden Softwaresystemen wurde durch Nachrichtenaustausch über ein für alle Teilnehmer verbindliches Rahmenprotokoll realisiert. Dabei wurde insbesondere berücksichtigt, daß eine externe Modellkomponente mit mehreren externen Softwaresystemen Nachrichten austauschen kann und umgekehrt.

Weiterhin wurde durch die Verwendung des TCP/IP–Protokolls bei der Realisierung der Kommunikation berücksichtigt, daß die zu integrierenden Softwaresysteme sowie das Simulationssystem über ein Mehrrechnersystem mit dynamischer Rechnerzuordnung verteilt sein können. Damit Nachrichten unabhängig von genauen physikalischen Rechner- und

Prozeßidentifikationen adressiert werden können, werden sämtliche Nachrichten an ein Transceiver geschickt, der sie an die entsprechenden Empfänger weiterleitet.

Neben den Erweiterungen für Modellkomponenten werden auch Module (C und PROLOG) für die Erweiterung der zu integrierenden Softwaresysteme zur Verfügung gestellt. Dazu gehören unter anderem Module zum Nachrichtenaustausch, der Vorverarbeitung von Nachrichten sowie zur Experimentdurchführung.

3.2 Experimentdurchführung

Neben dem Austausch von Daten und Informationen zwischen Modellkomponenten und Softwaresystemen muß auch für die Durchführung und Steuerung von Simulationsexperimenten gesorgt werden. Dazu wurden einerseits am Simulationssystem entsprechende Erweiterungen vorgenommen und andererseits ein Protokoll festgelegt, über das beispielsweise Beginn, Ende, Unterbrechung oder Fortsetzung eines Experiments allen angeschlossenen externen Softwaresystemen mitgeteilt werden. Eine Unterbrechung oder das Ende eines Simulationsexperiments kann aber nicht nur durch das Simulationssystem ausgelöst werden, sondern auch durch eines der angeschlossenen externen Softwaresysteme. Auch bei der Experimentdurchführung werden alle Nachrichten an den Transceiver übertragen, der sie dann an die entsprechenden Empfänger weiterleitet.

Neben Verfahren zur Experimentsteuerung beinhaltet die Experimentierumgebung eine Reihe weiterer Vefahren, unter anderem zur zeitlichen Synchronisation, zur Initialisierung eines definierten Anfangszustands sowie zum Wiederaufsetzen auf einen früheren Zustand.

Sollen während eines Simulationsexperiments Daten aufgezeichnet werden, so kann dies im allgemeinen mit Hilfe der vom Simulationssystem dafür zur Verfügung gestellten Verfahren geschehen, wobei unter Umständen eine vorherige Konvertierung von Datentypen erforderlich ist. Es hat sich gezeigt, daß es komplexe Datenstrukturen gibt, deren Werteverlauf man gerne aufzeichnen möchte (z. B. Veränderung der Wissensbasis eines Expertensystems), die sich aber nicht so einfach in die vom Simulationssystem zur Verfügung gestellten Datentypen konvertieren lassen. Um die allzu aufwendige Konvertierung von Daten zu vermeiden, wurde eine Möglichkeit vorgesehen, solche Daten auch extern aufzuzeichnen und einem Simulationsexperiment zuzuordnen.

3.3 Ergebnisaufbereitung und -präsentation

Für die Auswertung von Untersuchungsergebnissen stellt das Simulationssystem dem Anwender eine ganze Reihe von mathematischen Verfahren zur Analyse und Vergleich aufgezeichneter Daten zur Verfügung. Neben statistischen Auswertungsverfahren (z. B. Berechnung von Mittelwerten, Konfidenzintervallen, Momente empirischer Verteilungen) sind hier die Verfahren zur Durchführung von Zeitreihenanalysen (Einschwingphasen, Fourieranalysen, Frequenzspektrum) zu nennen. Um spezielle Anforderungen eines Anwenders nach weiteren Auswertungsverfahren zu berücksichtigen, besteht die Möglichkeit, entweder zusätzliche Verfahren direkt in das Simulationssystem zu integrieren oder aufgezeichnete Daten in Form von einfachen Textdateien zu exportieren und extern auszuwerten.

Die Darstellungsumgebung des Simulationssystems SIMPLEX-II stellt dem Anwender eine Reihe von Verfahren zur graphischen Präsentation von aufgezeichneten oder berechneteten Simulationsergebnissen in Form von Diagrammen (z. B. Kurven-, Balken und Kreisdiagramme) zur Verfügung. Neben der einfachen statischen Darstellung von Zusammenhängen in einem untersuchten System können dynamische Vorgänge durch eine graphische Animation veranschaulicht werden. Dazu können vom Anwender graphische Objekte und Animationslayouts erstellt und Animationen durchgeführt werden.

4 Anwendungsbeispiel

Eines der Arbeitspakete im PEDE-Projekt befaßt sich mit der Erstellung einer Bibliothek von Modellbausteinen sowie kompletten Modellumgebungen, die eine Untersuchung des Verhaltens komplexer Softwaresysteme in möglichst realistischen Szenarien und unter praxisrelevanten Bedingungen gestatten. Eine solche Modellumgebung wurde bereits für das in [Schrödel 92] beschriebene Flexible Fertigungssystem aufgebaut (vgl. Abbildung 3).

In diesem Fertigungssystem werden verschiedene Varianten von Ritzelwellen, Abtriebswellen und Zahnräder für eine einstufige Getriebebaureihe gefertigt und unter Verwendung von Wälzlagern, Paßfedern und Sicherungsringen vormontiert.

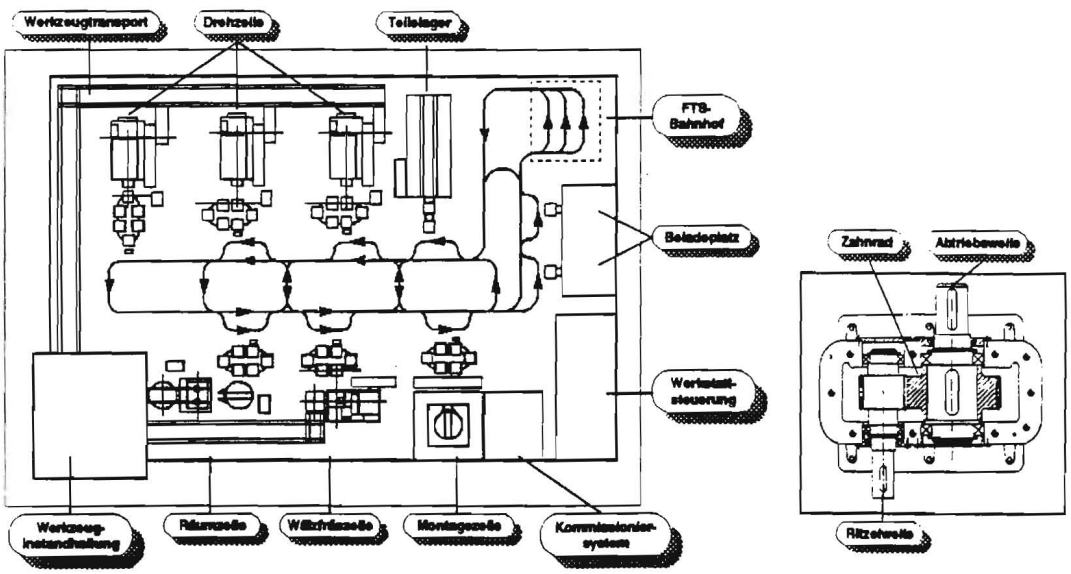


Abbildung 3: Flexibles Fertigungssystem zur Fertigung von Motorgetrieben

Das Flexible Fertigungssystem beinhaltet verschiedene, zum Teil alternativ benutzbare Fertigungs- und Montagezellen. Diese Zellen bestehen ihrerseits aus der eigentlichen Bearbeitungseinheit (z. B. Drehmaschine), einem Werkzeugspeicher, einem Palettenwechselsystem und mit Ausnahme der Montagezelle einem Portalroboter zur Handhabung von Werkstücken und Werkzeugen. Die Versorgung der einzelnen Zellen mit Werkstücken erfolgt durch ein induktiv geführtes Fahrerloses Transportsystem. Die Werkzeugversorgung erfolgt über ein Bandsystem.

Das gesamte Fertigungssystem wird über ein hierarchisch gegliedertes, rechnerintegriertes Werkstattsteuerungssystem überwacht und gesteuert. Dabei wird im wesentlichen zwischen Steuerungen auf Zellenebene (Zellensteuerung) und einer Steuerung auf Leitebene (Werkstattsteuerung) unterschieden. Aufgabe der Zellensteuerungen ist die Steuerung und Überwachung der Abläufe in der zugehörigen Zelle, die Bereitstellung erforderlicher Steuerungsprogramme und die Abwicklung der Kommunikation mit der Werkstattsteuerung. Die Werktstattsteuerung übernimmt die Termin- und Kapazitätsplanung, die Ermittlung von Fertigungsalternativen, die Maschinenbelegungsplanung und die Planung der Transportaufträge für alle Fertigungs- und Montageaufträge. Sie verfügt über einen kurzfristigen Planungshorizont und plant auf Grund der von einem übergeordneten Produktionsplanungs- und -steuerungssystem vorgegebenen Tagesscheiben sowie der von der Arbeitsplanung vorgegebenen Arbeitspläne.

Dem Anwender wird eine Modellbank zur Verfügung gestellt, die unter anderem Modellkomponenten für Fertigungs- und Montagezellen, Teilelager, Transportsysteme sowie Werkstücke und Werkzeuge enthält. Aus diesen Modellkomponenten lassen sich nahezu beliebige Modelle Flexibler Fertigungssysteme aufbauen, Softwaresysteme (z. B. Produktionsplanungs- und Steuerungssysteme, Scheduling-Systeme) einbetten und das Verhalten einzelner Teilsysteme sowie das Verhalten des Gesamtsystems gezielt untersuchen [1].

5 Stand der Implementierung und künftige Arbeiten

Dem Anwender steht derzeit ein erster Prototyp der Experimentierumgebung zur Verfügung, der es erlaubt, Softwaresysteme in ein Simulationsmodell innerhalb des Simulationssystems SIMPLEX–II zu integrieren und Simulationsuntersuchungen daran durchzuführen.

Dabei lag der Schwerpunkt der bisherigen Arbeiten vorwiegend auf der Realisierung von Methoden zum Datenaustausch und Kommunikation zwischen verteilten Softwaresystemen und Modellkomponenten innerhalb eines Simulationsmodells. Der Schwerpunkt künftiger Arbeiten liegt dagegen im Bereich der Experimentdurchführung. Hier gilt es noch eine Reihe von Problemen zu lösen, die überwiegend aus der räumlichen und zeitlichen Verteilung von Simulationsmodell und den darin integrierten Softwaresysteme resultieren.

Für einige der erkannten Probleme sind bereits erste Lösungsansätze bekannt und zum Teil schon realisiert. Ein Beispiel dafür ist die zeitliche Synchronisation zwischen externen Softwaresysteme und einem Simulationsmodell, welche durch ein modifiziertes Broadcast-Protokoll realisiert wurde [Reinema 93]. Bei einigen der gewählten Lösungsansätze hat sich gezeigt, daß sie nur in bestimmten Grenzen und unter ganz bestimmten Voraussetzungen verwendbar sind. Daraus resultiert beispielsweise die Einschränkung, daß der Einsatz der Experimentierumgebung zum jetzigen Zeitpunkt praktisch auf lokale Rechnernetze beschränkt ist, obwohl die externen Softwaresysteme und das Simulationssystem prinzipiell über größere Entfernungen verteilt sein können.

[1] Eine solche Einbettung wurde bereits erfolgreich realisiert. Dabei wurde ein Softwaresystem zur Werkstattsteuerung in das Simulationsmodell des beschriebenen Fertigungsystems integriert und eine gemeinsame Simulation durchgeführt.

Daneben gibt es aber auch noch andere Probleme, bei denen derzeit nicht eindeutig feststeht, inwieweit die in der Literatur vorgeschlagenen Lösungen übertragbar und einsetzbar sind. Ein solches Problem ist das Anhalten eines verteilten Softwaresystems in einem bestimmten Zustand. Bei verteilten Softwaresystemen kann es unter Umständen schwierig sein, konsistente Zustände zu finden, in denen sie angehalten werden können. Es kann vorkommen, daß sich einige Teilsysteme in Zuständen befinden, in denen eine Unterbrechung zu bestimmten Zeitpunkten nicht zulässig ist, da sonst Inkonsistenzen auftreten. Darüber hinaus kann bei einem verteilten Softwaresystem im allgemeinen nicht gewährleistet werden kann, daß alle Teilsysteme gleichzeitig und bei der Durchführung des gleichen Experiments auch immer an der selben Stelle angehalten werden können. Verantwortlich hierfür sind hauptsächlich mögliche Verzögerungen bei der Nachrichten- oder Signalübertragung.

Ein weiteres noch zu lösendes Problem ist das Speichern von Zuständen, auf die zu einem späteren Zeitpunkt wieder aufgesetzt werden soll sowie das Wiederaufsetzen auf einen solchen früheren Zustand. Dies ist nur dann möglich, wenn sich sowohl der entsprechende Zustand des Simulationsmodells als auch der Zustand der angekoppelten externen Softwaresysteme wiederherstellen läßt. Auch hier besteht das Problem, daß ein Zurücksetzen bestimmter Softwaresysteme auf einen beliebigen Zustand nicht zulässig sein kann, da sonst Inkonsistenzen auftreten können. In diesem Zusammenhang muß auch überlegt werden, auf welche Weise ein Rücksetzen beziehungsweise die Initialisierung ganz bestimmter, vom Anwender vorgebbare Anfangszustände realisiert werden kann.

Damit Experimentabläufe besser beschrieben und Experimente leichter durchgeführt werden können, ist eine deklarative Experimentbeschreibungssprache von großem Nutzen. Eine solche Experimentbeschreibungssprache sollte mindestens über sprachliche Mittel zur Formulierung von Bedingungen und Schleifen sowie zur Spezifikation und Auslösung von Ereignissen, die nicht Bestandteil des Verhaltens eines modellierten Systems sind, verfügen.

Neben der Modellumgebung für das im vorangegangenen Abschnitt dargestellte Anwendungsbeispiel steht derzeit noch eine weitere Modellumgebung zur Verfügung. Sie beinhaltet Modelle und Modellkomponenten für Just-In-Time Beschaffungsprozesse in der Automobilindustrie [Reinema 93]. Ziel künftiger Arbeiten wird es auch sein, weitere Modellumgebungen für ausgewählte Anwendungsszenarien zur Verfügung zu stellen.

6 Zusammenfassung

Im PEDE–Projekt wird mit der Experimentierumgebung PEDE–Lab ein neuer Ansatz für das Testen und Evaluieren komplexer Softwaresysteme verfolgt. Die entscheidende Idee bei der Realisierung der Experimentierumgebung war es, ein konventionelles Simulationssystem so zu erweitern, daß sich die zu untersuchenden Softwaresysteme (z. B. VKI-Systeme) in ein Simulationsmodell integrieren lassen und eine gemeinsame Simulation durchgeführt werden kann. Es ist keine zeit- und kostenintensive Modellierung komplexer Softwaresysteme mehr erforderlich. Die zu untersuchenden Softwaresysteme können prinzipiell als ihr eigenes Modell verwendet werden.

Ein weiterer Vorteil ist, daß eine Modellbeschreibung eines realen Systems nicht mehr

ausschließlich mit der zur Verfügung gestellten Modellbeschreibungssprache erfolgen mu
Bestimmte Sachverhalte, die sich mit der Modellbeschreibungssprache SIMPLEX-MD
[SIMPLEX 92] nur schwer oder gar nicht beschreiben lassen, können mit Hilfe andere
Repräsentationssprachen beschrieben werden, die dafür wesentlich besser geeignet sin
Die Softwaresysteme, die aus derartigen Beschreibungen hervorgehen, lassen sich dann mi
den zur Verfügung gestellten Methoden wieder in das Simulationsmodell integrieren.

Durch Nutzung der im Simulationssystem bereits verfügbaren Verfahren, beispielsweis
zur Erzeugung von Zufallszahlen, Ergebnisauswertung und -präsentation, konnte der zeit
liche Aufwand für die Realisierung der Experimentierumgebung reduziert werden. Dies
Verfahren bieten dem Anwender bereits sehr komfortable Möglichkeiten und reduzieren
den Aufwand für das Testen, die Evaluation und den Vergleich verschiedener komplexe
Softwaresysteme ganz erheblich.

Mit der Entwicklung der Experimentierumgebung PEDE-Lab wird ein wichtiger Schrit
zur besseren Unterstützung des Anwenders bei der Untersuchung des Verhaltens komplexe
und verteilter Systeme unternommen. Gerade bei der Entwicklung von VKI-Systemen kann
die Experimentierumgebung PEDE-Lab einen wertvollen Beitrag leisten, wenn es darum
geht, verschiedene alternative Ansätze und Softwaresysteme in standardisierten Testszena
rien miteinander zu vergleichen.

Die Weiterentwicklung von PEDE-Lab wird begleitet durch praxisnahe Einsatzbeispie
le, um einen Aufschluß über eventuell vorhandene Schwächen und Grenzen der Experimen
tierumgebung zu erhalten und geeignete Möglichkeiten zu deren Beseitigung ergreifen zu
können.

Literatur

[Bond 88] Bond, A. H.; Gasser, L.: *Readings in Distributed Artificial Intelligence.* Morgan Kaufmann Publishers, San Mateo, Calif., 1988.

[Brauer 91] Brauer, W.; Hernandez, D. (Hrsg.): *Verteilte künstliche Intelligenz und kooperatives Arbeiten.* Nr. 291 Reihe Informatik Fachberichte Springer, Berlin, 1991.

[Cohen 92] Cohen, P.; Porter, B., Ed.: *Experimental Methods in Artificial Intelligence - Tutorial at the 10th National Conference on Artificial Intelligence*, San Jose, July 1992. AAAI.

[Durfee 92] Durfee, E. H.; Sycara, K. P., Ed.: *Distributed Artificial Intelligence Tools - Tutorial at the 10th National Conference on Artificial Intelligence*, San Jose, July 1992. AAAI.

[Huhns 87] Huhns, M. N., Ed.: *Distributed Artificial Intelligence.* Pitman Publishing, London, 1987.

[McDowell 91] McDowell, C.; Miller, B., Ed.: *ACM/ONR Workshop on Parallel and Distributed Debugging*, Santa Cruz, Calif., February 1991. ACM.

[Reinema 93] Reinema, R.: *PEDE-Lab — Aufbau und Entwicklung einer Experimentierumgebung für Multi-Agenten-Systeme.* Diplomarbeit, Universität Erlangen-Nürnberg, März 1993.

[Schrödel 92] Schrödel, O.: *Flexible Werkstattsteuerung mit objektorientierten Softwarestrukturen.* Dissertation, Universität Erlangen-Nürnberg, Hanser, München, 1992.

[SIMPLEX 92] SIMPLEX, : *SIMPLEX-II Referenzhandbuch.* Universität Erlangen-Nürnberg und Siemens AG, Erlangen, 1992.

[Stoyan 92] Stoyan, H.; Beckstein, C.; Kraetzschmar, G. K.; Lutz, E.: *Erstellung und Ausführung von Plänen in verteilten Systemen.* In: *Antrag an die Deutsche Forschungsgemeinschaft auf Fortsetzung des Sonderforschungsbereichs 182 - Multiprozessor- und Netzwerkkonfigurationen.* Universität Erlangen-Nürnberg, 1992.

Verteiltes, wissensbasiertes Gleichheits-beweisen durch Teamwork

Jörg Denzinger
FB Informatik , Universität Kaiserslautern
6750 Kaiserslautern
E-Mail : denzinge@informatik.uni-kl.de

1. Einleitung

Das automatische Theorembeweisen ist immer noch ein interessantes Gebiet künstlichen Intelligenz. Zum einen benutzen viele Systeme Logik Wissensrepräsentation und benötigen dadurch eine Beweiskomponente. Zum anderen tre beim Theorembeweisen sehr große Suchprobleme auf, deren Lösung auch für and Disziplinen hilfreich sind. Dies trifft auch schon auf das reine Gleichheitsbeweisen das das Anwendungsgebiet unserer Teamwork-Methode ist. Wie viele and Beweisverfahren, so wird auch die Vervollständigung durch eine Menge von Inferenzreg beschrieben (vgl. Abschnitt 2). Ein naiver Verteilungsansatz könnte nun darin beste die Inferenzregelanwendungen auf die vorhandenen Prozessoren zu verteilen.

Dabei ergeben sich allerdings erhebliche Probleme. Zum einen gibt es nicht Inferenzregeln, die neue Gleichungen erzeugen, sondern auch solche, die alte Gleichun verändern oder löschen. Die Auswirkungen dieser Inferenzregeln müssen natürlich so allen Prozessoren mitgeteilt werden, um weitere Inferenzschritte mit den al Gleichungen zu verhindern. Dadurch entsteht aber ein sehr hoher Kommunikationsaufwa Ein zweites, viel größeres Problem besteht darin, daß die entstehenden Suchräume groß sind, daß nur sehr gute Auswahlheuristiken für die Anwendung von Inferenzregeln erfolgreichen Beweisen führen. Eine verteilte Realisierung einer solchen Heuristik aufwendig und wieder sehr kommunikationsintensiv, da in der Regel jede neu erzeu Gleichung zu Änderungen in der Reihenfolge der nächsten Inferenzen führt. Deshalb eine solche Vorgehensweise nicht sinnvoll.

Betrachtet man die bekannten Methoden der verteilten KI, wie z. B. Blackboards ([Ha85]) oder verhandlungsbasierte Methoden (vgl. [Sm81]), so stellt man fest, daß vi Methoden darauf basieren, daß man ein Problem zerlegen kann. Die entstehen Teilprobleme bilden dann natürliche Ansatzpunkte für die Verteilung. Beim automatisch Theorembeweisen ist - außer der unpraktikablen Zerlegung des Suchraums - eine Zerleg des Beweisproblems a priori sehr selten möglich. Die bekannten Heuristiken sind in Regel Vermutungen, welche Beweisschritte für einen Beweis wichtig sind. Sie stellen a

keine Zerlegung des Problems dar. In dieser Arbeit stellen wir mit der Teamwork-Methode (vgl. [De91], [De93]) ein Verfahren vor, das diese Probleme löst.

Nach einer kurzen Beschreibung des Gleichheitsbeweisens durch Vervollständigung in Abschnitt 2 werden wir in Abschnitt 3 die Teamwork-Methode beschreiben und ihre Anwendung auf die Vervollständigung skizzieren. In Abschnitt 4 werden wir unsere Erfahrungen mit dem DISCOUNT-System (vgl. [DP92]), einer Implementierung der verteilten Vervollständigung, anhand einiger Beispiele schildern. Schließlich werden wir in Abschnitt 5 einige Ausblicke auf Erweiterungen der Teamwork-Methode geben.

2. Gleichheitsbeweisen durch Vervollständigung

Unter Gleichheitsbeweisen versteht man die folgende Aufgabe :

Gegeben : Eine Menge E von Gleichungen, $E = \{s_i = t_i \mid i = 1,\ldots,n\}$, und ein Ziel $u = v$.

Frage : Ist $u = v$ eine Folgerung aus E, d.h. gilt $u =_E v$?

Als bestes Verfahren zur Lösung dieser Aufgabe hat sich die Vervollständigung nach Knuth und Bendix ([KB70]), verbessert zur Vervollständigung ohne Abbruch (vgl. [BDP89]), erwiesen. In [BDP89] wird die Vervollständigung durch ein Inferenzregelsystem beschrieben, das viele algorithmische Umsetzungen zuläßt. Die grundlegenden Operationen sind das Richten von Gleichungen, das Bilden kritischer Paare und die Reduktion von Termen.

Gleichungen, deren Terme mit einer Reduktionsordnung $>$ (vgl. [De79]) vergleichbar sind, werden zu Regeln "größerer Term $\rightarrow$ kleinerer Term" gerichtet. Diese Regeln werden benutzt, um andere Terme zu reduzieren. Ist t ein Term, $l \Rightarrow r$ eine Regel, σ eine Substitution mit $\sigma(l) \equiv t/p$ für eine Stelle p in t, so kann t zu $t' \equiv t[p \leftarrow \sigma(r)]$ reduziert werden, indem in t an der Stelle p $\sigma(l)$ durch $\sigma(r)$ ersetzt wird. Gibt es in einem Regelsystem keine Regel, die einen Term t reduzieren kann, so heißt t irreduzibel. Kann man t (eventuell in mehreren Schritten) zu t' reduzieren und ist t' irreduzibel, so heißt t' Normalform zu t. Eine Menge von Regeln und Gleichungen, bei denen alle Terme jeder Regel oder Gleichung (außer mit der eigenen Regel) irreduzibel sind, heißt interreduziert.

Neue Gleichungen werden durch das Bilden kritischer Paare erzeugt. Sind $l = r$ und $s = t$ Gleichungen (oder Regeln), p eine Stelle in l und gibt es einen mgu σ zu l/p und s, dann ist $\langle \sigma(r) , \sigma(l)[p \leftarrow \sigma(t)] \rangle$ ein kritisches Paar zu den beiden Gleichungen, falls $\sigma(r) \not> \sigma(l)$ und $\sigma(l)[p \leftarrow \sigma(t)] \not> \sigma(l)$ gilt.

Die von uns benutzte algorithmische Umsetzung der Inferenzregeln der Vervollständigung arbeitet wie folgt :

Eingabe sind E, $u = v$ und $>$. Die Menge CP ergibt sich initial aus E. Dann startet man mit leerem E und R. Solange es kritische Paare in CP gibt und die Normalformen von u

und v nicht identisch sind, wählt man ein s = t aus CP und bildet die Normalformen
$\hat{t}$ zu s und t bezüglich R. Falls $\hat{s}$ und $\hat{t}$ nicht identisch sind, vergleicht man $\hat{s}$ und
mit >. Sind die Terme vergleichbar, so erhält man eine Regel l → r. Mit dieser Re
werden zunächst R und E interreduziert, dann wird sie zu R hinzugefügt und
kritischen Paare zwischen l → r und R und E in CP aufgenommen. Sind $\hat{s}$ und
unvergleichbar, so wird $\hat{s}$ = $\hat{t}$ in E aufgenommen und es werden alle kritischen Pa
zwischen $\hat{s}$ = $\hat{t}$ und R und E gebildet und zu CP addiert.

Sind die Normalformen zu u und v irgendwann identisch, so gilt $u =_E v$. Ist CP leer
sind die Normalformen von u und v ungleich, so gilt nicht $u =_E v$.

Man beachte, daß der Reduktion eine wichtige Rolle zukommt. Bevor Regeln o
Gleichungen aufgenommen werden, werden ihre Normalformen gebildet. Außerdem w
mit jeder neuen Regel das bestehende System interreduziert. Dadurch werden R und
möglichst klein und kompakt gehalten. Andererseits werden R und E fast stän
verändert. Im Hinblick auf verteilte Systeme bereitet dies Schwierigkeiten, wenn R und
in einem allen Prozessoren gemeinsamen Speicher gehalten werden sollen, da Änderun
eines Prozessors zum Sperren aller Zugriffe der anderen Prozessoren führen müssen.

Wesentlich für die Effizienz des Algorithmus ist die Auswahl des nächsten
bearbeitenden kritischen Paares s = t. Es gibt viele unterschiedliche Strategien u
Heuristiken, die alle für manche Beispiele gut und für andere Beispiele nur sehr schle
geeignet sind. Diese Strategien und Heuristiken sind so unterschiedlich, daß sequentie
Versuche, sie in eine Heuristik zu kombinieren, fehlschlagen, d.h. die entstehend
Heuristiken sind in der Regel schlechter als die ursprünglichen. Dieses Problem w
durch unsere Teamwork-Methode erfolgreich gelöst.

3. Die Teamwork-Methode am Beispiel der Vervollständigung

Wie schon eingangs erwähnt, ergeben sich bei der Verteilung der Vervollständigung vie
Probleme. Versucht man bekannte Verfahren der verteilten KI anzuwenden, so fehlt d
Vervollständigung entweder die Struktur, die eine Zerlegung des Problems ermöglic
oder die Verfahren erlauben es nicht, das vorhandene Wissen über die Vervollständigu
wie z.B. Auswahlheuristiken für kritische Paare, sinnvoll zu integrieren. Ein Beispiel f
die zweite Art von Verfahren ist in [KH81] zu finden. Hier ist besonders die Prämis
daß kein Wissen verloren gehen darf, sehr schädlich, da jede überflüssige Regel o
Gleichung den Suchraum bei der Vervollständigung stark aufbläht. Um ein Vergessen v
Wissen zu ermöglichen, wird eine stärkere Kontrolle benötigt.

Mit der Teamwork-Methode haben wir versucht, einige gegensätzliche Aspekte verteil
Systeme zu kombinieren, wie zum Beispiel Wettbewerb mit Kooperation o
eigenständiges, unabhängiges Arbeiten mit zentraler Kontrolle. Wie wir in Abschnitt
unter Beweis stellen werden, ist dieser Versuch gelungen.

Vorbild unserer Teamwork-Methode ist das Projekt-Team, wie es viele Firmen zusammenstellen, wenn innovativ Probleme gelöst werden sollen. In ihm werden genau die geschilderten Gegensätze vereinigt. Wir haben eine Variante des Projekt-Teams gewählt, die die wichtigsten Eigenschaften erhält und gleichzeitig gut auf einem Netz von Computern implementiert werden kann.

Unser Team besteht aus Experten (Spezialisten), Gutachtern und einem Leiter. Der **Leiter** ist verantwortlich für die Zusammensetzung des Teams und für die Kooperation der einzelnen Komponenten. In der Regel sind mehr Experten vorhanden, als dem Team Computer zur Verfügung stehen. Deswegen ist es wichtig, daß die für ein Problem am besten geeigneten Experten diese Computer belegen. Diese Auswahl wird vom Leiter getroffen. Außer auf Hintergrundinformationen über die Experten verläßt sich der Leiter im wesentlichen auf Beurteilungen der Arbeit der Experten durch die Gutachter.

Der Leiter stellt allen ausgewählten Experten die gesamte Problemspezifikation zur Verfügung und gibt einen Zeitpunkt vor, an dem er über die Arbeit aller Experten unterrichtet werden möchte. Zu diesem Zeitpunkt findet ein sogenanntes Teamtreffen statt. Während eines Teamtreffens nimmt der Leiter die Berichte der Gutachter entgegen, bestimmt auf Grund der Berichte Experten für die vorhandenen Computer und nutzt die besten Resultate der Experten, die von den Gutachtern ebenfalls bestimmt werden, um eine genauere Problemspezifikation zu erstellen. Diese bildet den Ausgangspunkt für eine neue Arbeitsperiode. Somit ist der Leiter für die Strategie des Gesamtteams zuständig.

Die **Experten** haben die Aufgabe, die Lösung des gegebenen Problems zu finden oder zumindest voranzutreiben. Ihnen ist die Gesamtaufgabe bekannt und sie gehen diese Aufgabe mit ihren eigenen, individuellen Lösungsansätzen und -methoden an. Sie sind nur zwei Einschränkungen unterworfen : ihre Resultate müssen zu jedem Zeitpunkt den Gutachtern zur Verfügung gestellt werden können und sie müssen in einer Form abgefaßt sein, die von allen anderen Komponenten des Teams, auch von anderen Experten, verstanden wird. Während der Leiter strategische Entscheidungen mit Hilfe entsprechenden Wissens trifft, liegen die Entscheidungen der Experten auf der taktischen Ebene.

Die **Gutachter** stellen die Verbindung zwischen den Experten und dem Leiter dar. Jedem Experten wird ein eigener Gutachter zugewiesen. Sie haben kurz vor dem Teamtreffen zwei Aufgaben zu erfüllen. Zum einen müssen sie die Gesamtleistung ihres Experten in der letzten Arbeitsperiode beurteilen und zum anderen herausragende Resultate des Experten bestimmen. Beide Ergebnisse melden sie dem Leiter. Liegen keine herausragenden Resultate vor, werden auch keine weitergemeldet. Zur Durchführung ihrer Aufgaben verwenden die Gutachter Beurteilungswissen.

Unser Team ist in dieser Form sehr gut geeignet, Probleme zu lösen, deren Struktur kein a-priori-Erkennen von Teilproblemen ermöglicht, wie es beim Theorembeweisen meistens

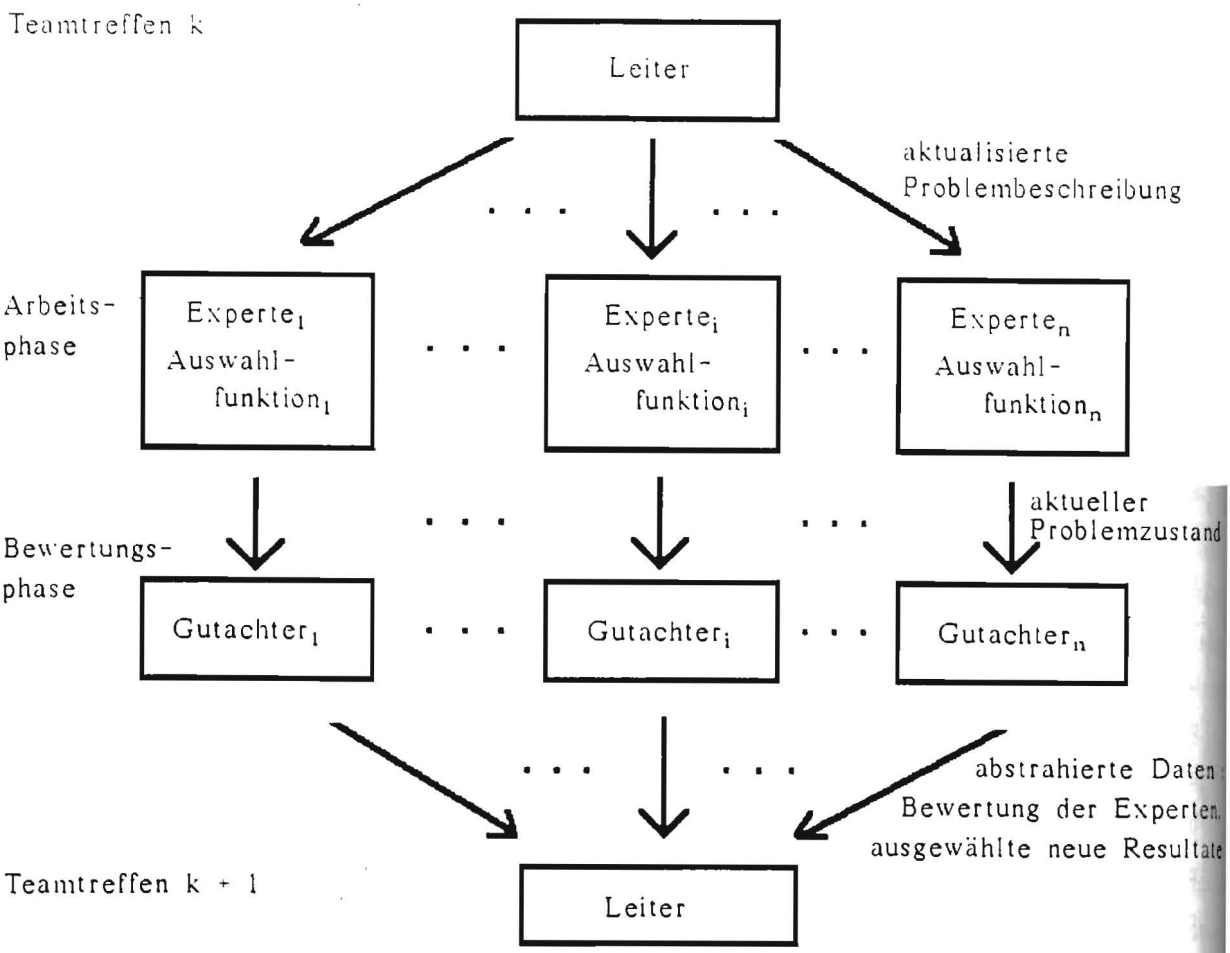


Bild 1 : Arbeitszyklus eines Teams zwischen 2 Teamtreffen

der Fall ist. Potentiell erfolgreiche Lösungsstrategien werden ausprobiert, ihre Erfolg beurteilt und verglichen und sinnvolle Teilresultate ausgewählt, bzw. erwiesenermaß erfolgreiche Lösungsstrategien weiter verfolgt. Das gleiche Modell läßt sich, mit klein Erweiterungen, auch auf Probleme anwenden, bei denen es - unter anderem - au erkennbare Teilprobleme gibt.

Stattet man den Leiter mit Möglichkeiten zur Erkennung von unabhängigen Teilproblem aus, so kann er spezielle Lösungsmethoden auf solche Teilprobleme ansetzen. Die Lösungsmethoden werden von sogenannten **Spezialisten** angewendet. Spezialisten erhalte im Gegensatz zu Experten, nur die Beschreibung eines Teilproblems, das sie lösen solle Außerdem melden sie nur die Lösung dieses Teilproblems direkt an den Leiter, oh Gutachter. Diese Meldung erfolgt allerdings auch nur bei Teamtreffen. Ist zu eine Teamtreffen noch keine Lösung gefunden, so arbeitet der Spezialist bis zum nächste Treffen weiter. Man beachte, daß der Einsatz eines Spezialisten nicht nur zu Anfang de Bearbeitung eines Problems erfolgen kann, sondern auch als Reaktion auf neu erziel Resultate. In Bild 1 ist ein Arbeitszyklus eines Teams graphisch dargestellt.

Wir werden nun am Beispiel der Vervollständigung zeigen, wie konkrete Realisierungen der Komponenten eines Teams aussehen und wie die Teamwork-Methode effizient implementiert werden kann.

Experten :

Die Experten führen alle den in Abschnitt 2 vorgestellten Algorithmus zur Vervollständigung aus. Allerdings können sie immer vor der Auswahl eines neuen kritischen Paares aus CP unterbrochen werden, um Teamtreffen zu ermöglichen. Unterschiedliche Experten werden durch unterschiedliche Auswahlheuristiken für kritische Paare erreicht.

Wie schon erwähnt, gibt es eine große Anzahl an Auswahlheuristiken für kritische Paare. In unserer Implementierung sind zum Beispiel unter anderem folgende, schon länger bekannte, Heuristiken enthalten (es wird ein Gewicht bestimmt und jeweils das kleinste kritische Paare ausgewählt):

Add-Weight : Summe der Anzahl der Symbole in beiden Termen des kritischen Paares.

Max-Weight : Maximum der Anzahl der Symbole pro Term des Paares.

GT-Weight : Anzahl der Symbole im bezüglich der Reduktionsordnung größeren Term des Paares.

Prefer-Rule : Eine Variante von Add-Weight. Allerdings werden kritische Paare, deren Terme (noch) nicht vergleichbar sind, zurückgestellt. Dadurch bleibt die Menge E leer.

Die Teamwork-Methode erlaubte zusätzlich noch die Entwicklung weiterer Heuristiken, die sich stärker auf bestimmte Teile der Problemstellung konzentrieren. Für eine sequentielle Vervollständigung sind solche Heuristiken nicht sinnvoll, da sie nur bei sehr wenigen Beispielen zum Erfolg führen. Im Team verrichten sie allerdings substantielle Arbeit. Solche Heuristiken sind zum Beispiel :

Polynom-Weight : Jedem Funktionssymbol wird ein Polynom in soviel Unbestimmten, wie die Funktion Argumente hat, zugewiesen. Das Gewicht eines Termes ergibt sich durch Auswertung der Polynome, wobei Variablen ein konstanter Wert zugewiesen wird.

Forced-Divergence : Es werden solche kritischen Paare ausgewählt, in denen viele geschachtelte Wiederholungen von Teiltermen vorkommen.

Goal-Match : In beiden Termen des Zieles wird ein Teilterm gesucht, den eine Seite des kritischen Paares matcht. Das Gewicht des Paares ist die Anzahl aller Symbole des Zieles, die nicht in den gefundenen Teiltermen liegen.

Goal-Sim : In beiden Termen des kritischen Paares wird ein Teilterm gesucht, eine Seite des Zieles matcht. Das Gewicht des Paares ist die Anzahl Symbole des Paares, die nicht in den gefundenen Teiltermen liegen.

Die Heuristik, bzw. der so entstandene Experte, Polynom-Weight erlaubt es, sich kritische Paare zu konzentrieren, die nur bestimmte Funktionssymbole enthalten (du entsprechende Wahl der Polynome). Verwendet man in einem Team mehr Polynom-Weight Experten, die sich auf verschiedene Mengen von Funktionssymbo konzentrieren, so erreicht man eine Aufteilung des Suchraums. Während eine solc Vorgehensweise besonders in der Angangsphase der Lösung eines Beweisproblems sinnv ist, sind Goal-Match und Goal-Sim darauf spezialisiert, Beweise zu beenden.

Goal-Match bevorzugt solche kritischen Paare, die als Regeln das Beweisziel reduzier werden. Goal-Sim wählt kritische Paare, in denen das Beweisziel als Teilter vorkommt. Die Hoffnung ist, das durch erneute Kritische-Paar-Bildung kritische Paa entstehen, in denen der Zielanteil immer größer wird. In der Regel sind zu Beginn d Beweissuche keine entsprechenden Paare vorhanden, so daß Goal-Match und Goal-Sim a die Hilfe anderer Experten angewiesen sind, die vom Team auch zur Verfügung geste werden.

Gutachter :

Das Problem bei der Realisierung der Gutachter besteht darin, Kritierien zu entwickel mit denen der Erfolg eines Experten und die Güte von Regeln und Gleichunge festgestellt werden können. Will man Beweise in einer schon untersuchte Anwendungsdomäne führen, ist es einfach, solche Kriterien zu finden. Aber sehr oft ha man kaum Wissen über die Domäne eines Beispieles. Für diesen Fall haben w Gutachter entwickelt, die statistische Kriterien benutzen. Insbesondere d Reduktionsoperation sorgt bei der Vervollständigung für sehr brauchbare Daten. Unser statistischen Gutachter benutzen als Bewertungskriterium für Experten eine gewichte Summe aus folgenden Zahlen :

- Gesamtanzahl der Regeln, Gleichungen und kritischen Paare
- Anzahl der neuen Regeln, Gleichungen und kritischen Paare dieser Arbeitsperiode
- Anzahl der gelöschten Regeln und Gleichungen
- Anzahl der Reduktionen von Regeln und Gleichungen
- Anzahl der Reduktionen des Zieles
- Gesamtanzahl aller durchgeführten Reduktionen

Man beachte, daß es, wie auch bei den Experten, sinnvoll ist, je nach Phase de Beweises, die Gutachter auszutauschen. Während am Anfang eines Beweisversuches d Menge der Regeln wachsen muß und auch viele kritische Paare vorhanden sein sollte

sollten in der Endphase solche Experten besser bewertet werden, deren Regelmengen kaum wachsen oder sogar schrumpfen.

Gute Regeln und Gleichungen können ebenfalls durch statistische Kriterien ermittelt werden. Wieder ist eine gewichtete Summe der schon aufgeführten Zahlen, jetzt aber nur für die zu beurteilende Regel oder Gleichung, Grundlage der Bewertung unserer Gutachter. Dabei werden nur solche Regeln und Gleichungen an den Leiter gemeldet, die besser als eine vorgegebene Beurteilung sind. Außerdem ist eine Höchstanzahl n an zu übermittelnden Regeln und Gleichungen vorgegeben, die nicht überschritten wird, um den Kommunikationsaufwand zwischen Gutachtern und dem Leiter klein zu halten.

Man beachte, daß die nicht ausgewählten Regeln und Gleichungen schlechter Experten vergessen werden, falls sie nicht von anderen Experten ebenfalls erzeugt wurden.

Der Leiter :

Neben der Auswahl der Experten und Gutachter ist die wesentliche Aufgabe des Leiters das Erstellen einer neuen Problembeschreibung aus den Resultaten der Experten bei jedem Teamtreffen. Wichtig dabei ist, daß die neue Problembeschreibung immer noch das alte Problem (hoffentlich vereinfacht) enthält.

Für unsere verteilte Vervollständigung haben wir die folgende Lösung dieser Aufgabe gewählt. Beim Teamtreffen nimmt der Leiter die aktuellen Mengen R, E und CP des besten Experten der letzten Arbeitsperiode als Ausgangspunkt. Zu diesem System addiert er die ausgewählten Resultate der anderen Experten. Sowohl die Addition von Regeln als auch von Gleichungen erfolgt dabei wie die Bearbeitung von kritischen Paaren aus Abschnitt 2. Die so entstandenen Mengen R, E und CP werden dann an alle Experten geschickt.

Das Zusammenstellen eines neuen Teams nach einer Arbeitsperiode läßt sich in zwei Teilfragen zerlegen : Welche alten Experten sollen ersetzt werden und welche neuen Experten sollen ihre Plätze, i.e. Prozessoren, einnehmen ? Zum Finden der Experten, die ausgetauscht werden sollen, gehen wir nach zwei Grundsätzen vor. Zum einen sollte periodisch ein neuer Experte ins Team aufgenommen werden, um nicht ständig in eingefahrenen Bahnen zu bleiben. Für diesen Experten hat der am schlechtesten bewertete Experte der letzten Periode zu weichen. Außerdem führt der Leiter Buch über alle Bewertungen eines Experten. Ist ein Experte die beiden letzten Arbeitsphasen deutlich schlechter bewertet worden als der jeweilige beste Experte, so kann er im Moment nicht viel zur Lösung beitragen und wird ersetzt. Dadurch sind schnelle Reaktionen des Leiters möglich.

Bei der Auswahl neuer Experten geht der Leiter ebenfalls nach zwei Gesichtspunkten vor. Zum einen bevorzugt er Experten, von denen er weiß, daß sie gut zum am besten

bewerteten Experten passen. Zum anderen bemüht er sich, allen Experten die Gelegenh zu geben, sich an der Lösung des Problems zu versuchen.

Die Implementierung :

Wir haben eine verteilte Vervollständigung, basierend auf der Teamwork-Methode, DISCOUNT-System, auf einem Cluster von Sun-Workstations in C unter U implementiert (vgl. [DP92]). Dabei haben wir großen Wert auf einen klei Kommunikationsaufwand zwischen den Rechnern gelegt. Betrachtet man die Aufgaben Aktivitäten der einzelnen Komponenten eines auf der Teamwork-Methode basieren Systems, so stellt man fest, daß man alle drei Arten von Komponenten durch ei Prozeß mit unterschiedlichen Arbeitsmodi realisieren kann. Insbesondere ist der Le immer nur bei den Teamtreffen aktiv und die Gutachter können so von den Exper unterstützt werden, daß sie nur kurz vor den Teamtreffen aktiv sein müssen. Sonst immer nur die Experten aktiv. Es ergibt sich folgender Ablauf auf einem Prozessor :

Kurz vor dem Teamtreffen wechselt der Prozessor vom Experten- in Gutachtermodus. Ist die Begutachtung abgeschlossen, gibt es zwei Möglichkeiten. Lief dem Prozessor am Ende des letzten Teamtreffens der Leiter, so wechselt er in Leitermodus. Ansonsten bleibt er im Gutachtermodus. Der Prozessor im Leitermo bestimmt den besten Experten, dessen Prozessor nun in den Leitermodus überg während der alte Leiterprozessor zurück in den Gutachtermodus wechselt. Der n Leiter nimmt nun die ausgewählten Resultate der Gutachter entgegen. Daraufhin wech alle Prozessoren, bis auf den Leiter, in den Expertenmodus. Der Leiter versendet, n Integration der empfangenen Resultate, sein System an alle anderen Prozessoren wechselt danach wieder in den Expertenmodus und eine neue Arbeitsphase beginnt.

Durch diese Implementierung müssen nur noch die Berichte der Gutachter, die sehr kl sind, und, per Broadcast, das neue Startsystem verschickt werden. Da die Prozessoren Teamtreffen dazu benutzen, um die Daten kompakt im Speicher abzulegen und überha ihre Speicherungsstruktur besser zu organisieren, fällt der entstandene Kommunikatio aufwand praktisch nicht ins Gewicht. Dadurch konnten die guten Laufzeitresultate nächsten Abschnittes erzielt werden.

4. Erfahrungen mit dem DISCOUNT-System

Der Erfolg jedes Theorembeweisers wird an zwei Dingen gemessen, den Problemen, zu lösen er im Stande ist, und der Geschwindigkeit, in der er diese Probleme lösen k Besonders die Geschwindigkeit oder Effizienz erlaubt uns eine gute Bewertung des Erf oder Mißerfolgs unserer Teamwork-Methode, da wir ja den Vergleich zu sequentiellen Version unserer Vervollständigung ziehen können. Bei der Bewertung er

sich allerdings das Problem, daß uns viele Heuristiken für die sequentielle Version zur Verfügung stehen. Mögliche Vergleichswerte könnten die Zeit der besten Heuristik überhaupt, der besten Heuristik, die auch als Experte im erfolgreichen Team benutzt wurde, oder verschiedene Durchschnittswerte sein.

Natürlich stellen schnellere Laufzeiten gegenüber der besten vorhandenen Heuristik einen sehr schwer zu erzielenden Erfolg dar. Trotzdem zeigt die folgende Tabelle, daß wir deutlich bessere Laufzeiten (angegeben in Sekunden) erzielen, selbst wenn wir berücksichtigen, daß wir zwei Rechner benutzen und deshalb von vorn herein einen Speed-up von mindestens 2 erwarten.

Beispiel	Team	1.Experte	2. Experte	bester Experte
a11	15.044	40.908	95.407	29.441
a12	13.518	23730.000	81.383	32.822
a13	174.189	--	--	--
ring3	307.692	5153.000	--	3019.000
hirsh9-4	1.690	24.927	42.862	4.718
hirsh9-5	1.739	17.696	60.548	4.178

Ebenfalls bemerkenswert ist, daß es uns mit Beispiel *a13* gelang, im Team ein Beispiel zu lösen, das wir mit keiner unserer Heuristiken sequentiell lösen konnten. Somit konnten wir in beiden Bewertungskriterien für Theorembeweiser Erfolge erzielen.

Unsere Beispiele stellen einen Querschnitt durch verschiedene Anwendungsdomänen dar. Die Beispiele *a11* bis *a13* beweisen, eine Gleichungsaxiomatisierung der Aussagenlogik benutzend, Tautologien. Die Beispiele *hirsh9-4* und *hirsh9-5* stammen aus dem Gebiet der Robbins-Algebren und das Beispiel *ring3* beweist, das ein Ring, in dem $x^3=x$ gilt, kommutativ ist. Für nähere Angaben zu diesen Beispielen, die alle von verschiedenen Autoren als schwierig eingestuft werden, sei auf [AD93] und [De93] verwiesen.

Wir wollen anhand eines Beispiels, *ring3*, erörtern, wie die erzielten Effizienzsteigerungen durch die Teamwork-Methode erreicht werden. Wie schon erwähnt, lautet die Beweisaufgabe, zu zeigen, daß ein Ring (Operatoren f, j, i inverses Element bzgl. j, 0 neutrales Element zu j), in dem f(x,f(x,x)) = x gilt, in f kommutativ ist. Das von uns verwendete Team bestand aus den Experten Prefer-Rule und Add-Weight mit statistischen Gutachtern, die maximal 5 Regeln und 2 Gleichungen auswählen durften. Die Dauer der Arbeitsperioden wurde linear gesteigert. Die Lösung wurde in der 5. Arbeitsperiode gefunden. Wir betrachten die Arbeitsperioden im einzelnen.

1. Arbeitsperiode :

bester Experte : Prefer-Rule

4 Regeln und 2 Gleichungen von Add-Weight ausgewählt.

Benötigt wurde eine dieser Gleichungen (j(x,j(y,z)) = j(y,j(z,x))) zum Beweis

2. Arbeitsperiode :

bester Experte : Prefer-Rule

5 Regeln und 1 Gleichung von Add-Weight ausgewählt.

Keines dieser ausgewählten Resultate wird zum Beweis benötigt.

3. Arbeitsperiode :

bester Experte : Prefer-Rule

2 Gleichungen von Add-Weight ausgewählt.

Beide werden nicht zum Beweis benötigt.

4. Arbeitsperiode :

bester Experte : Prefer-Rule

2 Gleichungen von Add-Weight ausgewählt.

Beide werden nicht zum Beweis benötigt.

5. Arbeitsperiode :

Die zu beweisende Gleichung wird von Add-Weight erzeugt.

Dieses Beispiel demonstriert beide Arten der Kooperation zwischen den Experten, die die Effizienzsteigerungen verantwortlich sind. Zum einen können ausgewählte Resulta von anderen Experten, die von dem besten Experten (noch) nicht hergeleitet wurden, ab für den Beweis nötig sind, natürlich die Beweisfindung stark beschleunigen, wie im Fa der ausgewählten Gleichung nach der 1. Arbeitsperiode. Zum anderen kann ein Exper ausgehend vom Gesamtsystem des besten Experten des letzten Teamtreffens, den Bewe weiter vorantreiben oder sogar, wie hier, beenden. Dieser Wechsel der Heuristik ka auch mehrmals erfolgen (so zum Beispiel bei *al3*). Wie das Beispiel *ring3* zeigt, könn auch beide Arten zusammenwirken.

Unsere Erfahrungen haben gezeigt, daß die Teamwork-Methode zu deutlich besser Ergebnissen führt als unsere Heuristiken in sequentiellen Läufen. Da unser sequentiell Beweiser durchaus in Konkurrenz mit allen anderen bekannten Gleichheitsbeweisern trete kann - z.B. ist er der einzige uns bekannte Beweiser, der das Beispiel *ring3* ohne d Verwendung von AC-Unifikation lösen kann - sind die Effizienzgewinne nicht durc Schwächen im Grundbeweiser entstanden.

Der Grund für die Gewinne liegt vielmehr im Auftreten **synergetischer Effekte** durch d Kooperation und den Wettbewerb der Experten. Damit wird, zumindest für das Gebiet de Deduktionssysteme, der Anspruch der verteilten KI, eine neue Dimension de Modellierung mit entsprechend besseren Resultaten zu eröffnen, bestätigt.

5. Ausblicke

Mit der Teamwork-Methode haben wir ein Verteilungsverfahren vorgestellt, das es erlaubt, Probleme verteilt zu lösen, deren Struktur keine Verteilung nahelegt. Am Beispiel des Gleichheitsbeweisens durch Vervollständigung haben wir gezeigt, daß die Teamwork-Methode große Effizienzgewinne erlaubt, schon beim Verwenden von nur zwei Prozessoren. Eine Version unseres Systems, das durch Broadcasting die Verwendung von mehr als zwei Prozessoren erlaubt, befindet sich in der Erprobung.

In Kapitel 3 haben wir mit den Spezialisten eine Möglichkeit angegeben, wie a priori identifizierbare Teilprobleme durch die Teamwork-Methode behandelt werden können. Solche Teilprobleme sind beim Theorembeweisen aber erst dann zu finden (mit entsprechenden Lösungsverfahren für die Spezialisten), wenn man feste Anwendungsdomänen betrachtet. Eine solche feste Anwendungsdomäne mir herausfordernden Beispielen, die verbandsgeordneten Gruppen, wird von uns zur Zeit auf ihre Eignung untersucht.

Um die Eignung der Teamwork-Methode als generelles Verfahren zu unterstreichen, werden wir sie zunächst auf weitere Theorembeweisverfahren anwenden. Später sind auch weitere Anwendungen, z.B. im Gebiet Expertensysteme, denkbar.

Literaturverzeichnis

[AD93] Avenhaus, J. ; Denzinger, J. : Distributing equational theorem proving, erscheint in Proc. RTA'93, Montreal, 1993.

[BDP89] Bachmair, L. ; Dershowitz, N. ; Plaisted, D.A. : Completion without Failure, Coll. on the Resolution of Equations in Algebraic Structures, Austin (1987), Academic Press, 1989.

[De79] Dershowitz, N. : Orderings for Term-rewriting Systems, Theoretical Computer Science 17, 1979, pp. 279-301.

[De91] Denzinger, J. : Distributed Knowledge-based Deduction using the Team Work Method, SEKI-Report SR-91-12, Universität Kaiserslautern, 1991.

[De93] Denzinger, J. : Teamwork : Eine Methode zum Entwurf verteilter, wissensbasierter Theorembeweiser, eingereichte Dissertation, Universität Kaiserslautern, 1993.

[DP92] Denzinger, J. ; Pitz, W. : Das DISCOUNT-System : Benutzerhandbuch, SEKI-Working-Paper SWP-92-16, Universität Kaiserslautern, 1992.

[Ha85] Hayes-Roth, B. : A Blackboard Architecture for Control, AI 26(3), 1985, pp. 251-321.

[KB70] Knuth, D.E. ; Bendix, P.B. : Simple Word Problems in Universal Algebra, Computational Algebra, J. Leach, Pergamon Press, 1970, pp. 263-297.

[KH81] Kornfeld, W.A. ; Hewitt, C. : The Scientific Community Metaphor, IEEE Transactions on Systems, Man, and Cybernetics 11, 1981, pp. 24-33.

[Sm81] Smith, R.G. : A Framework for Distributed Problem Solving, UMI Research Press, Ann Arbor, 1981.

Ein verbandsbasierter Ansatz zur Koordinierung divergierender Ziele

Barbara Messing
Institut für Angewandte Informatik und Formale Beschreibungsverfahren
Universität Karlsruhe
Postfach 6980
7500 Karlsruhe 1
E-mail: messing@aifb.uni-karlsruhe.de

Gründungsworkshop der FG 1.1.6 VKI, 29./30. April 1993

Zusammenfassung

Eine der zentralen Fragen der Verteilten Künstlichen Intelligenz ist: Wie können widersprüchliche Standpunkte und Zielkonflikte zwischen verschiedenen Agenten erkannt und ausgesöhnt werden?In diesem Beitrag wird ein Ansatz vorgestellt. Voten verschiedener Agenten als Wahrheitswerte einer mehrwertigen Logik aufzufassen, die neben den üblichen logischen Verknüpfungen zwei weitere Operatoren anbietet. Die algebraischen Strukturen dafür sind Bilattices - nach M. Ginsberg -, die einen Rahmen für mehrwertige, nichtmonotone und Defaultlogiken bilden. Als Formalismen zur Behandlung unvollständiger und widersprüchlicher Informationen bieten sie sich aber auch an, um divergierende, verteilte Daten zu koordinieren.

1 Einleitung

1.1 Problemstellung und Lösungsansätze

Eine der zentralen Fragen der Verteilten Künstlichen Intelligenz ist: Wie können widersprüchliche Standpunkte und Zielkonflikte zwischen verschiedenen Agenten erkannt und ausgesöhnt werden? Wie können kohärente Entscheidungen getroffen werden, die lokale und globale Effekte in Einklang bringen?

Um diese Probleme zu lösen, müssen Mechanismen entwickelt werden, um Präferenzen, Zielvorstellungen oder Antworten einzelner Agenten auszuwerten. Hier ist zu unterscheiden, ob und in welchem Maße die Agenten durch einen Supervisor gesteuert werden.

Soll aus den Zielvorstellungen einer Gesellschaft von Agenten eine gemeinsame Strategie entwickelt werden, ergeben sich enge Berührungspunkte zur Entscheidungstheorie, insbesondere zur Wahlverhaltensforschung.

Ephrati und Rosenschein ([EpR91], [EpR92a]) wenden Ergebnisse aus der Entscheidungstheorie auf Multi-Agentensysteme an, zum Beispiel die Clarkesche Steuer: Agenten verteilen Gewichtungsfaktoren auf verschiedenen Ziele; bei der Gesamtentscheidung wird derjenige, ohne den das Ergebnis anders ausgefallen wäre, mit einer Steuer bestraft.

Ebenfalls aus der Entscheidungstheorie stammt die rekursive Modellierungsmethode [GDW91]. Hierbei werden Auszahlungsfunktionen verwendet, die für jede mögliche Entscheidung und jeden Agenten die Nützlichkeit angeben. Die globale Entscheidung wird nun rekursiv entwickelt, indem jeder Agent nicht nur seine eigene Auszahlungsfunktion betrachtet, sondern auch überlegt, wie sich wohl ein anderer Agent verhält. Diese wiederum überlegt, was der erste Agent über seine Entscheidungsfindung denken könnte - und so weiter.

Die Grundlage für die schlußendliche Entscheidung sind in den beiden genannten Ansätzen im wesentlichen monetäre Aspekte: Einem Agenten und einem Ziel wird jeweils ein Nutzen zugeordnet; dieser soll nach Möglichkeit maximiert werden.

Eine andere Möglichkeit, verschiedene Präferenzrelationen zu kombinieren ist die Anwendung des Dempster-Shafer Belief-Kalküls. Barnett [Bar91] stellt einen darauf basierenden Algorithmus vor, der anhand einzelner, gewichteter Präferenzen eine Reihenfolge der Ausführung von Zielen bestimmt, so daß die Präferenzrelationen möglichst wenig mißachtet werden (dieses Problem ist NP-vollständig).

Je nach Bereich ist es sinnvoll, über das „abstimmen" hinaus von einem globalen Standpunkt eines Supervisors aus verschiedene Ziele zu koordinieren. Wie in [EpR92b] beschrieben darf dies aber kein „master-slave"-Verhältnis sein. Während vom Supervisor erwartet wird, daß er den Überblick hat, können einzelne Agenten Wissen haben, das der Supervisor nicht hat, z.B. findet ein auf dem Mond ausgesetzter Roboter möglicherweise Bodenverhältnisse vor, mit denen der Supervisor auf der Erde nicht rechnet. Es gilt hier, eine sinnvolle Abwägung zwischen den verschiedenen Plänen von Supervisor und Agenten zu finden.

In [ShT92] wird vorgeschlagen, bei Konflikten als Mittelweg zwischen globaler Kontrolle aller Agenten und lokalen Verhandlungen zwischen einzelnen Agenten gesellschaftliche Gesetze aufzustellen. Jede Situationsveränderung muß mit diesen Gesetzen verträglich sein. Als Beispiel wird das Rechtsfahrgebot im Straßenverkehr angeführt. Drohende Kollisionen werden dann weder durch globale Regelungen für alle Fahrzeuge in einer Situation, noch durch zeitraubende Einzelverhandlungen abgewendet, sondern dadurch, daß sich jeweils zwei Fahrzeuge an die Konvention halten.

1.2 Ein Ansatz mit mehrwertiger Logik

In diesem Beitrag wird ein Ansatz vorgestellt, Voten verschiedener Agenten als Wahrheitswerte einer mehrwertigen Logik aufzufassen, die neben den üblichen logischen Verknüpfungen zwei weitere Operatoren anbietet. Von hier aus können lokale Verhandlungen und globale Koordinierung in einem größeren Rahmen behandelt werden.

Angenommen, es wird eine Anfrage an eine Menge von Agenten gestellt, deren Antwort „ja" oder „nein" ist oder ganz entfällt. Das Ergebnis ist in einem Paar $(\{a_{1_1}, \ldots, a_{1_t}\}, \{a_{2_1}, \ldots, a_{2_t}\})$ zusammenzufassen, wobei die erste Komponente angibt, welche Agenten mit „ja" antwortet und die zweite, welche Agenten mit „nein" antworten.

Auf der Menge dieser Paare sind in natürlicher Weise zwei partielle Ordnungen definiert:

$(U,V) \leq_t (S,T) :\Leftrightarrow U \subseteq S$ und $T \subseteq V$ bzw. $(U,V) \leq_k (S,T) :\Leftrightarrow U \subseteq S$ und $V \subseteq T$.

„t" steht für „truth" und „k" für „knowledge". $(U,V) \leq_t (S,T)$ bedeutet, daß (S,T) eher zu der globalen Antwort „ja" tendiert als (U,V), während $(U,V) \leq_k (S,T)$ gedeutet wird als die Aussage „(S,T) ist eine umfassendere Information als (U,V), denn es haben mehr Agenten überhaupt geantwortet". Beide Anordnungen induzieren jeweils eine Verbandsstruktur und sind zusätzlich auf eine gewisse Weise ineinander verschränkt. Strukturen dieser Art werden Bilattices genannt.

Diese Begriffsbildung geht zurück aus Matthew L. Ginsberg [Gin88] und fungierte zunä
als Rahmen für mehrwertige, nichtmonotone und Defaultlogiken. In diesem Zusammenhang
auch die Arbeiten von Melvin Fitting zu nennen ([Fit91],[Fit92]). Bilattices als Formalisme
Behandlung unvollständiger und widersprüchlicher Informationen bieten sich aber auch an,
divergierende, verteilte Daten zu koordinieren. Die einzelnen Wahrheitswerte spiegeln zum
einen Wahrheitsgehalt, zum anderen einen Informationsgehalt wider. Insbesondere wird feh
Information durch den Wahrheitswert „unknown" explizit dargestellt. Neben ihrer
Anschaulichkeit bieten Bilattices die Möglichkeit, in flexibler Weise Koordinierungsoperato
zu konstruieren.

In [Mes93] werden Bilattices im Zusammenhang mit der Integration von Wissensbasen
verwendet. Im folgenden Beitrag werden Bilattices als formale Grundlage für die Koordinie
divergierender Ziele vorgestellt. In Kapitel 2 geben wir eine Einführung in die Theorie der
Bilattices. Im Anschluß betrachten wir Beispiele (Kapitel 3).

2 Theoretischer Hintergrund

Ausgehend von mehrwertigen Logiken, die eine Ordnung auf Wahrheitwerten verwende
definiert Ginsberg [Gin88] Bilattices als Mengen von Wahrheitswerten mit zwei darauf erkl
Ordnungen, die beide eine Verbandsstruktur induzieren (daher der Name), zum einen der
abgestufte Wahrheitsgehalt zwischen „falsch" und „wahr", zum anderen der anwachsende
Informationsgehalt zwischen „unbekannt" und „überbestimmt".

2.1 Definition: Bilattice

*Ein **Bilattice** ist eine Struktur* $\mathcal{B} = (B, \otimes, \oplus, \wedge, \vee)$, *so daß* $(B, \otimes, \oplus)$ *und* $(B, \wedge, \vee)$ *vollstä*
Verbände sind und für die es eine Abbildung
$\neg : B \to B$, *genannt **Negation** gibt, so daß gilt :*
i) $\neg(\neg a) = a$,
ii) $\neg$ *ist ein Verbandshomomorphismus von*
$(B, \wedge, \vee)$ *nach* $(B, \vee, \wedge)$ *und von* $(B, \otimes, \oplus)$ *nach* $(B, \otimes, \oplus)$.
Sind $x,y \in B$, *dann ist* $x \leq_t y :\Leftrightarrow x \wedge y = x, x \leq_k y :\Leftrightarrow x \otimes y = x$.
Die minimalen bzw. maximalen Elemente bzgl. $\leq_t$ *(degree of truth) heißen **false** bzw. **true***
bzgl $\leq_k$ *(degree of knowledge) **unknown** bzw **both**.*

$\wedge$ und $\vee$ sind motiviert durch die üblichen logischen Verknüpfungen. $\otimes$ ist der „Konsens
Operator". p$\otimes$q gibt an, worauf p und q maximal übereinstimmen.

$\oplus$ ist der „gutgläubige" Operator. p $\oplus$ q „vereinigt" die Ansichten von p und q ohne Rück
auf möglicherweise entstehende Widersprüche.

Die Interpretation der k-Richtung als anwachsendes **Wissen** ist im Fall verschiedener Age
nicht zwingend. Daß ein Wahrheitswert k-größer als ein anderer ist, bedeutet in diesem
Zusammenhang eher, daß mehr Evidenz oder eine stärkere Präferenz durch einen oder mehr
Agenten vorliegt.

2.2 Der Bilattice T(I,I)

Als Beispiel betrachten wir für das Einheitsintervall I = [0,1] (oder Teilmengen davon)

$T(I,I) := \{(x,y) | x,y \in I\}$

mit $(x,y) \leq_t (u,v) : \Leftrightarrow x \leq u$ und $v \leq y$, sowie $(x,y) \leq_k (u,v) : \Leftrightarrow x \leq u$ und $y \leq v$,

und $\neg(u,v) := (v,u)$.

Hierbei wird x als „pro"- und y als „contra"-Faktor gedeutet:

(0,1) bedeutet hier, daß „nichts dafür und alles dagegen" spricht, entspricht also dem Wert „falsch" bzw. „ablehnen". $\neg(0,1) = (1,0)$ bedeutet, daß „nichts dagegen und alles dafür" spricht, (0,0) entspricht „nichts bekannt" und (1,1), daß maximal viel dafür, aber auch dagegen spricht.

Beispielsweise ist $(0.1,0.5) \otimes (0.4,0.2) = (0.1,0.2)$ und $(0.1,0.5) \oplus (0.4,0.2) = (0.4,0.5)$.

$\wedge$- und $\vee$- Verknüpfung werden in der mehrwertigen Logik i.a. als Minimum- bzw. Maximumbildung aufgefaßt (siehe z.B. [Sch90]). Entsprechend wird hier bei der $\wedge$-Verknüpfung das Minimum des „pro"-Anteils und das Maximum des „contra"-Anteils gebildet - für die $\vee$-Verknüpfung dual. So ist z.B. $(0.4,0.5) \wedge (0.3,0.2) = (0.3,0.5)$ und $(0.4,0.5) \vee (0.3,0.2) = (0.4,0.2)$.

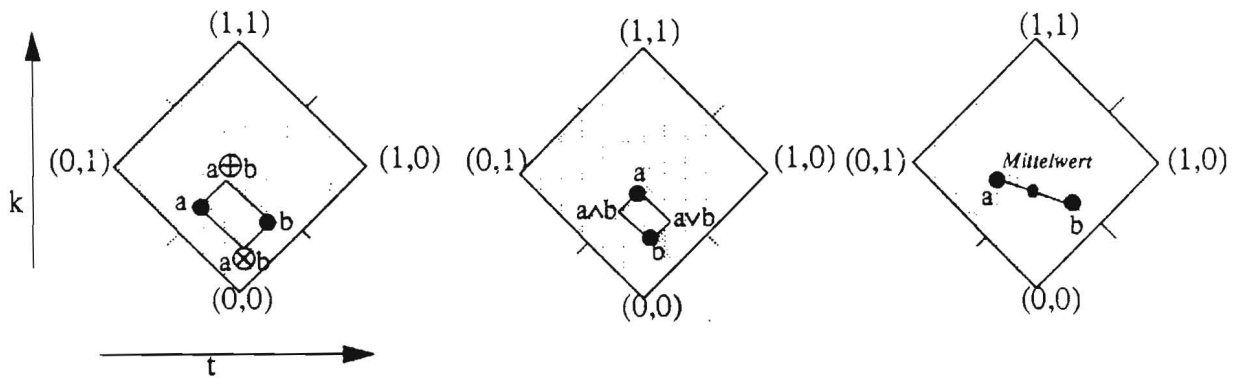


Abbildung 1. T(I,I)

Statt „pro" und „contra"- Faktoren kann man die Elemente von T(I,I) auch als Nutzen- und Kostenfaktoren auffassen. Dann gibt $a \oplus b$ an, wieviel Nutzen bzw. Kosten bei a und b maximal entstehen, $a \otimes b$ minimalen Nutzen bzw. Kosten, $a \wedge b$ minimiert den Nutzen und maximiert die Kosten, $a \vee b$ umgekehrt.

Im übrigen können in T(I,I) auch Konvexkombinationen gebildet werden wie in einem gewöhnlichen Vektorraum, d.h.

$\lambda(x,y) + \mu(x',y') = (\lambda x + \mu x', \lambda y + \mu y')$ für $\lambda+\mu = 1$. Für $\lambda = \mu = 0.5$ entspricht das einer Mittelwertbildung.

2.3 Der Bilattice D_n

Als nächstes betrachten wir den Bilattice der Default-Logik, D_1 und seine Verallgemeinerung auf n-1 Defaultwerte, dargestellt in Abbildung 2 .

Es ist auch hier nicht zwingend, „k" als „knowledge" zu interpretieren. $x \leq_k y$ kann hier auch aufgefaßt werden als „y dominiert x". Urteile von Agenten können in einer entsprechenden Hierarchieebene eingeordnet werden und bei der Verknüpfung mit $\oplus$ wird der Wert der höchsten

Hierarchieebene angenommen, auf der ein Urteil vorliegt. Urteile „niederer" Ebenen werden dem Fall nicht beachtet (schraffierte Fläche in Abbildung 2).

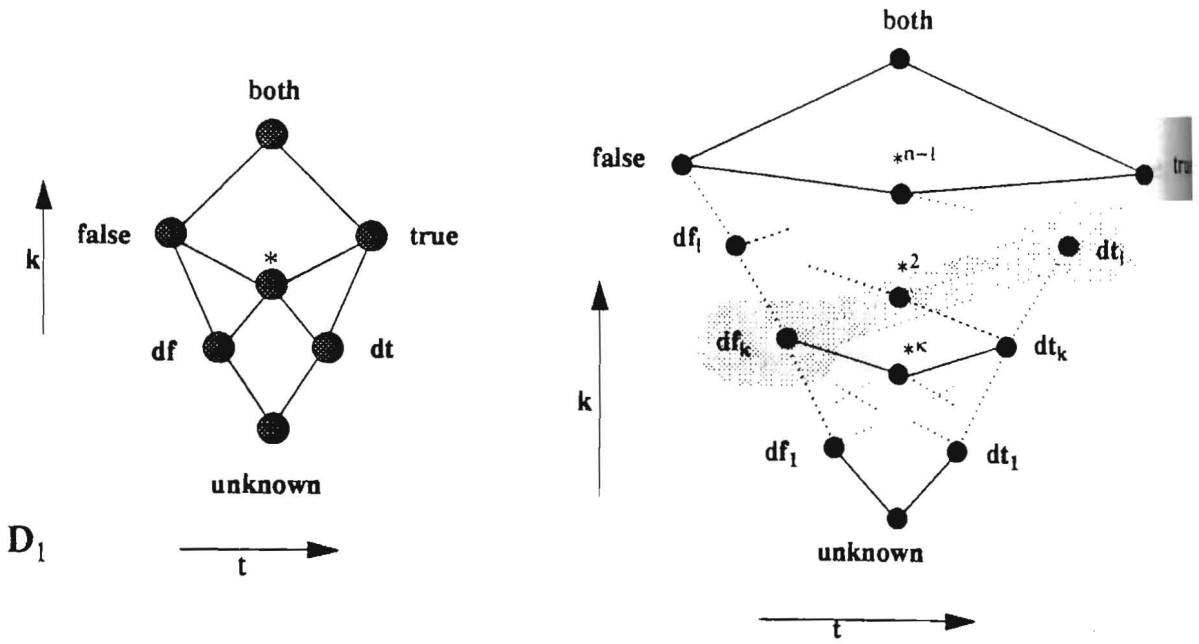


Abbildung 2. Die Bilattices D_1 und D_n.

2.4 Distributivität

Sei $(V,\cup,\cap)$ ein Verband. V heißt distributiv :⇔ für alle a, b, c ∈ V gilt
$(a\cup b)\cap c = (a\cap c)\cup (b\cap c)$ und $c\cap (a\cup b) = (c\cap a)\cup (c\cap b)$.
Sei $\mathcal{B} = (B, \wedge, \vee, \otimes, \oplus)$ ein Prä - Bilattice. $\mathcal{B}$ heißt
t - distributiv :⇔ $(B, \wedge, \vee)$ distributiv,
k - distributiv :⇔ $(B, \otimes, \oplus)$ distributiv,
kreuzweise distributiv :⇔ $\wedge$ und $\vee$ erfüllen die Distributivgesetze mit $\otimes$ und $\oplus$,
distributiv :⇔ $\mathcal{B}$ t-, k- und kreuzweise distributiv.

2.5 Lemma

In einem distributiven Bilattice gilt:

i) false $\otimes$ true = unknown, false $\oplus$ true = both,

ii) both $\wedge$ unknown = false, both $\vee$ unknown = true.

In einem distributiven Bilattice sind z.B. Minimum- und Konsensbildung vertauschbar. T(I,I) ist distributiv. Der wesentliche Unterschied zwischen distributiven und nicht-distributiven Bilattices besteht darin, daß es durch die Operatoren $\oplus$ oder $\otimes$ in einem distributiven Bilattice nicht möglich ist, widersprüchliche Ansichten zu eindeutigen Antworten zu verknüpfen. Bei der Konsensbildung in T(I,I) entsteht im Falle widersprechender Ansichten stets das Ergebnis „unentschieden": $(x_1,0) \otimes (0,y_2) = (0,0)$.

In D_n entscheidet bei der Verknüpfung mit $\oplus$ das Urteil auf höherer Ebene:
$df_k \oplus dt_l = dt_l$, falls $k < l$.

2.6 Die Bewertungsfunktion

Die Bewertungsfunktion skaliert die Wahrheitswerte eines Bilattices auf Werte zwischen -1 (ablehnen) und +1 (zustimmen), bzw. berechnet die Differenz zwischen Nutzen und Kosten. Zur Definition verwenden wir die sogenannte Höhenfunktion in Verbänden (siehe [BIR67]).

Definition 2.6.1 Höhe

Sei $\mathcal{B} = (B, \wedge, \vee, \otimes, \oplus)$ ein Bilattice, $x \in B$.
$h_t(x) := sup\ \#\{K/\ K$ t-Kette zwischen f und $x\}$ ***t-Höhe*** *von x.*
$h_t(B) := h_t(t)$ heißt ***t-Höhe*** *von $\mathcal{B}$.*

Definition 2.6.2 Bewertungsfunktion

i) Sei $\mathcal{B} = (B, \wedge, \vee, \otimes, \oplus)$ ein endlicher Bilattice.

$\gamma: B \rightarrow [-1,1]$

$$x \rightarrow \frac{h_t(x) - h_t(\neg x)}{h_t(B)}$$ *heißt* ***Bewertungsfunktion.***

ii) In $T([0,1],[0,1])$ sei die Bewertungsfunktion durch

$\gamma(x,y) = x-y$ erklärt.

Man rechnet leicht nach, daß die Definition der Bewertungsfunktion in T(I,I) die natürliche Fortsetzung vom endlichen zum unendlichen Fall ist, d.h. es gilt

2.7 Lemma

In T(I,I) mit endlichem $I = \{ \frac{1}{n}, \frac{2}{n}, ..., 1 \}$ ist $\gamma(x,y) = x-y$.

❐

3 Beispiele

3.1 Das Szenario

Wir betrachten eine Gruppe $\mathcal{A}$ von Agenten und eine (endliche) Menge $\mathcal{S}$ von Situationen und die Aufgabe der Agenten, sich in einer gegebenen Situation s_0 auf eine Folgesituation $F(s_0)$ zu einigen. Sei $\mathcal{B} = (B, \wedge, \vee, \otimes, \oplus)$ ein Bilattice. Jeder Agent gibt zu jeder möglichen Folgesituation einen Wert $x \in B$ an, der seine Präferenz bezüglich der Folgesituation angibt. Diese Einschätzungen notieren wir für die Agenten $A_1, A_2, ...$ als Funktionen $\pi_i : S \times S \rightarrow B$ $(i=1,2,...)$.

Die Koordinierungsfunktion coord: $S \rightarrow 2^S$ gibt in Abhängigkeit von den Präferenzen der einzelnen Agenten an, was aufgrund übergeordneter Kriterien oder Maßstäben als Folgesituation in Frage kommt. Im allgemeinen wird sich keine eindeutige Folgesituation finden lassen.

3.2 Beispiel: „Veto“

Soll das Urteil eines bestimmten Agenten dann zum Tragen kommen, wenn er ablehnt, so aber nicht, so läßt sich das in T([0,1],[0,1]) ausdrücken durch

$ko_k((x_a,y_a),(x_k,y_k)) = (x_a,y_a) \wedge ((x_k,y_k) \vee (1-y_k,1-x_k))$.

Hierbei bezeichnet (x_a,y_a) das Ergebnis ohne Agent k, und (x_k,y_k) das Urteil des Agenten Man sieht, daß, wenn in (x_k,y_k) der Wert $y_k = 0$ ist, $ko_k((x_a,y_a),(x_k,y_k)) = (x_a,y_a)$ ist, d.h. der V unverändert ist, wenn aber $(x_k,y_k) = (0,1)$ ist, ist $ko_k((x_a,y_a),(x_k,y_k)) = (0,1)$.

Die Abbildung $(x,y) \rightarrow (1-y,1-x)$ wird Conflation genannt. Es ist die analoge Operation zu Negation in der k - Dimension.

Ein Veto kann eingesetzt werden, wenn Constraints formuliert werden sollen, um bestimm Zustände zu verbieten. In [ShT92] werden mit Constraints soziale Gesetze formuliert, wodur bestimmte Folgezustände ausgeschlossen werden.

Werden zum Beispiel für die Situation s_0 und die Folgesituationen s_1 und s_2 von den Agen A_1 und A_2 die Werte der unten angegebenen Tabelle angegeben, dann sind bei der Kosten - Nutzenrechnung s_1 und s_2 gleich günstig oder ungünstig, wie sich durch Berechnung von $\gamma(0.5\pi_1(s_0,s_1)+ 0.5\pi_2(s_0,s_1))$ und $\gamma(0.5\pi_1(s_0,s_2)+ 0.5\pi_2(s_0,s_2))$ ergibt.

Durch eine Gesetzgebung π_G, die (s_0,s_1) ausschließt, bleibt jedoch nur s_2 übrig.

	π_1	π_2
(s_0,s_1)	(0.6,0.2)	(0.2,0.5)
(s_0,s_2)	(0.5,0.1)	(0.1,0.4)

	π_G
(s_0,s_1)	(0,0)
(s_0,s_2)	(0,1)

Abbildung 3. Urteile von Agenten

Mögliche Koordinierungsfunktion ist hier

$s \rightarrow \{t \in S \mid \gamma(ko_k(\text{Mittelwert}(\pi_1(s,t),\pi_2(s,t)), \pi_G(s,t))) \geq 0 \text{ maximal}\}$.

Es ist wichtig, daß die Bewertungsfunkion einen positiven Wert anzeigt, denn sonst wird entgegen ablehnender Beurteilung gehandelt. In diesem Fall ist coord(s) = $\emptyset$.

Es ist klar, daß auch durch derartige „Gesetze“ die Nachfolgesituation nicht immer eindeu ist.

3.3 Beispiel: Bücherkauf

Wir denken uns ein Forschungsinstitut, das Bücher beschaffen will. Es bestehe aus drei Abteilungen, wovon eine ein neu eingerichteter Sonderforschungsbereich sei. Dann kann man der Bücherbeschaffung wie folgt vorgehen:

- Innerhalb der einzelnen Gruppen ($\{A_1,A_2,A_3\}$, $\{A_4,A_5\}$ und $\{A_6,A_7\}$) wird der Minimalkonsens gebildet.

- Können sich die beiden „alteingesessenen“ Abteilungen ($\{A_1,A_2,A_3\}$ und $\{A_4,A_5\}$)nicht eindeutig äußern, wird ein Buch (tendenziell) nicht beschafft.

- Die Voten der beiden alteingesessenen Abteilungen werden um den Faktor 0.8 verringert. Das Votum des Sonderforschungsbereichs ($\{A_6,A_7\}$) wird nicht verringert.

Die Situation bestehen in diesem Fall aus Mengen von Büchern, die vorhanden sind. Insbesondere ist s_0 die Situation, daß kein Buch vorhanden ist (jedenfalls kein neues). Wir wollen uns bei den Nachfolgesituationen auf diejenigen beschränken, die sich von der vorhergehenden höchstens durch ein Element unterscheiden (d.h. höchstens ein Buch wurde gekauft).

Als Koordinierungsfunktion läßt sich aus den obigen Kriterien die in Abbildung 3 gezeigte Funktion F konstruieren.

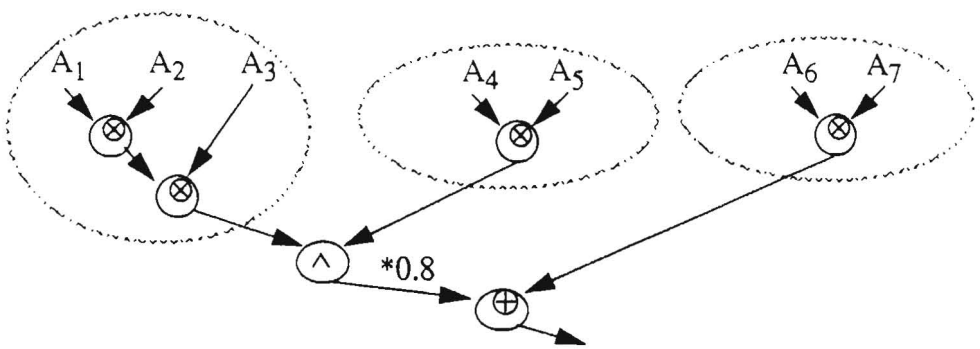


Abbildung 4. Graphische Darstellung von F

Die Bewertung einer Folgesituation durch die Teilnehmer hängt nun davon ab, welches Buch beschafft wird. Angenommen, die einzelnen Personen geben für die verschiedenen Bücher die folgenden Werte an:

π_i (s,t)	A_1	A_2	A_3	A_4	A_5	A_6	A_7
$b_1 \in t$	(1,0)	(1,0)	(1,0)	(0.7,0)	(0.9,0)	(0,0.7)	(0,0.8)
$b_2 \in t$	(0.7,0)	(0,0.9)	(0,0)	(0,1)	(0,0.9)	(1,0)	(1,0)
$b_3 \in t$	(0,1)	(0,0.5)	(0,0.1)	(1,0)	(1,0)	(0,1)	(0.5,0)

Hierbei gehen wir (der Einfachheit halber) davon aus, daß das Urteil nicht davon abhängt, welche Bücher schon da sind. Es ergeben sich die folgende Werte:

	$A_1 \otimes A_2 \otimes A_3$	$A_4 \otimes A_5$	$0.8*((A_1 \otimes A_2 \otimes A) \wedge (A_4 \otimes A_5))$	$A_7 \otimes A_8$	$0.8*((A_1 \otimes A_2 \otimes A_3) \wedge (A_4 \otimes A_5)) \oplus (A_7 \otimes A_8)$
$b_1 \in t$	(1,0)	(0.7,0)	(0.56,0)	(0,0.7)	(0.56,0.7) $\gamma = -0.14$
$b_2 \in t$	(0,0)	(0,0.9)	(0,0.72	(1,0)	(1,0.72) $\gamma = 0.28$
$b_3 \in t$	(0,0.1	(1,0)	(0.8,0.08)	(0,0)	(0.8,0.08) $\gamma = 0.72$

Wird coord(s,t) = $\{t \in S, t \supseteq s | \gamma(F(s,t)) \geq 0$ maximal$\}$ gewählt, ergibt sich für $s_0 = \emptyset$
$s_1 = \{b_3\}$, $s_2 = \{b_2, b_3\}$. s_2 hat keine Folgesituation.

Ist dieses Verfahren in der Agentengesellschaft bekannt, müssen die einzelnen Gruppen versuchen, sich untereinander zu einigen, da ihr Votum sonst quasi verfällt. Außerdem müssen Werte sorgfältig abgewogen werden, um ein zufriedenstellendes Resultat zu erhalten.

Beschränkt man sich bei den Urteilen auf die Werte „ja", „nein" und „unentschieden", kann man auch die Urteile der Gruppen in geeigneter Weise in einen hierarchischen Bilattice (D_n) einbetten. Nach der Konsensbildung innnerhalb der einzelnen Gruppen kann man z.B. die Ergebnisse der ersten beiden Gruppen auf die erste Stufe in D_2 abbilden und mit $\wedge$ verknüpfen, das Ergebnis der Einigung im Sonderforschungsbereich auf die zweite Stufe abbilden und dann die Verknüpfung mit $\oplus$ bilden. Sofern dann im Sonderforschungsbereich eine einhellige Meinung gebildet wurde, wird diese vorgezogen, anderenfalls bestimmen die anderen beiden Gruppen die Entscheidung. Hinter dieser Verknüpfungsweise steht eine andere Strategie als im obigen Beispiel.

4 Schlußbemerkung und Ausblick

Bilattices sind anschauliche, flexible Strukturen, durch die fehlende und widersprüchliche Informationen sichtbar gemacht werden. Sie bilden einen Rahmen, unterschiedliche Voten von Agenten auf verschiedene Weise zu verknüpfen, vor allem, wenn unterschiedliche Informiertheit, unterschiedliche Funktionen von Agenten oder hierarchische Strukturen vorliegen. Eine Vielfalt von Verknüpfungsmöglichkeiten arithmetischer und logischer Art ergibt sich einerseits durch die Wahl von Koordinierungsfunktionen, die stückweise zusammengesetzt werden können und andererseits in der Wahl verschiedener Bilattices.

Die Koordinierungsfunktionen können zwischen der Gleichbehandlung aller Voten und der Hierarchiebildung ausbalanciert werden, wobei -wie beschrieben- auch Gruppen gebildet werden können. Hier bietet sich eine inkrementelle Vorgehensweise an, wobei die Bilattice - Operatoren eine Art Bausteine bilden. Der Einsatz in den verschiedenen Anwendungsbereichen der VKI ist Gegenstand künftiger Forschung.

Literatur

[Bar91] J.A. Barnett. Combining opinions about the order of rule execution. *Proc. AAAI(91)*

[BIR67] G. Birkhoff. Lattice Theory (3.Auflage) Amer. Math. Soc. Coll. Publ. Vol. 25. Providence RI

[EpR91] E. Ephrati, J. Rosenschein. The Clarke Tax as a consensus mechanism among automated agents. *Proc. AAAI(91)*

[EpR92a] E. Ephrati, J. Rosenschein. Reaching agreement through partial revelation of preferences. *Proc. ECAI (92)*

[EpR92b] E. Ephrati, J. Rosenschein. Constraint intelligent action: Planning under the influence of a master agent. *Proc. AAAI(92)*

[Fit91] M.Fitting. Bilattices and the semantics of logic programming. *J.Logic Programming 11*,1991

[Fit92] M.Fitting. Kleenes logic, generalized. To appear in *J.Logic and Computation*

[Gin88] M.Ginsberg. Multivalued logics. *Computational Intelligence* 4(3), 1988

[Gin91] M.Ginsberg. Bilattices and modal operators. *J.Logic and Computation, 1991*

[GDW91] P. Gmytrasiewicz, E. Durfee, D. Wehe: A decision-theoretic approach to coordinating multiagent interaction. *Proc. IJCAI(1991),Vol.1*

[Mes93] B. Messing. Integration von Wissensbasen: Ein Ansatz mit Bilattices. *Forschungsberichte des Instituts für Angewandte Informatik und Formale Beschreibungsverfahren, Nr. 262, Universität Karlsruhe(TH)*

[Sch90] P.H. Schmitt. Perspectives in multiple-valued logic. *In R. Studer(Ed.): Natural Language and Logic. Symposium Hamburg, 1989*. Lecture Notes in Artificial Intelligence, 459, Springer 1990

[ShT92] Y. Shoham, M. Tennenholz. On the synthesis of useful social laws for artificial agent societies (preliminary report). *Proc. AAAI(92)*

Verteilte Begründungsverwaltung

Thilo Horstmann

Gesellschaft für Mathematik und Datenverarbeitung (GMD)
I3.CSCW
Postfach 1316
5205 Sankt Augustin 1

Übersicht

Um eine domänenunabhängige Theorie von interagierenden, autonomen Agenten zu schaffen, zielt die aktuelle VKI-Forschung auf die Definition abstrakter Agentenmodelle ab. Sie erlaubt die Formalisierung von Kooperationsstrategien und Multi-Agenten Inferenzmechanismen. Um die Autonomität eines Agenten durch globale Verwaltungs- und Kontrollmechanismen nicht zu beeinträchtigen, sollen Agenten über Schlußfolgerungen und Begründungen anderer Agenten verfügen können. Diese Anforderung erfüllen verteilte Begründungsverwaltungssysteme. Wir stellen hier ein solches Verteiltes Begründungsverwaltungssystem (DTMS)[1] vor und führen zentrale Definitionen für Konsistenz in einem Multi-Agenten Szenario ein.

Einleitung

Die bisherige Forschung auf dem Gebiet der Verteilten Künstlichen Intelligenz (VKI) führte zu einer breiten Vielfalt von Anwendungen, die durch autonome, lose zusammenhängende Problemlösungskomponenten gekennzeichnet sind. Jeder einzelne Knoten, oder *Agent*, ist u.a. ausgestattet mit individueller Problemlösungsexpertise. VKI-Anwendungen umfassen einen weiten Bereich von Kooperierenden Expertensystemen über Verteiltes Planen bis hin zu Computergestützter Gruppenarbeit. Um eine domänenunabhängige Theorie von interagierenden, autonomen Agenten zu schaffen, zielt die aktuelle VKI-Forschung auf die Definition abstrakter Agentenmodelle ab, die die Formalisierung von Kooperationsstrategien und Multi-Agenten Inferenzmechanismen gestattet.
Die Anforderungen an Multi-Agenten Inferenzmechanismen sind vielfältig. In den meisten Fällen ist es nicht erwünscht, die Autonomität der Agenten durch eine globale Kontrollstruktur einzuschränken, die die Verwaltung aller Inferenzen und Regeln verschiedener Agenten übernimmt. Gründe werden ausführlich in [3] diskutiert. Vielmehr verlangen wir, daß die Agenten autonom Inferenzen herleiten können und zwar auf der Grundlage eigenen und fremden Wissens.
Diese Anforderung wird durch verteilte
verwaltungssysteme erfüllt. Sie erweitern
Begründungsverwaltung auf Multi-Agent
Das DTMS muß dabei einen bestimmten
sistenz der Schlußfolgerungen eines Mult
ferenzprozesses gewährleisten. Verschiede
Konsistenz werden später in diesem Auf
Eine weitere wichtige Aufgabe des DTM
der Bereitstellung von Informationen für
dule des Agenten. Z.B. kann eine Infere
Mehrfachberechnungen vermeiden und ein
onseinheit kann Kooperationsprozesse au
len, konsistenten Zustand der Wissensbas
Das hier vorgestellte DTMS basiert auf
Wir glauben, ein JTMS ist für unsere Zwe
eignet als ein ATMS aus folgendem Grund
eines Agenten, d.h. seine Annahmen, Prä
geln, werden während eines Multi-Agente
zesses sehr umfangreich. Das ATMS muß
tretenden Kontexte berechnen. Die schnell
des ATMS wird daher überschattet von
tiellen Verhalten des Markierungsalgorith
In diesem Aufsatz erweitern und modifi
in [1] vorgestellte DTMS. Hier werden Da
keiten aussagenlogisch dargestellt. Es ba
vollständigen Markierungsalgorithmus vo
toren von [1] definieren "shared beliefs" u
Stufen von Konsistenz in einem Multi-Ag
ein. Wie wir aber später sehen werden,
Ansatz ein Agent über einen anderen A
minieren, *obwohl er weniger Information*
len Fällen geht so wichtige Information e
Agenten in einem Multi-Agenten Inferenz
loren.
Unsere Definitionen von Konsistenz garan
breitung von Information zu allen relevan
weil sie Beliefs mit den dazugehörigen
berücksichtigt. Wenn später ein Agent
für ein erworbenes Belief falsifiziert, kann
der das Belief ursprünglich bewiesen ha
neute Beweisfindung einbezogen werden.
Wir modellieren Datenabhängigkeiten in

[1] engl. Distributed Truth Maintenance System

gik Erster Stufe. Regeln und Beliefs, die bereits inferiert wurden, werden von einem Prolog Meta Interpreter ausgewertet.

Das DTMS ist durch die folgenden Punkte gekennzeichnet:

- Erhaltung eines konsistenten Zustandes von Schlußfolgerungen.

  Wenn die Wissensbasis verändert wird, erfordert wegen der Aufzeichnung von Datenabhängigkeiten die Wiederherstellung von Konsistenz nur wenig Rechenaufwand.

- Datenabhängigkeiten werden in Prädikatenlogik 1. Stufe statt in Aussagenlogik ausgedrückt.

- Explizite Repräsentation von Beweisen gestattet einfache Generierungen von Erklärungen.

- Schnittstelle zum Austausch von Schlußfolgerungen, Daten und Beweisen zwischen Agenten.

- Meta-Level Prädikate unterstützen den Entwurf von klar spezifizierten, autonomen Agenten.

  Durch sie können dynamische Ziele, auf die die Begründungsverwaltung angewendet werden soll und statische Ziele, unterschieden werden.

- Das DTMS ist eine Erweiterung von konventionellen TMS und nicht auf das Einsatzgebiet VKI beschränkt.

Terminologie

Eine DTMS-Anfrage bewirkt die Interpretation einer endlichen Menge von Horn-Klauseln[2]. Wir teilen diese Menge in zwei disjunkte Mengen ein: Die Menge der (dynamischen) Rechtfertigungnen $\mathcal{J}$,[3] und die Menge der statischen und System-Prädikate $\mathcal{S}$. Beim Aufruf erstellt oder modifiziert das DTMS *Beliefs*. Informell ist ein Belief eine atomare Formel, der 4 Attribute zugeordnet werden:

- *Status:* eins der Symbole **in** oder **out**
- *Constraint:* eine Formel 1. Ordnung in Konjunktiver Normal Form
- *Support:* eine Liste atomarer Formeln
- *Konsequenzen:* eine Liste atomarer Formeln

[2] In diesem Aufsatz benutzen wir die Notation für logisches Programmieren wie in [7] eingeführt

[3] Eine Rechtfertigung ist eine Klausel mit nicht leerem Rumpf. Eine Rechtfertigung mit dem Symbol *true* als Rumpf ist eine *Prämisen-Rechtfertigung*

Das *Status*-Feld kennzeichnet, ob das Atom mit den mentanen Klauseln beweisbar ist (in bzw. out). Formel in *Constraint* kann als Grund für diese Zu sung angesehen werden. Die Felder *Support* und *Ko quenzen* bezeichnen die Abhängigkeiten des Atoms l der momentanen Menge von Beliefs. D.h., *Konseq zen* repräsentiert genau die Beliefs, die neuberec werden müssen, wenn der Status wechselt. In der deren Richtung bezeichnen die Elemente von *Sup* die Beliefs, die zur Berechnung des Status' des At zur Anwendung kamen.

Diese Begriffe werden im folgenden genauer definie

Definition 1 *Sei $\mathcal{A}$ eine endliche Menge positive terale, $\mathcal{L}(\mathcal{A})$ deren Potenzmenge und $\mathcal{F}(\mathcal{A})$ die M aller Formeln, die aus Elementen von $\mathcal{A}$ gebildet we können. Ein* **Zustand** *Ψ is a 4-Tupel $\Psi = (\lambda, \mu, \nu, \xi$ daß*

1. $\lambda : \mathcal{A} \rightarrow \{\mathtt{in}, \mathtt{out}\}$
2. $\mu : \mathcal{A} \rightarrow \mathcal{F}(\mathcal{A})$
3. $\nu : \mathcal{A} \rightarrow \mathcal{L}(\mathcal{A})$
4. $\xi : \mathcal{A} \rightarrow \mathcal{L}(\mathcal{A})$

Definition 2 *Sei $\theta = \{v_1/t_1, \cdots, v_n/t_n\}$ eine Su tution und* j *eine Rechtfertigung. Dann ist j_θ eine* **stanz** *von* j, *wenn alle Vorkommen der Variablen* j *gleichzeitig ersetzt werden durch den Term t_i (i $\cdots$, n).*

Definition 3 *Sei $\Psi = (\lambda, \mu, \nu, \xi)$ ein Zustand un eine Instanz einer Rechtfertigung mit dem Kopf h_θ Rumpf b_θ. Sei $\mathcal{P}$ eine Menge von Programmklau Dann ist j_θ*

1. **gültig** *(bzgl. Ψ), wenn θ eine korrekte Antw ubstitution für $\mathcal{P} \cup \{b\}$ ist[4]. In diesem Fall gil jedes positive Literal p von b_θ, $\lambda(p) =$ **in** und jedes positive Gegenstück eines negativen Liter von j_θ, $\lambda(n) =$ **out**. Wir sagen, j_θ rechtfertigt*
2. **ungültig** *(bzgl. Ψ), wenn θ keine korrekte wortsubstitution für $\mathcal{P} \cup \{b\}$ ist. In diesem Fall es entweder ein positives Literal l von b_θ mit $\lambda($* **out**, *oder ein positives Gegenstück eines negat Literals l von b_θ mit $\lambda(l) =$* **in**. *Wir sagen, l fiziert j_θ.*

[4] D.h., $\forall(b\theta)$ ist eine logische Folgerung von $\mathcal{P}$.

gik Erster Stufe. Regeln und Beliefs, die bereits inferiert wurden, werden von einem Prolog Meta Interpreter ausgewertet.

Das DTMS ist durch die folgenden Punkte gekennzeichnet:

- Erhaltung eines konsistenten Zustandes von Schlußfolgerungen.

  Wenn die Wissensbasis verändert wird, erfordert wegen der Aufzeichnung von Datenabhängigkeiten die Wiederherstellung von Konsistenz nur wenig Rechenaufwand.

- Datenabhängigkeiten werden in Prädikatenlogik 1. Stufe statt in Aussagenlogik ausgedrückt.

- Explizite Repräsentation von Beweisen gestattet einfache Generierungen von Erklärungen.

- Schnittstelle zum Austausch von Schlußfolgerungen, Daten und Beweisen zwischen Agenten.

- Meta-Level Prädikate unterstützen den Entwurf von klar spezifizierten, autonomen Agenten.

  Durch sie können dynamische Ziele, auf die die Begründungsverwaltung angewendet werden soll und statische Ziele, unterschieden werden.

- Das DTMS ist eine Erweiterung von konventionellen TMS und nicht auf das Einsatzgebiet VKI beschränkt.

Terminologie

Eine DTMS-Anfrage bewirkt die Interpretation einer endlichen Menge von Horn-Klauseln[2]. Wir teilen diese Menge in zwei disjunkte Mengen ein: Die Menge der (dynamischen) Rechtfertigungnen $\mathcal{J}$,[3] und die Menge der statischen und System-Prädikate $\mathcal{S}$. Beim Aufruf erstellt oder modifiziert das DTMS *Beliefs*. Informell ist ein Belief eine atomare Formel, der 4 Attribute zugeordnet werden:

- *Status:* eins der Symbole `in` oder `out`
- *Constraint:* eine Formel 1. Ordnung in Konjunktiver Normal Form
- *Support:* eine Liste atomarer Formeln
- *Konsequenzen:* eine Liste atomarer Formeln

[2] In diesem Aufsatz benutzen wir die Notation für logisches Programmieren wie in [7] eingeführt

[3] Eine Rechtfertigung ist eine Klausel mit nicht leerem Rumpf. Eine Rechtfertigung mit dem Symbol *true* als Rumpf ist eine *Prämisen-Rechtfertigung*

Das *Status*-Feld kennzeichnet, ob das Atom mit den momentanen Klauseln beweisbar ist (`in` bzw. `out`). Die Formel in *Constraint* kann als Grund für diese Zuweisung angesehen werden. Die Felder *Support* und *Konsequenzen* bezeichnen die Abhängigkeiten des Atoms bzgl. der momentanen Menge von Beliefs. D.h., *Konsequenzen* repräsentiert genau die Beliefs, die neuberechnet werden müssen, wenn der Status wechselt. In der anderen Richtung bezeichnen die Elemente von *Support* die Beliefs, die zur Berechnung des Status' des Atoms zur Anwendung kamen.

Diese Begriffe werden im folgenden genauer definiert:

Definition 1 *Sei $\mathcal{A}$ eine endliche Menge positiver Literale, $\mathcal{L}(\mathcal{A})$ deren Potenzmenge und $\mathcal{F}(\mathcal{A})$ die Menge aller Formeln, die aus Elementen von $\mathcal{A}$ gebildet werden können. Ein* **Zustand** *Ψ is a 4-Tupel $\Psi = (\lambda, \mu, \nu, \xi)$ so daß*

1. $\lambda : \mathcal{A} \rightarrow \{\mathtt{in}, \mathtt{out}\}$
2. $\mu : \mathcal{A} \rightarrow \mathcal{F}(\mathcal{A})$
3. $\nu : \mathcal{A} \rightarrow \mathcal{L}(\mathcal{A})$
4. $\xi : \mathcal{A} \rightarrow \mathcal{L}(\mathcal{A})$

Definition 2 *Sei $\theta = \{v_1/t_1, \cdots, v_n/t_n\}$ eine Substitution und* j *eine Rechtfertigung. Dann ist j_θ eine* **Instanz** *von* j, *wenn alle Vorkommen der Variablen v_i in* j *gleichzeitig ersetzt werden durch den Term t_i ($i = 1, \cdots, n$).*

Definition 3 *Sei $\Psi = (\lambda, \mu, \nu, \xi)$ ein Zustand und j_θ eine Instanz einer Rechtfertigung mit dem Kopf h_θ und Rumpf b_θ. Sei $\mathcal{P}$ eine Menge von Programmklauseln. Dann ist j_θ*

1. **gültig** *(bzgl. Ψ), wenn θ eine korrekte Antwortsubstitution für $\mathcal{P} \cup \{b\}$ ist*[4]. *In diesem Fall gilt für jedes positive Literal p von b_θ, $\lambda(p) =$* `in` *und für jedes positive Gegenstück eines negativen Literals n von j_θ, $\lambda(n) =$* `out`. *Wir sagen, j_θ rechtfertigt h_θ.*
2. **ungültig** *(bzgl. Ψ), wenn θ keine korrekte Antwortsubstitution für $\mathcal{P} \cup \{b\}$ ist. In diesem Fall gibt es entweder ein positives Literal l von b_θ mit $\lambda(l) =$* `out`, *oder ein positives Gegenstück eines negativen Literals l von b_θ mit $\lambda(l) =$* `in`. *Wir sagen, l falsifiziert j_θ.*

[4] D.h., $\forall(b\theta)$ ist eine logische Folgerung von $\mathcal{P}$.

Beispiel 1 *Sei* $\mathcal{A} = \{a(2), b(2), c(2), d(2)\}$ *mit* $\lambda(a(2)) = \lambda(d(2)) =$ `in` *und* $\lambda(b(2)) = \lambda(c(2)) =$ `out`. *Weiter haben wir drei Rechtfertigungen j1:* $d(X) \leftarrow a(X), \neg b(X)$, *j2:* $d(X) \leftarrow a(X), b(X)$ *und j3:* $c(X) \leftarrow d(X)$. *Dann ist die Instanz von* $j1_{\{X/2\}}$ *gültig, aber* $b(2)$ *falsifiziert* $j2_{\{X/2\}}$.

Definition 3 verallgemeinert die Definition einer aussagenlogischen Rechtfertigung auf den prädikatenlogischen Fall. Somit repräsentiert eine Rechtfertigung in Prädikatenlogik 1. Stufe die Menge aller Instanzen einer Rechtfertigung.
Um jetzt den zentralen Term *Konsistenz* einzuführen, benötigen wir weitere Definitionen.

Definition 4 *Sei* $\Psi = (\lambda, \mu, \nu, \xi)$ *ein Zustand und sei* $j_1, \cdots, j_n$ *eine Menge von Rechtfertigungen mit alle dem gleichen Prädikat* a *als Kopf. Falls jede Rechtfertigung* j_i *falsifiziert wird von einem* b_i, *so bezeichnen wir die Menge* $\{b_1, \cdots, b_n\}$ *mit* **inval (a)**.

Definition 5 *Sei* p *ein Prädikat und* $\mathcal{J}$ *eine Menge von Rechtfertigungen. Die* **Definition einer Rechtfertigung** *(geschrieben* def (p)*) ist die Disjunktion aller Rümpfe von Klauseln von* $\mathcal{J}$ *mit demgleichen Prädikat* p *im Kopf.*

Definition 6 *Sei* c *eine Konjunktion von atomaren Formeln* $a_1, \cdots, a_n$. *Dann bezeichnet [c] die Menge von Atomen* $a_1, \cdots, a_n$.

Definition 7 *Sei* $\Psi = (\lambda, \mu, \nu, \xi)$ *ein Zustand.* con (a) *ist die Menge von Atomen, deren Elemente die Atome* $c \in \mathcal{A}$ *sind, so daß* a *ein Element von* $\nu(c)$ *ist.*

Definition 8 *Sei* $\Psi = (\lambda, \mu, \nu, \xi)$ *ein Zustand und* $\mathcal{P} = \mathcal{J} \cup \mathcal{S}$ *die Vereinigung von disjunkten Mengen von Programmklauseln.* Ψ *ist* **konsistent**, *wenn folgende Bedingungen gelten:*

1. *wenn* $\lambda(a_\theta) =$ `in`, *dann gilt entweder*
   (a) *es gibt ein* $j \in \mathcal{J}$ *so daß* j_θ *rechtfertigt* a_θ. *In diesem Fall* $\mu = b_\theta, \nu = [b_\theta], \xi = con(a_\theta)$ *oder*
   (b) θ *ist eine korrekte Antwortsubstitution* $\mathcal{S} \cup \{a\}$. *In diesem Fall,* $\mu = system$[5], $\nu = [system], \xi = con(a_\theta)$.
2. *wenn* $\lambda(a_\theta) =$ `out`, *dann ist* θ *keine* [illegible] *wortsubstitution von* $\mathcal{P} \cup \{a\}$. *Wenn* [illegible] *finition für a gibt, dann* $\mu = cnf($[illegible]$)$, $\nu = inval(a)$, $\xi = con(a_\theta)$. *In allen* [illegible] $\mu = system$, $\nu = [system]$, $\xi = con(a_\theta)$
3. *es gibt keine Folge* $(a_0, \cdots, a_n)$ *mit* [illegible] $\mathcal{A}$, *so daß* $a_0 = a_n$ *und für* $i = 1, \cdots,$ [illegible] *und* a_{i-1} *ist in* $\mu(a_i)$.

Definition 8 impliziert einige erwähnenswert[illegible] Bedingung 3 verhindert zyklische Abhängigk[illegible] Support-Feld von `in`-Beliefs um so die W[illegible] heit zu garantieren. Weiterhin beinhaltet ein[illegible] ter Zustand von Beliefs die korrekte Zuweisu[illegible] schen Zuständen zu atomaren Formeln und [illegible] Verkettung der Beliefs in Übereinstimmung [illegible] gischen Abhängigkeiten.

Beispiel 2 *Der Zustand in dem Beispiel* [illegible] *tion 3 ist* inkonsistent*: Der Belief* $c(2)$ *ist o*[illegible] *Instanz* $j3_{\{X/2\}}$ *ist gültig. Somit ist Bedin*[illegible] *Definition 8 verletzt.*

Um einen konsistenten Zustand von Belief[illegible] sentieren, speichert das DTMS für ein Atom [illegible] entsprechenden Werte von λ, μ, ν, ξ in den [illegible] ten `status, constraint, support` und [illegible] ces von `dtms_node/6`[7]:

```
dtms_node(datum,status,constraint,
          support,consequences,rule_id)
```

Das Symbol `true` kann in dem Supportfeld [illegible] in zwei Fällen auftreten: `datum` ist `in` und [illegible] tigt mit einer Prämisenrechtfertigung oder da[illegible] und die Rechtfertigung, die mit `datum` mat[illegible] haltet keine weiteren Unterziele. Z.B. die Rech[illegible] $p \leftarrow fail$, ist falsifiziert durch das Symbol [illegible] $cnf(not(def(p))) = true$ ist das Supportfeld [illegible] p das Symbol `true`.

Zusätzlich bezeichnet das Argument `rule_id` [illegible] deutigen Bezeichner der Rechtfertigung, die f[illegible] dung des Supportfeldes benutzt wurde.

Die allgemeine Darstellung von Rechtfertigun[illegible]

```
dtms_rule (justification, rule_id)
```

[5] D.h., das Symbol *system* im Supportfeld kennzeichnet, daß der Beweis mit Regeln geschah, auf die keine Begründungsverwaltung angewendet wird

[6] cnf bezeichnet die Konjunktive Normal Form eine[illegible] Formel.

[7] Das abstrakte mathematische Objekt *Menge* wir[illegible] Prolog Objekt *list* repräsentiert.

[8] Die Beliefs *true* und *fail* sind in und out, aber n[illegible] in dem DTMS repräsentiert

Die Köpfe der Rechtfertigungen definieren die Atome, auf die die Begründungsverwaltung angewendet wurde. Im Gegensatz zu herkömmlichen TMS müssen diese Ziele also nicht explizit angegeben werden.

Beliefs in dem DTMS

Im folgendem betrachten wir eine Menge von DTMS, jedes bezeichnet durch eindeutige DTMS-Bezeichner. Jedes DTMS beinhaltet statische Prädikate, lokale Rechtfertigungen und seine eigenen Beliefs. Zusätzlich kann ein DTMS Beliefs von anderen DTMS *erwerben* oder Beliefs zu anderen DTMS *propagieren*. Die nächsten Definitionen präzisieren diese Begriffe.

Definition 9 *Sei $\mathcal{P}$ eine Menge von Programmklauseln, $\mathcal{B}$ eine Menge von Beliefs und $\Psi = (\lambda, \mu, \nu, \xi)$ der Zustand von $\mathcal{B}$. Dann nennen wir das Tripel $\delta = (\mathcal{P}, \mathcal{B}, \Psi)$ ein* **DTMS**. *Der DTMS-Bezeichner δ ist logisch äquivalent zu dem Symbol* true.

Definition 10 *Sei $\mathcal{D} = \{\delta_1 = (\mathcal{P}_1, \mathcal{B}_1, \Psi_1), \cdots, \delta_n = (\mathcal{P}_n, \mathcal{B}_n, \Psi_n)\}$ eine Menge von DTMS. Eine* **positive dtms-rule** *von $\mathcal{P}_j$ ist eine Rechtfertigung der Form*

$$a \leftarrow \delta_i$$

und eine **negative dtms-rule** *ist von der Form*

$$a \leftarrow \neg\delta_i$$

wobei a eine atomare Formel und $i \in \{1, \cdots, n\}, (i \neq j)$.

Die Definitionen 9 und 10 implizieren, daß die dtms-rule $a \leftarrow \delta_i$ gültig ist, wenn $\lambda(a) =$ **in** und die negative rule $a \leftarrow \neg\delta_i$ is gültig, wenn $\lambda(a) =$ **out**.

Informell besagt eine dtms-rule folgendes: Ein Belief a, das von einem DTMS mit einer positiven dtms-rule hergeleitet wurde, kann interpretiert werden als "I glaube an a, weil DTMS δ_i es beweisen kann" und eine negative als "Ich glaube nicht an a, weil weder ich noch DTMS δ_i a beweisen können[9]." Somit repräsentiert eine dtms-rule eine Inferenz in einem anderen Agenten.

Beliefs können also wie folgt kategorisiert werden:

Definition 11 *Sei $\mathcal{D} = \{\delta_1 = (\mathcal{P}_1, \mathcal{B}_1, \Psi_1), \cdots, \delta_n = (\mathcal{P}_n, \mathcal{B}_n, \Psi_n)\}$ eine Menge von DTMS. Ein Belief $b_i \in \mathcal{B}_i$, das das Atom l_i bezeichnet, ist*

[9] Vorausgesetzt natürlich, daß das DTMS keine eigene gültige Rechtfertigung für a besitzt.

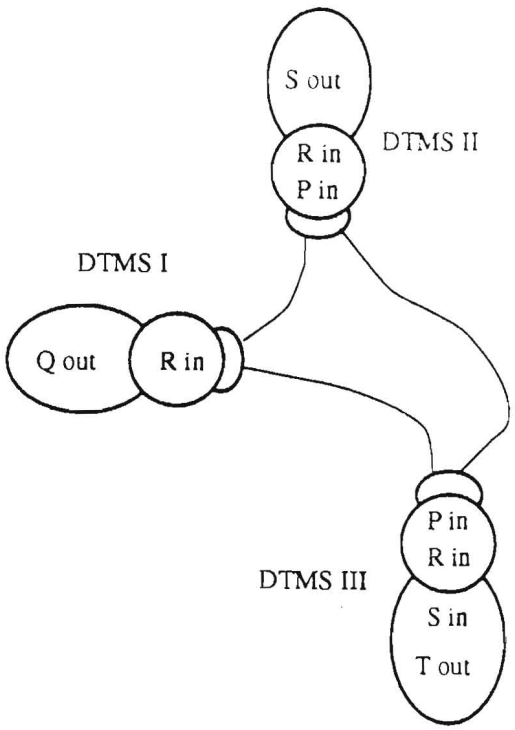


Abbildung 1: Gemeinsame und wechselseitige Beliefs

1. **privat** *von δ_i, wenn es kein Belief b_j in $\mathcal{B}_j$ gibt, so daß l_i mit l_j unifiziert werden kann, $(i \neq j)$.*
2. **gemeinsam** *zu δ_i und δ_j, wenn es ein Belief b_j in $\mathcal{B}_j$ gibt, so daß l_i mit l_j unifiziert werden kann, $(i \neq j)$. Der Status von b_i kann unterschiedlich zu dem Status von b_j sein.*
3. **propagiert** *zu DTMS δ_j, wenn $\mathcal{P}_j$ entweder eine positive dtms-rule der Form $l_i \leftarrow \delta_i$ oder eine negative dtms-rule der Form $l_i \leftarrow \neg\delta_i$ $(i \neq j)$ beinhaltet.*
4. **erworben** *von DTMS δ_j, wenn $\mathcal{P}_i$ entweder eine positive dtms-rule der Form $l_i \leftarrow \delta_j$ oder eine negative dtms-rule der Form $l_i \leftarrow \neg\delta_j$, $(i \neq j)$ beinhaltet.*
5. **wechselseitig** *zu δ_i und δ_j, wenn b_i propagiert ist zu δ_j.*

Einen Belief zu propagieren bedeutet eine dtms-rule zu übergeben. D.h., das DTMS, das die dtms-rule erwirbt, kann mit ihr eigene, lokale Inferenzen herleiten. Insbesondere kann das DTMS einen eigenen Belief erstellen mit demselben Atom und Status wie in dem propagierendem DTMS. Deshalb sprechen wir von dem Propagieren von Beliefs anstelle von Rechtfertigungen.

Abbildung 1 zeigt ein Beispiel für Beliefs. Belief Q ist privat zu DTMS I, Belief S ist gemeinsam von DTMS II und III und Belief R is wechselseitig zu I, II und III.

Die Köpfe der Rechtfertigungen definieren die Atome, auf die die Begründungsverwaltung angewendet wurde. Im Gegensatz zu herkömmlichen TMS müssen diese Ziele also nicht explizit angegeben werden.

Beliefs in dem DTMS

Im folgendem betrachten wir eine Menge von DTMS, jedes bezeichnet durch eindeutige DTMS-Bezeichner. Jedes DTMS beinhaltet statische Prädikate, lokale Rechtfertigungen und seine eigenen Beliefs. Zusätzlich kann ein DTMS Beliefs von anderen DTMS *erwerben* oder Beliefs zu anderen DTMS *propagieren*. Die nächsten Definitionen präzisieren diese Begriffe.

Definition 9 *Sei $\mathcal{P}$ eine Menge von Programmklauseln, $\mathcal{B}$ eine Menge von Beliefs und $\Psi = (\lambda, \mu, \nu, \xi)$ der Zustand von $\mathcal{B}$. Dann nennen wir das Tripel $\delta = (\mathcal{P}, \mathcal{B}, \Psi)$ ein* **DTMS**. *Der DTMS-Bezeichner δ ist logisch äquivalent zu dem Symbol* true.

Definition 10 *Sei $\mathcal{D} = \{\delta_1 = (\mathcal{P}_1, \mathcal{B}_1, \Psi_1), \cdots, \delta_n = (\mathcal{P}_n, \mathcal{B}_n, \Psi_n)\}$ eine Menge von DTMS. Eine* **positive dtms-rule** *von $\mathcal{P}_j$ ist eine Rechtfertigung der Form*

$$a \leftarrow \delta_i$$

und eine **negative dtms-rule** *ist von der Form*

$$a \leftarrow \neg\delta_i$$

wobei a *eine atomare Formel und $i \in \{1, \cdots, n\}, (i \neq j)$.*

Die Definitionen 9 und 10 implizieren, daß die dtms-rule $a \leftarrow \delta_i$ gültig ist, wenn $\lambda(a) =$ **in** und die negative rule $a \leftarrow \neg\delta_i$ is gültig, wenn $\lambda(a) =$ **out**.

Informell besagt eine dtms-rule folgendes: Ein Belief a, das von einem DTMS mit einer positiven dtms-rule hergeleitet wurde, kann interpretiert werden als "I glaube an a, weil DTMS δ_i es beweisen kann" und eine negative als "Ich glaube nicht an a, weil weder ich noch DTMS δ_i a beweisen können[9]." Somit repräsentiert eine dtms-rule eine Inferenz in einem anderen Agenten.

Beliefs können also wie folgt kategorisiert werden:

Definition 11 *Sei $\mathcal{D} = \{\delta_1 = (\mathcal{P}_1, \mathcal{B}_1, \Psi_1), \cdots, \delta_n = (\mathcal{P}_n, \mathcal{B}_n, \Psi_n)\}$ eine Menge von DTMS. Ein Belief $b_i \in \mathcal{B}_i$, das das Atom l_i bezeichnet, ist*

[9] Vorausgesetzt natürlich, daß das DTMS keine eigene gültige Rechtfertigung für a besitzt.

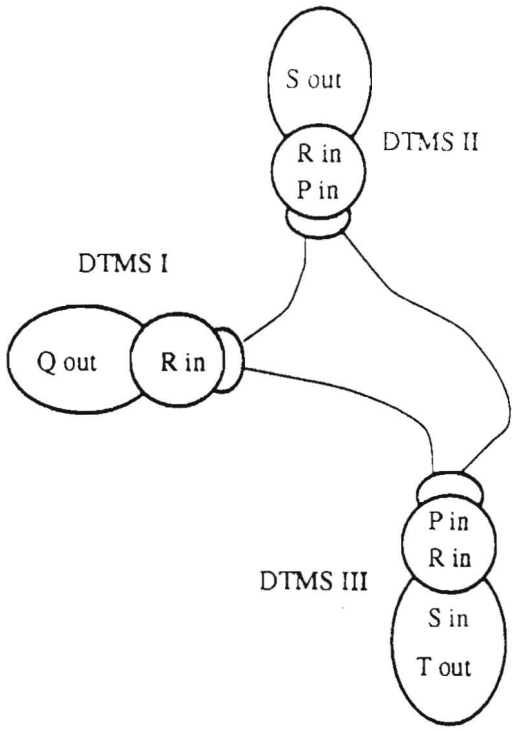


Abbildung 1: Gemeinsame und wechselseitige Beliefs

1. **privat** *von δ_i, wenn es kein Belief b_j in $\mathcal{B}_j$ gibt, so daß l_i mit l_j unifiziert werden kann, $(i \neq j)$.*
2. **gemeinsam** *zu δ_i und δ_j, wenn es ein Belief b_j in $\mathcal{B}_j$ gibt, so daß l_i mit l_j unifiziert werden kann, $(i \neq j)$. Der Status von b_i kann unterschiedlich zu dem Status von b_j sein.*
3. **propagiert** *zu DTMS δ_j, wenn $\mathcal{P}_j$ entweder eine positive dtms-rule der Form $l_i \leftarrow \delta_i$ oder eine negative dtms-rule der Form $l_i \leftarrow \neg\delta_i$ $(i \neq j)$ beinhaltet.*
4. **erworben** *von DTMS δ_j, wenn $\mathcal{P}_i$ entweder eine positive dtms-rule der Form $l_i \leftarrow \delta_j$ oder eine negative dtms-rule der Form $l_i \leftarrow \neg\delta_j$, $(i \neq j)$ beinhaltet.*
5. **wechselseitig** *zu δ_i und δ_j, wenn b_i propagiert ist zu δ_j.*

Einen Belief zu propagieren bedeutet eine dtms-rule zu übergeben. D.h., das DTMS, das die dtms-rule erwirbt, kann mit ihr eigene, lokale Inferenzen herleiten. Insbesondere kann das DTMS einen eigenen Belief erstellen mit demselben Atom und Status wie in dem propagierendem DTMS. Deshalb sprechen wir von dem Propagieren von Beliefs anstelle von Rechtfertigungen.

Abbildung 1 zeigt ein Beispiel für Beliefs. Belief Q ist privat zu DTMS I, Belief S ist gemeinsam von DTMS II und III und Belief R is wechselseitig zu I, II und III.

Viele Fragen treten auf, wenn wir DTMS betrachten, die ihre Beliefs propagieren können. Wir diskutieren sie in dem nächsten Abschnitt, wenn wir Begriffe für Konsistenz in verteilten Szenarien einführen.

Konsistenz

Definition 10 zusammen mit Definition 8 erlaubt uns die Konsistenz eines DTMS zu definieren. Informell ist ein DTMS lokal konsistent, wenn seine privaten und erworbenen Beliefs konsistent bzgl. der Menge seiner Programmklauseln ist.

Definition 12 *Sei $\{\delta_1, \cdots, \delta_n\}$ eine Menge von DTMS. Das DTMS $\delta_i = (\mathcal{P}, \mathcal{B}, \Psi)$ ist* **lokal konsistent**, *wenn sein Zustand Ψ konsistent ist.*

Definition 13 *Sei $\mathcal{D} = \{\delta_1 = (\mathcal{P}_1, \mathcal{B}_1, \Psi_1), \cdots, \delta_n = (\mathcal{P}_n, \mathcal{B}_n, \Psi_n)\}$ eine Menge von DTMS. $\mathcal{D}$ is* **beweiskonsistent**, *wenn die folgenden Bedingungen gelten:*

1. *Jedes DTMS $\delta_i \in \mathcal{D}$ ist lokal konsistent.*
2. *Wenn ein Belief $b \in \mathcal{B}_i$ propagiert ist zu DTMS $\delta_j (i \neq j)$, dann sind alle Vorfahren $f_1, \cdots, f_m$ von b entweder propagiert zu DTMS δ_j oder erworben.* b *und das erworbenene Gegenstück in DTMS δ_j sind entweder beide* `in` *oder beide* `out`.
3. *Ein erworbenes Belief $b_i \in \mathcal{B}_i$ ist nicht propagiert.*
4. *Es gibt keine Menge von Beliefs $(b_0, \cdots, b_n) \in \mathcal{B}_1 \cup \cdots \cup \mathcal{B}_n$, so daß $b_0 = b_n$ und für $i = 1, \cdots, n$, gilt entweder*
   (a) *$\lambda(b_i) =$* `in` *und b_{i-1} ist in $\mu(b_i)$ oder*
   (b) *es gibt ein* k *so daß $b_{i-1} \in \mathcal{B}_k$ und $\delta_k \in \mu(b_i)$.*

Definition 13 erlaubt die unterschiedliche Markierung von gemeinsamen Beliefs. Bedingung 4 garantiert die Wohl-Fundiertheit der wechselseitigen Beliefs.

Beispiel 3 *Seien $\mathcal{D} = \{\delta_1 = (\mathcal{P}_1, \mathcal{B}_1, \Psi_1), \delta_2 = (\mathcal{P}_2, \mathcal{B}_2, \Psi_2)\}$ zwei DTMS mit*

$\mathcal{P}_1 = \{b \leftarrow a, a \leftarrow \delta_2\}$
$\mathcal{B}_1 = \{b, a\}$ *mit* $\lambda(b) = \lambda(a) =$ `in`, $\nu(b) = a$, $\xi(b) = \{\}$ *und* $\nu(a) = \delta_2$, $\xi(a) = b$.

$\mathcal{P}_2 = \{a \leftarrow b, b \leftarrow \delta_1\}$
$\mathcal{B}_2 = \{b, a\}$ *mit* $\lambda(b) = \lambda(a) =$ `in`, $\nu(a) = b$, $\xi(a) = \{\}$ *und* $\nu(b) = \delta_1$, $\xi(b) = a$.

Wir nehmen an, daß Belief b von δ_1 zu δ_2 zu δ_1 propagiert sind. $\{\delta_1, \delta_2\}$ sind nicht k*stent, weil die Wohl-Fundiertheitsbedingung Aber jedes einzelne DTMS ist lokal konsisten note 12).*

Beachte, daß ein Belief nur einmal in dem pagiert sein kann. D.h., ein DTMS kann k propagieren, daß es bereits von einem ande erworben hat. Auf der anderen Seite könn viele DTMS denselben Belief erwerben.

Dieses Konzept definiert klar einen konsis stand von wechselseitig abhängigen Beliefs zw schiedenen Agenten. Dieser Zustand ist durch tausch von Schlußfolgerungen *und* deren Beg gekennzeichnet. Dadurch wird gewährleistet mationen, die zur Falsifizierung oder Valid Beliefs relevant sind, zu allen betreffenden D pagiert werden: Da ein DTMS die Begründ erworbenen Beliefs kennt, kann es das ents DTMS informieren, wenn eine Begründung u Ein DTMS wird dabei nicht mit Information da nur eine minimale Repräsentation der fr renzen nötig ist. Dies wird in dem folgend deutlich.

Beispiel zum Gebrauch des DTM

Um unsere Definition von Konsistenz zu moti zu erklären, diskutieren wir ein verteiltes Sz trachte die folgenden Klauseln und Abbildu

Dynamische Klauseln von DTMS I:

```
dtms_node(engine_ok, in,
        (valve1_open , not 'temp>90'),
        [valve1_open, 'temp>90'], [],
dtms_node(valve1_open, in, true,
        [true], [engine_ok], j2).
dtms_node('temp>90', out, true,
        [true], [engine_ok], _g55).
dtms_rule(engine_ok :- (valve1_open ,
        not 'temp>90'), j1).
dtms_rule(valve1_open :- true, j2).
```

Dynamische Klauseln von DTMS II:

```
dtms_node(engine_ok, in,
        (valve2_open , not 'temp>90'),
        [valve2_open, 'temp>90'], [],
```

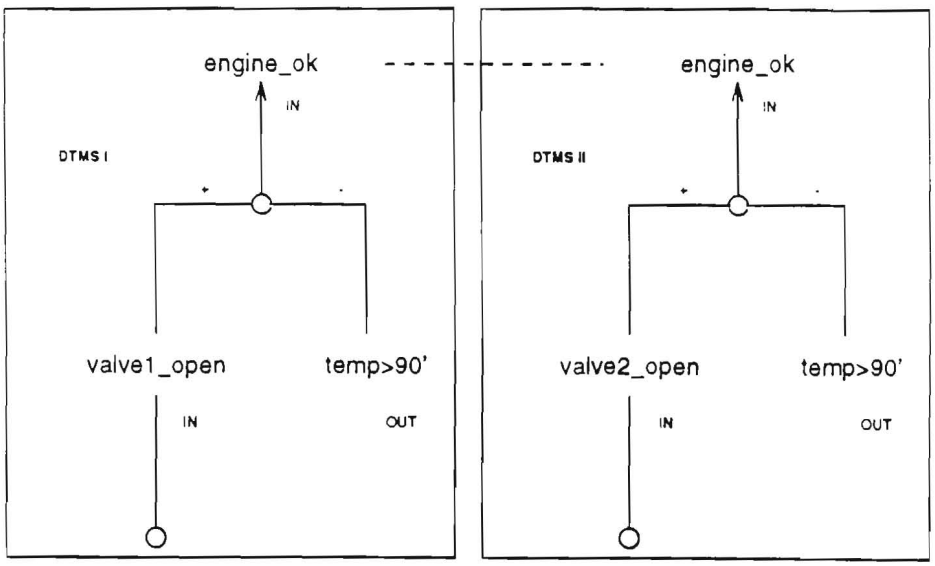


Abbildung 2: Beispiel Szenario

```
dtms_node(valve2_open, in, true,
        [true], [engine_ok], j2).
dtms_node('temp>90', out, true,
        [true], [engine_ok], _g55).
dtms_rule(engine_ok :- (valve2_open ,
        not 'temp>90'), j1).
dtms_rule(valve2_open :- true, j2).
```

DTMS I und DTMS II sind zwei autonome Überwachungssysteme, die das korrekte Arbeiten einer Maschine überwachen. Die Maschine arbeitet korrekt, wenn entweder Ventil 1 oder Ventil 2 geöffnet sind, die Temperatur aber nicht $90°C$ übersteigt. Da die Temperatur so entscheidend ist, verwenden wir zwei autonome Kontrollsysteme, jedes verbunden mit einem DTMS. Weiterhin überwacht jedes Kontrollsystem jeweils ein Ventil. Wir wollen, daß die DTMS eine kooperative Lösung über den Belief **engine_ok** hervorbringen (repräsentiert durch die gestrichelte Linie in Abbildung 2), d.h., wir wollen nicht, daß die DTMS **engine_ok** jeweils unterschiedliche Zustände zuordnen.
Eine initiale Anfrage über **engine_ok** resultiert in der Lösung **engine_ok** is **in**, in Übereinstimmung mit beiden DTMS. Nehmen wir nun an, daß DTMS II eine neue Rechtfertigung für den Belief **temp>90** erhält, weil sein Sensor eine Temperatur größer als $90°C$ detektiert. Dadurch würde DTMS II die Beliefs **temp>90** und **engine_ok** jeweils zu **in** und **out** markieren, im Widerspruch zu den entsprechenden Markierungen in DTMS I.

Der verteilte Markierungsalgorithmus in [1], würde nun den Belief **engine_ok** von DTMS II mit dem Symbol **in** markieren[10] , weil DTMS I noch eine gültige Rechtfertigung für **engine_ok** besitzt (Der Temperatursensor von DTMS I hat aus irgend einem Grund nicht das Ansteigen der Temperatur erkannt). Das System soll jedoch nicht das Funktionieren der Maschine anzeigen. Der Grund für das Fehlverhalten ist: In unserem Beispiel ist nur das Ergebnis einer Inferenz propagiert, nicht aber die Gründe dafür. Somit kann DTMS II nicht wissen, daß DTMS I weiterhin inferiert, daß die Maschine korrekt arbeitet, weil es keinen Grund zur Annahme hat, daß die Temperatur $90°C$ übersteigt. Also wird DTMS II nicht DTMS I über seine Registrierung der hohen Temperatur benachrichtigen und DTMS I wird dominieren, obwohl es weniger Informationen besitzt.

Mit unserer Terminologie ist der oben dargestellte Zustand *nicht* beweiskonsistent, weil Bedingung 2 von Definition 13 verletzt ist. Wir verlangen, daß die Vorfahren eines propagierten Beliefs ebenso propagiert werden. In unseren System wird das folgende geschehen, wenn die DTMS eine kooperative Lösung über Belief b hervorbringen sollen. Über ein Kooperationsprozess wählen die DTMS einen Beweis eines einzelnen DTMS aus. Dieses DTMS ist *logisch verantwortlich* für Belief b[11]. In diesem DTMS werden *alle* Vorfahren $f_1 \cdots f_n$ von b als propagiert markiert. Alle anderen in den Kooperationsprozess einbezogenen DTMS erwerben DTMS Regeln der Form

dtms_rule((f_i**:-responsible_dtms)**, id_i**)**.

[10] Genauer, das Symbol **external**. Es ist logisch äquivalent zu dem Symbol **in**, bezeichnet aber daß die gültige Rechtfertigung sich in einem anderen Agenten befindet.

[11] An dieser Stelle können komplexe Kooperationsmethoden zur Anwendung kommen, um das logisch verantwortliche DTMS auszuwählen. In unserem Beispiel kann es zufällig ausgesucht werden.

bzw.

```
dtms_rule((f_i :-not responsible_dtms), id_i).
```

für jeden Vorfahr $f_1 \cdots f_n$. Die positive dtms-rule wird hinzugefügt, wenn der entsprechende Vorfahr in dem logisch verantwortlichem DTMS **in** ist, ansonsten die negative. [12]

Wir nehmen an, daß nun DTMS I logisch verantwortlich für den Belief **engine_ok** ist. DTMS I propagierte seine Beliefs **engine_ok**, **valve1_open**, **temp>90** zu DTMS II. Für jeden propagierten Belief existiert ein Eintrag der Form **transmitted_node (node, dtms_list)** der in dem zweiten Argument angibt, zu welchen DTMS der Belief propagiert wurde. Die drei DTMS-nodes beschreiben den aktuellen Zustand der Beliefs, wie oben beschrieben. DTMS II, erwarb drei dtms-rules von DTMS I:

```
tms_rule('temp>90' :- not dtms_1, j5).
tms_rule(valve1_open :- dtms_1, j4).
tms_rule(engine_ok :- dtms_1, j3).
```

und erstellte die folgenden Knoten:

```
tms_node(valve2_open,in,true,[true],[],j2).
tms_node(engine_ok,in,dtms_1,[dtms_1],[],j3).
tms_node(valve1_open,in,dtms_1,[dtms_1],[],j4).
tms_node('temp>90',out,dtms_1,[dtms_1],[],_g55).
```

Wir sehen an dem Zustand der Beliefs von DTMS II, daß der Belief **engine_ok** nun von DTMS I und nicht von seiner eigenen Rechtfertigung j1 begründet wird (gl. das 4. Argument des zweiten dtms_node). j1 von TMS II ist aber immer noch gültig.

Die DTMS sind ***beweiskonsistent***: Jedes DTMS ist lokal konsistent alle wechselseitigen Beliefs haben denselben Status und alle Vorfahren von propagierten Beliefs sind ebenso propagiert. Weiterhin ist die Menge der dtms-Beliefs wohl-fundiert bzgl. Bedingung 4 von Definition 13.

[2] Damit können wir das Beispiel aus Definition 13 wie folgt in Prolog ausdrücken:

DTMS I:

```
dtms_rule((b:-a), j1).
dtms_rule((a:-dtms_2), j2).
dtms_node(b, in, a, [a], [], j1).
dtms_node(a, in, dtms_2, [dtms_2], [b], j2).
```

DTMS II:

```
dtms_rule((a:-b), r1).
dtms_rule((b:-dtms_1), r2).
dtms_node(a, in, b, [b], [], r1).
dtms_node(b, in, dtms_1, [dtms_1], [a], r2).
```

Nehmen wir jetzt an, daß DTMS II eine neue Rechtfertigung **dtms_rule ('temp>90':-true** hält, die die Beweiskonsistenz verletzt. Da die von **temp>90** ursprünglich von DTMS I herrührt, DTMS II den neuen Belief mittels der dtms-rule **90 :- dtms_2** zu DTMS I propagieren. Da DTMS eine neue gültige dtms-rule für den Belief **temp** DTMS II erwirbt, wird DTMS I seinen Belief **engine_ok** mit **out** markieren. Dies bezieht wiederum die **engine_ok** in DTMS II mit ein, weil **engine_ok** DTMS I zu DTMS II propagiert wurde. Somit ten wir die Markierungen **engine_ok** zu **out** in DTMS:

Knoten von DTMS I:

```
dtms_node(valve1_open, in, true,
        [true], [], j2).
dtms_node('temp>90', in, dtms_2,
        [dtms_2], [engine_ok], j3).
dtms_node(engine_ok, out, not valve1_ope
        'temp>90',['temp>90'],[],_g55)
```

Knoten von DTMS II:

```
dtms_node(valve2_open, in, true,
        [true], [], j2).
dtms_node(valve1_open, in, dtms_1,
        [dtms_1], [], j3).
dtms_node('temp>90', in, true, [true],
        [engine_ok], j6).
dtms_node(engine_ok, out, ((not valve2_
        'temp>90') , dtms_1),
        ['temp>90', dtms_1], [], _g55)
```

Wie wir in diesem Beispiel sehen, sind jetzt beide logisch verantwortlich für den Status von engine **out**. DTMS I inferierte **valve1_open** und propagierte es zu DTMS II, während umgekehrt DTMS II **out** inferierte und zu DTMS I propagierte.

Unser Konzept erlaubt ein DTMS mit eigenen Rechtfertigungen und Regeln von anderen DTMS zu nutzen. Dabei wird ein bestimmter Grad von Konsistenz erhalten, der korrekte und konsistente Antworten auf Anfragen gewährleistet, dabei aber den Informationsfluß zwischen Agenten gering hält.

Weiterhin braucht ein DTMS nicht die Struktur der Abhängigkeiten von Beweisen in anderen DTMS verwalten, nur die Vorfahren und deren Zustände müssen propagiert werden. Dies erlaubt den DTMS Inferenzen auf eine effiziente Weise zu speichern.

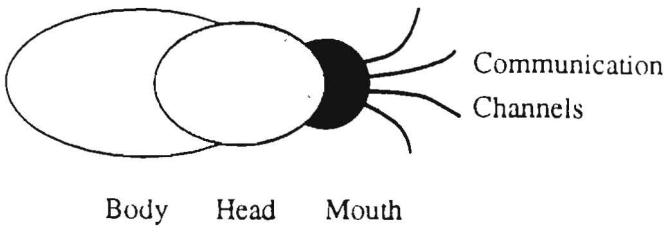


Abbildung 3: Teile eines Agenten.

Das DTMS im abstrakten Agenten Modell

Um mit dem DTMS Maschinen-Agenten zu modellieren, benutzen wir das abstrakte Agenten Modell aus [10], [8]. Diese Modell teilt einen Agenten in drei Hauptteile. (Abbildung 3): Der *Mund* des Agenten realisiert die Kommunikationsfunktionalität des Agenten. Über Kommunikationskanäle empfängt er Nachrichten und leitet sie an seinen Kopf weiter. In der anderen Richtung verschickt er die Nachrichten vom Kopf des Agenten. Der Mund muß mit genügendem Netzwerkwissen ausgestattet sein, so z.B. mit den physikalischen Adressen anderer Agenten oder Wissen, wie diese Adressen zu erreichen sind. Komplexere Implementationen von dem Mund eines Agenten können verschiedene Kommunikationsformate, wie z.B. natürliche Sprache, graphische Darstellungen oder Bitströme, behandeln. Die Autoren von [10] unterscheiden auch die Priorität einer Nachricht oder den Typ der erwarteten Antwort.

Der *Kopf* des Agenten beinhaltet Mechanismen zur Kooperation und zur Inferenzkontrolle. Er ist zum einen ausgestattet mit Kenntnissen über eigene Fähigkeiten (autoepistemisches Wissen) als auch mit Kenntnissen über Fähigkeiten, Zustand und Verhalten anderer Agenten (epistemisches Wissen). Die folgende Liste beschreibt weitere Punkte, mit denen der Kopf ausgestattet sein sollte:

- Wissen über den Zustand der aktuellen Aufgabe
- Aufgabenzergliederungs-Algorithmen
- Methoden, die das eigene Kooperationsverhalten in Abhängigkeit der globalen Kooperationsstrategie verändern.

Der Kopf des Agenten ist somit eine Art Vermittler zwischen der eigenen Funktionalität und dem globalen Problemlösungskontext.

Schließlich realisiert der Rumpf die Basisfunktionalität des Agenten. Nach [10] ist deren Komplexität nicht beschränkt: Ein einfacher Sensor kann genauso wie ein Expertensystem den Rumpf des Agenten bilden.

Das hier vorgestellte DTMS stellt sowohl meta-logische Auswertung von Inferenzen (Kopf) als auch Basisfunktionalität (Rumpf) zur Verfügung, da es als Prolog Meta Interpreter entworfen wurde, s. Abbildung 4.

Die DTMS Metaebene beinhaltet meta-logische Prädikate, benutzer-definierte Rechtfertigungen, den aktuellen Zustand der Beliefs und das DTMS-Kernel die systen und benutzerdefinierten statischen Prädikate. Die Metaebene überwacht die Evaluierung aller Ziele, verwaltet die Buchhaltung und definiert die DTMS-Schnittstelle. Der Kernel definiert die low-level Prädikate, die durch einen Metaaufruf evaluiert werden, aber deren Beweis nicht durch den Buchhaltemechanismus verwaltet werden soll (vgl. Definition 8).

Das meta-logische Prädikat `dtms_solve/5` spielt eine zentrale Rolle in der Meta-Ebene. Die Definition von `dtms_solve/5` realisiert eine Modifikation eines Standard Prolog Meta-Interpreter. Dieser hat als Argument eine Prolog Anfrage g und versucht sie in Übereinstimmung mit den Klauseln der Meta-Ebene und des Kernels sowie dem momentanen Status der Beliefs zu beweisen. Dabei werden, wenn nötig, alle Datenabhängigkeiten gespeichert oder modifiziert.

Rechtfertigungen werden in der Meta-Ebene definiert. Dies sind dynamische Programmklauseln, die die Atome definieren, auf die die Begründungsverwaltung angewendet werden soll. Atome, auf die keine Begründungsverwaltung angewendet werden soll, werden hingegen im Kernel definiert. Dies sind z.B. Prädikate wie **`member/3`** oder **`append/3`**. Im Kernel sind die Prädikate der Meta-Ebene unsichtbar, aber umgekehrt kann die Meta-Ebene die Kernel Prädikate auswerten. So modelliert der Kernel die statische, die Meta-Ebene die dynamische Welt.

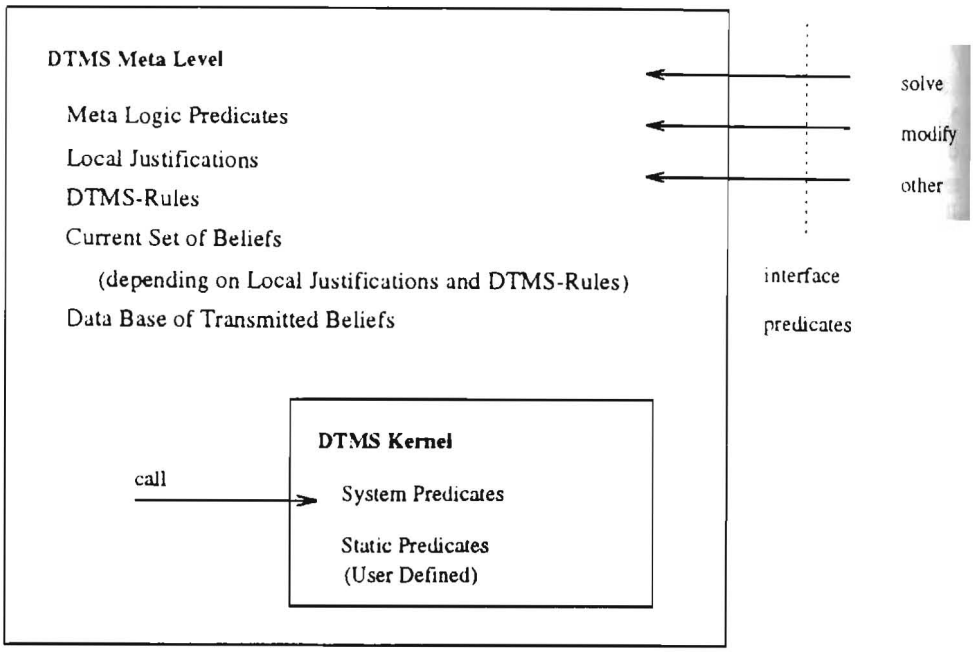


Abbildung 4: DTMS-Architektur.

Zusammenfassung

In diesem Aufsatz zeigten wir ein neues Konzept zur Modellierung von interagierenden DTMS auf und demonstrierten es anhand einer konkreten Implementation. Der zentrale Begriff in diesem Konzept ist der neue Term *Beweiskonsistenz*, der Konsistenz von wechselseitig abhängigen Beliefs in verschiedenen DTMS definiert. Dieser Zustand ist charakterisiert durch den Austausch von Schlußfolgerungen *und* deren Begründungen. Wir zeigten, daß im Gegensatz zu bisherigen Ansätzen, unsere Definition von Konsistenz den Agenten komplexere Inferenzbildungen erlaubt, da das von einem Agenten erworbene Wissen zu allen anderen relevanten Agenten propagiert wird. Da ein DTMS die Begründungen für erworbene Beliefs kennt, kann er so das DTMS informieren, von dem das Belief ursprünglich stammt. Wir zeigten, daß ein DTMS nicht mit Informationen überladen wird, da nur eine minimale Repräsentation von Beweisen propagiert werden muß.

Unser Algorithmus zur Herstellung von Beweiskonsistenz basiert auf einer prädikatenlogischen Sprache 1. Stufe. Damit können Datenabhängigkeiten eleganter ausgedrückt werden als im aussagenlogischem Fall. Die Konstruktion des DTMS als Prolog Meta Interpreter befreit den Anwendungsdesigner von Inferenzkontrollaufgaben und erlaubt ihm zwischen Zielen zu unterscheiden, auf die die Begründungsverwaltung angewendet werden soll und auf welche nicht. Somit wird unnötiger Informationsaustausch zwischen den Agenten vermieden.

Meta-logische Evaluierung von Zielen ist k
siver und kann in einigen Fällen die Perfo
DTMS überschatten. Einsatzbereiche des DT
daher vorwiegend in den Bereichen, in dene
für das Übermitteln von Nachrichten groß
der Rechenzeit des DTMS ist. Dies sind z.B
Anwendungen, die auf mail-basierten Proto
setzen (Gruppenkalender, Task Manager, ...)

Das DTMS beschränkt die Autonomität in e
tigen Fall: Nur ein Agent darf zu einer bestim
aktiv sein, da z.Z. keine Synchronisationsme
vorhanden sind. Dies kann Gegenstand späte
ten sein.

Danksagung

Ich bedanke mich beim Projekt KIK des
I3.CSCW der GMD, die mir das Erarbeiten
satzes ermöglichten. Besonderen Dank sage
ner und A. Burt, die zahlreiche Versionen d
deliegenden Originalfassung lasen und komm

Literatur

[1] D. M. Bridgeland und M. N. Huhns. I
Truth Maintenance. In *Proc. of AAAI*
72–77, Boston, MA, 1990.

[2] S. Costantini und G. A. Lanzarone. Metalevel Representation of Analogical Inference. In E. Ardizzone, S. Gaglio und F. Sorbello, Hrsg., *Trends in Artificial Intelligence: Proc. of the 2nd Congress of the Italian Association for Artificial Intelligence, AI*IA*, Seiten 460–464. Springer, Berlin, Heidelberg, 1991.

[3] Edmund H. Durfee, Victor R. Lesser und Daniel D. Corkill. Trends in Cooperative Distributed Problem Solving. In *Transactions on Knowledge and Data Engineering*, 1989.

[4] J. R. Galliers. The Positive Role of Conflict in Cooperative Multi-Agent Systems. In Y. Demazeau und J.-P. Müller, Hrsg., *Decentralized A.I. : Proc. of the First European Workshop on Modelling Autonomous Agents in a Multi-Agent World, Cambridge, England*, Seiten 33–46. North-Holland, Amsterdam, 1990.

[5] J. R. Galliers. Cooperative Interaction as Strategic Belief Revision. In S. M. Deen, Hrsg., *CKBS'90: Proc. of the International Working Conference on Cooperating Knowledge Based Systems*, Seiten 148–163. Springer, Berlin, Heidelberg, 1991.

[6] Thilo C. Horstmann. Distributed Truth Maintenance. Technischer Bericht D-91-11, German Center for Artificial Intelligence, Kaiserslautern, 1991.

[7] John W. Lloyd. *Foundations of Logic Programming.* Symbolic computation: Artificial Intelligence. Springer, 2nd Auflage, 1987.

[8] D. Mahling, T. Horstmann, A. Scheller-Houy, A. Lux, D. Steiner und H. Haugeneder. Wissensbasierte Unterstützung von Gruppenarbeit oder: Die Emanzipation der maschinellen Agenten. In J. Friedrich und K. H. Roediger, Hrsg., *German Chapter of the ACM: Computergestützte Gruppenarbeit (CSCW)*, Band 34, Seiten 297–294. B. G. Teubner, Stuttgart, 1991. (in German).

[9] D. Russinoff. An Algorithm for Truth Maintenance. Technischer Bericht AI-062-85, MCC, 1985.

[10] Donald Steiner, Dirk Mahling und Hans Haugeneder. Human Computer Cooperative Work. In M. Huhns, Hrsg., *Proc. of the 10th International Workshop on Distributed Artificial Intelligence*, 1990.

GroupScheduling

Modellgestützte Methoden und Werkzeuge für das kooperative Scheduling beim betrieblichen Ressourcenmanagement

Leena Suhl
Technische Universität Berlin
Wirtschaftsinformatik / AEDV, Sekr. FR 5-5
Franklinstr. 28/29
W-1000 Berlin 10

In kooperativen betrieblichen Scheduling-Prozessen wird die zeitliche Zuordnung von Re sourcen zu Aktivitäten von einer Gruppe von Planern in Zusammenarbeit bestimmt. Jed Planer ist für einen Teilplan zuständig; alle greifen aber auf gemeinsame Ressourcen z so daß die Teilpläne koordiniert werden müssen. Die Organisation solcher Prozesse wir am Beispiel der Flug- und Kapazitätsplanung in Linienfluggesellschaften betrachtet. Ve schiedene Lösungsansätze im prototypischen System GroupScheduler werden diskutiert.

1 Einführung

Die Aufgabe der *Zeitwirtschaft* eines Unternehmens ist es, die vorhandenen Betriebsmitte möglichst wirtschaftlich einzusetzen. In einem Scheduling-Prozeß werden gewünscht Aktivitäten zu den betrieblichen Ressourcen unter Einhaltung aller gegebenen Restriktio nen zeitlich zugeordnet. Ein komplexer Scheduling-Prozeß erfordert die Beteiligung meh rerer Planer (Produktmanager, Ressourcenmanager, Kurz- und Langfristplaner usw.)

Das Forschungsprojekt **GroupScheduling** an der TU Berlin wird von der DFG in Schwerpunktprogramm "Verteilte DV-Systeme in der Betriebswirtschaft" gefördert Gegenstand des Projektes sind kooperative Scheduling-Prozesse beim betrieblichen Res sourcenmanagement. Insbesondere werden solche Prozesse betrachtet, in den eine Grob planung schrittweise verfeinert wird, und die Plandaten erst im Laufe des Planungsprozes ses genau spezifiziert werden. Die Planer sind Menschen, die über ein Rechnernetz kom munizieren. Alle Entscheidungen werden von Menschen getroffen, aber Vorschläge Alternativen und what-if-Analysen werden mit modellbasierten Planungsmethoden er zeugt.

Das Konzept einer *Scheduling-Workbench* hat sich bei der Lösung von semistrukturierten betrieblichen Scheduling-Problemen bewährt. Ein Planer kann jederzeit zwischen einer globalen, algorithmischen Planung, einer inkrementellen, interaktiven Planung und de

Simulation wählen. Vom System werden Vorschläge zum Erreichen der gegebenen Ziele generiert sowie Konsistenzprüfungen vorgenommen.

Im Forschungsprojekt wird das Workbench-Konzept auf verteilte Systeme erweitert, so daß die Planer zeitlich und räumlich getrennt arbeiten können. Die Einzelplatz-Workbench wird durch Konzepte zur Koordinierung der Arbeit und Lösung von Konflikten ergänzt. Der globale Planungszustand eines Schedules wird zentral gespeichert. Zusätzlich hat jeder Planer eine private Datenbasis sowie eine Workbench für die interaktive Planung, what-if-Analysen usw. Änderungen in der zentralen Datenbasis werden von den betroffenen Planern genehmigt. Konfliktsituationen entstehen, wenn mehrere Planer zum gleichen Zeitpunkt dieselbe Ressource anfordern.

Die Beteiligung von menschlichen Planern ist notwendig, weil nicht alle Komponenten semistrukturierter Scheduling-Probleme formalisiert werden können. Besondere Anforderungen an gruppenorientierte Scheduling-Unterstützungssysteme entstehen dadurch, daß die menschliche Arbeit nicht eingeschränkt werden darf, aber formale Modelle jederzeit zur Verfügung stehen müssen.

2 Der Flugplanungsprozess

Alle Scheduling-probleme bestehen aus vier Basiskomponenten: Aktivitäten, Ressourcen, Restriktionen und Ziele. Die Aktivitäten werden zu Ressourcen zugeordnet, so daß alle gegebenen Restriktionen eingehalten werden, und die Ziele möglichst gut erreicht sind. Unterschiedliche betriebliche Scheduling-Prozesse gibt es bei der Produktionsplanung, Personaleinsatzplanung, Stundenplanung usw. In diesem Vortrag wird als Beispiel der Prozeß der Flug- und Kapazitätsplanung einer Linienfluggesellschaft diskutiert [FraSu91, Su93]. Die vorgeschlagenen Ansätze sind zum großen Teil auch auf andere Scheduling-Prozesse übertragbar.

Das Produkt einer Linienfluggesellschaft, der *Flug*, ist ein verbindliches Transportleistungsangebot zu einem gegebenen Zeitpunkt. Ein Flug ist determiniert durch Zeit und Ort des Abflugs und der Ankunft. Jeder Flug wird mit einer Flugnummer gekennzeichnet und kann mehrere Teilstrecken (Legs) und Zwischenlandungen beinhalten. Je nach erwartetem Sitz- und Frachtplatzbedarf sowie benötigter Reichweite werden die Flüge dem jeweils günstigen Flugzeugtyp (Muster) zugeordnet. Die Flüge einer Planungsperiode (meistens ein halbes Jahr: Sommer oder Winter) werden in einem *Flugplan* festgelegt.

In einer Fluggesellschaft wird an einem Flugplan über mehrere Jahre gearbeitet, so daß das Ergebnis immer genauer wird. Der Prozeß kann in vier Phasen gegliedert werden: Produktplanung (was produziert wird), Produktionsplanung (wann und wieviel produziert

wird), die Zuordnung von Ressourcen zu der geplanten Produktion und zuletzt die operationale Umsetzung.

In Abb. 1 ist der Planungsprozeß schematisch dargestellt, in dem die Festlegung von Plandaten (oben in der Abbildung) den ersten Schritt bildet. In den ersten vier Schritten des Planungsprozesses werden langfristige Planungsprobleme mit einem Zeithorizont von mehreren Jahren behandelt. Nach diesen Schritten sind die verfügbaren Verbindungen und Kapazitäten festgelegt.

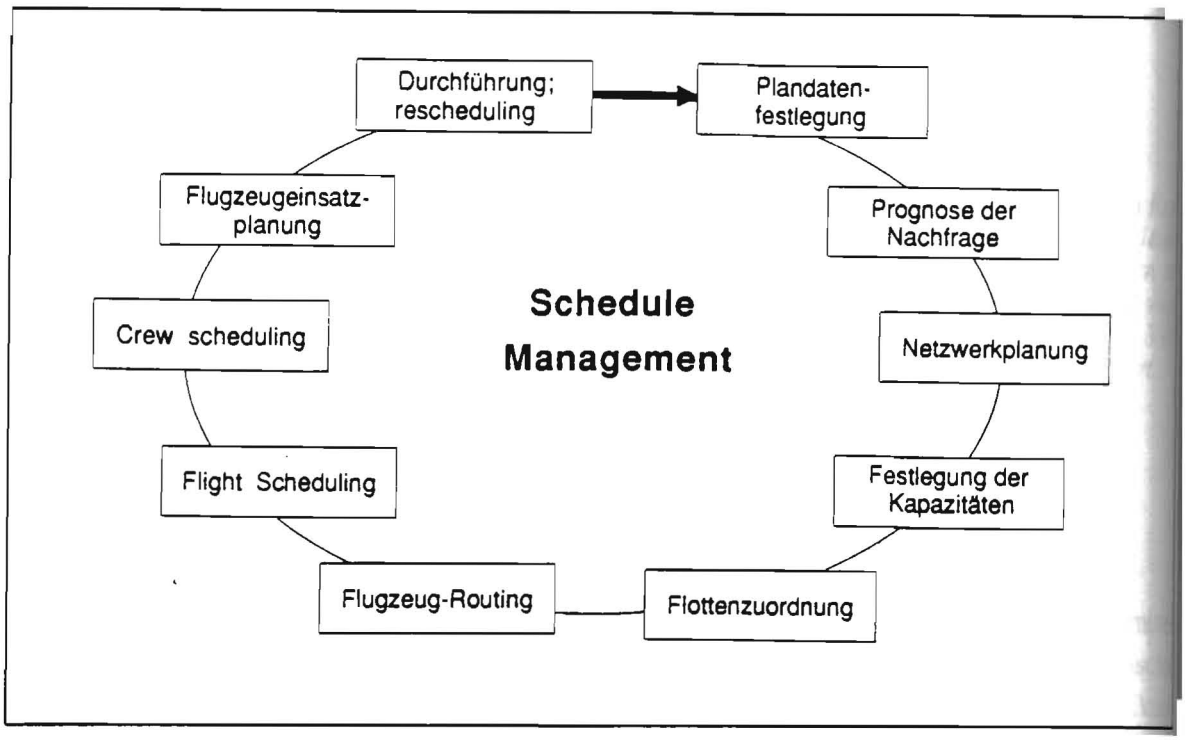


Abbildung 1: Der Planungsprozeß bei Fluggesellschaften

Als Basis für die Planung werden für Plandaten wie Flugzeiten, Mindestbodenzeiten und Wartungshäufigkeiten Erfahrungswerte festgelegt. Aufgrund einer Marktanalyse und Nachfrageprognose werden die angebotenen Verbindungen als Abflug-Ankunft-Paare (Origin-Destination-Paare) definiert. Daraus ergibt sich eine Netzwerkstruktur, die oft eine Stern-Struktur aufweist. Die Produktionskapazitäten, vor allem die Anzahl der Flugzeuge bestimmten Typs, müssen mehrere Jahre im voraus festgelegt werden. Die angebotenen Kapazitäten pro Verbindung werden durch die Flughäufigkeit und Passagierzahl festgelegt.

In den nächsten Schritten Flottenzuordnung, Flugzeug-Routing und Flug-Scheduling wird das Produktionsprogramm festgelegt. Als Ergebnis dieser Schritte wird der Flugplan generiert, veröffentlicht und potentiellen Kunden angeboten.

Eine Linienfluggesellschaft verfügt normalerweise über mehrere Flotten (z.B. Douglas DC10, Boeing 747, Airbus 320 usw.), die unterschiedliche Passagierkapazitäten, Kraftstoffverbräuche und Reichweiten haben. Bei der Flottenzuordnung (fleet assignment) wird jedem Flug ein passender Flugzeugtyp zugeordnet. Unter dem Flug-Scheduling-Problem ist die Festlegung von Abflug- und Ankunftzeiten für jeden Flug gemeint. Beim Flugzeug-Routing werden für einzelne Flugzeuge *Umläufe* (auch *Rotationen* genannt) generiert. In einem *Rotationsplan* werden die Umläufe einzelner Flugzeuge als zeitliche Sequenzen dargestellt. Ein Umlauf dauert meistens eine Woche und wird periodisch durchgeführt. Die Kombination aus der Flottenzuordnung, dem Flugzeug-Routing und dem Flug-Scheduling wird *Flugplanungsproblem* genannt. Durch die Lösung des Flugplanungsproblems wird die benötigte Anzahl von Flugzeugen zur Durchführung der vorgeschlagenen Flüge bestimmt. Wird diese Anzahl als Grundlage zur Kapazitätsplanung benutzt, wird ein kombiniertes *Kapazitäts- und Flugplanungsproblem* gelöst.

Im Flugplanungsproblem werden die Flugzeug-Umläufe noch nicht zu physischen Flugzeugen zugeordnet. Für die Umsetzung eines Flugplans werden außer Flugzeuge weitere Ressourcen wie Besatzungen und Abfertigungskapazitäten an Flughäfen benötigt. Diese Ressourcen werden ein bis zwei Monate vor der Durchführung von Flügen zugeordnet. Die wichtigsten Schritte der Ressourcenzuordnung sind die Besatzungseinsatzplanung (crew scheduling) und die Flugzeugeinsatzplanung (physical aircraft assignment). Bei der *Besatzungseinsatzplanung* werden jedem Flug das benötigte Cockpit- und Flugbegleitungspersonal zugeordnet. Die Besatzungseinsatzplanung ist vielfältigen Restriktionen bzgl. Arbeitszeiten, Ruhezeiten und Tarifregeln unterordnet. Bei der *Flugzeugeinsatzplanung* wird ein physisches Flugzeug jedem geplanten Umlauf zugeordnet. In dieser Phase werden auch Wartungsereignisse für Flugzeuge eingeplant.

Bei der Umsetzung eines Flugplanes werden ständig Änderungen aufgrund von Wetter- und Verkehrsverhältnissen sowie technischen Problemen vorgenommen. Der Flugplan wird dabei dynamisch modifiziert. Am Flughafen wird auch kurzfristig ein Flugsteig (gate) sowie das Abfertigungspersonal jedem Flug zugeordnet.

3 Das Kapazitäts- und Flugplanungsproblem

In diesem Abschnitt wird das Kapazitäts- und Flugplanungsproblem der Linienfluggesellschaften genauer betrachtet. Die Flugwünsche werden von Produkt-Managern in Form eines Sollprogramms generiert (Abb. 2). Aus den Wünschen wird ein realisierbarer Flugplan aufgestellt; d.h. für eine zu bestimmende Flotte wird ein konsistenter Arbeitsplan erstellt, der alle Wünsche abdeckt. Dabei sind vielfältige interne und externe Restriktionen einzuhalten: Verfügbarkeit von Besatzungen, Wartungs- und Abfertigungskapazitäten, Start- und Landeerlaubnisse auf den Flughäfen, Öffnungszeiten, Lärmbestimmungen usw. Ziel-

setzung ist die Wirtschaftlichkeit des Planes, die durch das Subziel der Flottenminimierung repräsentiert wird.

o	Flug FLI 414	FRA YMX PHL Abflug montags, mittwochs und freitags zwischen 10 und 12 Uhr mit einer DC10
o	Flug FLI6203	TXL FRA Abflug täglich 7:00 mit einem Airbus A300-600
o	Flug FLI 736	FRA HKG Abflug mittwochs und sonntags egal welche Uhrzeit mit einer Boeing 747-400.

Abbildung 2: Eingabe als Sollprogramm

Zu jeder Flugstrecke ist eine zu erwartende *Blockzeit* für jedes in Frage kommende Flugzeugmuster gegeben. Die Blockzeit ist die Zeitdauer eines Legs vom Verlassen des Abfluggates bis zum Stillstand am Ankunftgate. An jedem Flughafen gibt es eine musterabhängige minimale *Bodenzeit* (Abfertigungszeit). Bei der Flug- und Kapazitätsplanung werden die Wünsche für jedes Muster zu Umläufen der einzelnen Flugzeuge zusammengesetzt. Die Umläufe sollen so generiert werden, daß die Flottengröße minimal ist. Zu jedem Flug gehört eine wohldefinierte Menge von möglichen Nachfolgern: Flüge, die von dem gegebenen Ankunftort abfliegen und deren Abflugzeit größer oder gleich ist als die erstmögliche Ankunftzeit plus minimale Bodenzeit.

Ergebnisse der Kapazitäts- und Flugplanung sind die benötigte Flottenstruktur, der Flugplan und der Rotationsplan, die als Grundlage für die weiteren Planungsstufen dienen. Die Bedürfnisse der nachgelagerten Ressourcenplanung werden durch bewußt eingeplante Puffer berücksichtigt oder in einem interaktiven Koordinationsprozeß angeglichen. Die Rotationen werden für eine Woche generiert und wiederholen sich während der Flugplanperiode (Abb. 3). Eine Bedingung ist, daß die Sollprogramme für einzelne Flotten balanciert sind, d.h. an jedem Flughafen ist die Anzahl der ankommenden Flüge gleich der Anzahl der abfliegenden Flüge.

```
Flight schedule
Time period 1992/05/11 - 1992/05/17
Pattern(s) D1C
Number of rotations 10

Flights with the same number bound together
   ...   ...   ...

Rotation 1

FLI001   FRA   SJU   1.  6:35   1. 17:30   1. 19:00
FLI004   SJU   FRA   1. 21.10   2.  6:00   2.  8:30
FLI018   FRA   IAD   2. 10:25   2. 19:10   2. 20:00
FLI019   IAD   FRA   2. 22:20   3.  6:00   3.  8:30
FLI006   FRA   CCS   3. 10:25   3. 20:25   3. 21:25
FLI002   CCS   FRA   3. 22:30   4.  8:05   4. 10:35
FLI022   FRA   YVR   4. 10:35   4. 20:50   4. 22:20
FLI023   YVR   FRA   4. 22:20   5.  8:05   5. 10:35
FLI018   FRA   IAD   5. 10:35   5. 19:20   5. 20:10
FLI019   IAD   FRA   5. 22:20   6.  6:00   6.  8:30
FLI005   FRA   SJU   6. 10:25   6. 21:30   6. 23:00
FLI003   SJU   FRA   6. 23:15   7.  8:05   7. 10:35
FLI022   FRA   YVR   7. 10:35   7. 20:50   7. 22:20
FLI023   YVR   FRA   7. 22:20   8.  8:05   8. 10:35
   ...   ...   ...
```

Abbildung 3: Ausschnitt aus einem Rotationsplan

4 Anforderungen an ein Scheduling-Unterstützungssystem

Die Kapazitäts- und Flugplanung ist ein typisches Beispiel eines *semistrukturierten, verteilten* Scheduling-Problems. Im ersten Ansatz scheint das Problem klar strukturiert und einer algorithmischen Lösung zugänglich zu sein. Die Arbeitsweise der Planungsexperten zeigt jedoch, daß das Problem viel komplexer ist. Ein Experte kann schnell erkennen, an welcher Stelle er durch Eingriffe in die Problemstellung eine Verbesserung der Lösung erzielen kann. Blockzeiten, Bodenzeiten, Wartungszeiten usw. sind selbst Plandaten, die auf statistisch ermittelten Standards bzgl. Arbeitszeit, Fluggeschwindigkeit usw. basieren. Durch eine Vereinbarung, schneller zu fliegen oder zu arbeiten, können diese Größen partiell verändert werden. Manchmal lassen sich Slots mit anderen Fluggesellschaften tauschen, auch wenn die Flughafenkapazität erschöpft ist. Bei solchen Entscheidungen sind die Kenntnisse des Planungsexperten über den individuellen Fall notwendig.

Das Expertenwissen des Planers liegt darin zu wissen, wann und unter welchen Bedingungen er Eingriffe in die Problemstruktur und in den Lösungsablauf vornehmen kann. Der Planer ist aber kapazitätsmäßig nicht in der Lage, Pläne regelmäßig von Grund auf neu zu

gestalten. Der Planungsprozeß ist ein permanenter inkrementeller Prozeß, bei dem laufend alle von den Produktmanagern aufgrund der aktuellen Marktsituation gewünschten Änderungen eingearbeitet werden. Ein entscheidungsunterstützendes System zur Kapazitäts- und Flugplanung sollte dem Anwender beide Möglichkeiten bieten: Eine interaktive Umgebung zur inkrementellen Planung sowie die Möglichkeit, Optimierungsalgorithmen für wohldefinierte Subprobleme integriert zu benutzen, und Ressourcenbenutzung mit anderen Planern abzustimmen.

Im folgenden werden die wichtigsten Bedingungen aufgestellt, die ein Flug- und Kapazitätsplanungssystem im Idealfall erfüllen sollte [Su93].

Inkrementelle Planung

Oft beruht ein neuer Flugplan auf einem alten Plan, so daß nur kleinere Änderungen vorgenommen werden müssen. Ein erfahrener Flugplaner kann solche Änderungen im Rahmen einer *inkrementellen* Planung problemlos bewältigen. Basisoperationen der inkrementellen Planung sind Einfügen, Löschen und Bewegen von Flügen und Wartungsereignissen sowie Vertauschen von Umlaufabschnitten. Bei der inkrementellen Planung ist der Anwender für alle Planänderungen zuständig; die Konsistenz (Einhaltung von allen Restriktionen) soll vom System nach jeder Änderung automatisch überprüft werden. In einer Zwischenstufe der Planung muß es aber möglich sein, auch inkonsistente Systemzustände zu bearbeiten und zu speichern.

Von großer Bedeutung bei der interaktiven Planung sind vom System generierte *Vorschläge*, die dazu dienen, die vom Benutzer gegebene Ziele besser zu erreichen. Z.B. kann das System die Verkürzung eines Bodenereignisses vorschlagen, wenn dadurch ein Flugzeug in der Lösung gespart würde. Der Anwender soll aufgrund seiner praktischen Erfahrung entscheiden, ob eine solche Abweichung von den Plandaten tatsächlich vorgenommen werden soll.

Globale Planung

Wenn ein völlig neuer Plan generiert werden soll oder größere Änderungen eines existierenden Planes notwendig sind, kann die Problemkomplexität den Planer im interaktiven Modus überfordern. In solchen Situationen kann der Flugplaner eine Anfangslösung durch eine *globale Planungsmethode* erzeugen.

Eine globale Planungsmethode berechnet einen Flug- und Rotationsplan auf der Basis eines gegebenen Sollprogramms. Dabei werden optimierende Algorithmen der mathematischen Programmierung oder suboptimierende heuristische Algorithmen benutzt. Weil semistrukturierte Planungsprobleme sich nicht vollständig formalisieren lassen, ist es

wichtig, daß ein neu generierter Flugplan nachträglich im interaktiven Modus geändert (editiert) werden kann. Daher sollten die benutzten Algorithmen zur globalen Planung mit der Komponente zur inkrementellen Planung integriert werden, so daß ein nahtloser Übergang ermöglicht wird.

Eine weitere Anwendung der globalen Planung sind what-if-Analysen: Der Flugplaner möchte schnell testen, welche Auswirkungen eine Änderung der Plandaten hat. Es muß möglich sein, alternative Pläne als Lösungen von what-if-Analysen im System zu verwalten.

Simulation

Bei der operationalen Umsetzung eines Flugplanes werden die erzeugten Planungsdaten fast nie exakt eingehalten. Wenn eine experimentelle Verteilung von Planungsdaten wie Blockzeiten, Bodenzeiten und Wartungszeiten im System enthalten ist, kann die Durchführung eines Flugplanes simuliert werden. Diskrete Simulation wird als eine letzte Stufe vor der Veröffentlichung eines Planes benutzt. Strategien zur Konfliktlösung bei einem Ausfall oder einer Verspätung eines Fluges werden dabei getestet, um mögliche Engpässe erkennen zu können.

Individuelle Arbeitsgestaltung

Computerbasierte Unterstützung von Planungsaufgaben soll einem Anwender bei der Arbeit helfen und ihn nicht einschränken. Aus der Praxis ist bekannt, daß individuelle Flugplaner ihre Arbeit nach sehr unterschiedlichen Arbeitsorganisationen und Abläufen gestalten. Ein Plan kann in kleinen Schritten nach der gedanklichen Vorstellung eines Planers konstruiert werden. Er kann aber auch als Ergebnis nach mehreren what-if-Analysen global generiert werden. Dem einzelnen Planer soll freigestellt werden, wie er mit Hilfe des Systems zu den gewünschten Ergebnissen gelangt.

Unterstützung der Gruppenarbeit

In einer größeren Fluggesellschaft sind mehrere Planer bei der Erstellung eines Flugplans beteiligt. Jeder Planer hat einen bestimmten Kompetenzbereich, z.B. Langfrist- oder Kurzfristplanung bzw. Langstrecken- oder Kurzstreckenplanung eines geographischen Zielgebiets. Alle greifen jedoch auf gemeinsame Ressourcen (Flugzeuge) zu. Daher muß die Gruppenarbeit koordiniert werden. In einem System mit mehreren Planern gibt es gemeinsame Daten sowie individuelle Planungsdaten jedes Planers. Die Modellierung solcher Prozesse in verteilten entscheidungsunterstützenden Systemen ist eine Aufgabe des Forschungsprojektes GroupScheduling.

Graphische Benutzungsoberfläche

Die Gestaltung einer graphischen Benutzungsoberfläche ist für die interaktive Planung besonders wichtig. Ein computergestütztes Planungssystem sollte mehrere Darstellungen eines Planes anbieten. Eine wichtige Darstellungsform ist ein Balkendiagramm, in dem mehrere Auswahl- und Kombinationsmöglichkeiten angeboten werden. Eine weitere Darstellungsart ist ein Verbindungsdiagramm, wo die Ereignisse an einem Flughafen betrachtet werden. Wenn mehrere Darstellungen gleichzeitig möglich sind - in einem fensterorientierten System oder auf mehreren Bildschirmen - sorgt ein integriertes Datenbank- und Dialogverwaltungssystem für die Konsistenz aller Darstellungen. Eine interaktive Änderung in einer Darstellung wird unmittelbar in allen anderen Darstellungen vorgenommen.

Das Workbench-Konzept

Ein Software-System als *Workbench* bietet eine Umgebung zur rechnergestützten Konstruktion von Zeitplänen (Schedules). Unter einer Workbench (Werkbank) versteht man einen Arbeitstisch, auf dem mehrere Werkzeuge eines Handwerkers zur Verfügung stehen. Am Arbeitstisch werden verschiedene Objekte mit Hilfe der Werkzeuge bearbeitet. Analog wird ein Flugplan von einem professionellen Planer mit Hilfe rechnergestützter Werkzeuge bearbeitet. Als Werkzeuge können u.a. ein graphischer Flugplan-Editor, mathematische Optimierungsalgorithmen, heuristische Methoden oder eine Wissenbasis mit einer Inferenzkomponente angeboten werden.

Die Einführung des Workbench-Konzeptes unterstützt auch die Arbeit des Systementwicklers, weil im System eine einheitliche Datenbasis und Benutzeroberfläche realisiert werden. Eine offene Schnittstelle ermöglicht die Integration verschiedener Algorithmen und Simulationsmethoden. Wenn die Basiskomponenten von Scheduling-Problemen objektorientiert modelliert sind, wird ein großer Teil der Eigenschaften und Attribute einzelner Objekte in jeder neuen Anwendung vererbt.

5 Mögliche Ansätze aus der VKI

In Konzepten und Systemen der Verteilten Künstlichen Intelligenz (VKI) sind mehrere Metaphere aus dem täglichen Leben abgebildet worden. Im schon klassischen Artikel von Davis and Smith über Contract Nets [DavSmi83] wurden Verhandlungen als ein Metapher für das verteilte Problemlösen diskutiert. Die Ausgangsbasis war es, menschliche Handlungsweisen in verteilte KI-Systeme zu übertragen. Analog wurde das Verhalten einer

Scientific Community als ein Metapher für ein verteiltes, paralleles System benutzt [KorHew81].

Nach zehn Jahren Entwicklung im Bereich der VKI können wir die Frage andersrum stellen: Welche Architekturen von VKI-Systemen können bei der gruppenorientierten Arbeit von Menschen benutzt werden? Es ist denkbar, daß die Planer als autonome Agenten mit eigenen Zielen funktionieren, wobei der Konsens durch Verhandlungen (wie im Contract Net) erreicht wird. Wenn die Entscheidungsfindung eher hierarchisch organisiert ist, wird der Planungsprozess durch eine zentrale Kontroll- und Entscheidungskomponente modelliert. Viele Mischformen sind natürlich auch möglich.

Die folgenden aus der Literatur bekannten Ansätze könnten bei der Modellierung kooperativer betrieblicher Scheduling-Prozesse in Frage kommen. Insbesondere wird die Eignung zur kooperativen Lösung des Kapazitäts- und Flugplanungsproblems diskutiert.

Contract Net

Ein Auftraggeber (contractor) bewertet Angebote (bids) potentieller Auftragnehmer bezüglich gegebener Kriterien [Smi80, DavSmi83]. Agenten, deren Angebote akzeptabel sind, werden daraufhin vom Auftraggeber zur Bearbeitung ihrer Teilaufgabe verpflichtet. Bei der Flugplanung könnte die Anforderung von Ressourcen (Flugzeugen, Besatzungen, Wartungskapazitäten, ...) in Form von Aufträgen erfolgen. Einzelne Ressourcen würden als Auftragnehmer modelliert. Dieser Ansatz ist in GroupScheduling (noch) nicht gefolgt. Ein potentieller Problem dabei wäre, daß außer der Sichten jedes einzelnen Planers (Agenten) eine Gesamtsicht bei der Konfliktlösung und algorithmischen Lösung erforderlich ist.

Blackboard

Mit Blackboards verbindet man generell eine opportunistische, auf weitgehender Autonomie basierende Arbeitsweise. Mit dem Blackboardansatz kann jedoch generell jede Form der Kooperation modelliert werden [Hay85]. Ziele, Aufgaben und Ergebnisse werden kontinuierlich von den Agenten auf dem Blackboard veröffentlicht und dienen allen Agenten zur Bestimmung ihrer weiteren Aktivitäten. Bei der Flugplanung würde man alle Ressourcenkonflikte zu einem Zeitpunkt auf dem Blackboard darstellen. Weil alle Konflikte von einem einzelnen Planer wegen der eingeschränkten Sicht nicht erkannt werden können, ist in diesem Ansatz ein zentraler Koordinator erforderlich, der auf Konfliktsituationen zwischen den einzelnen Planern prüft.

Functionally Accurate, Cooperative Paradigm (FA/C)

Das Paradigma der funktional akkuraten, kooperativen Systeme wurde für den Fall vorgeschlagen, in dem mehrere Teilnehmer (Agenten) auch unvollständige, miteinander inkonsistente Informationen austauschen können [LesCor81]. Das heißt, das das Gesamtsystem sich nicht immer in einem konsistenten Zustand befindet, aber das Teilsystem eines Agenten konsistent bezüglich seiner Eingabe und Ausgabe ist.

Die traditionellen Ansätze der Computerunterstützung bei der Flugplanung gingen davon aus, daß das gespeicherte Gesamtsystem in einem konsistenten Zustand ist. Dies bedeutete, daß Ressourcenkonflikte außerhalb der Rechnerunterstützung gelöst werden mußten. Analog zu dem FA/C Paradigma ist es jedoch sinnvoll, inkonsistente Zustände des Gesamtflugplanes während der Planung temporär zuzulassen. Dadurch können computergestützte Methoden bei der Suche nach einer konsistenten, akzeptablen Gesamtlösung eingesetzt werden.

Multistage Negotiation

Das Konzept Multistage Negotiation wurde ursprünglich zur verteilten Überwachung von Kommunikationsnetzwerken vorgeschlagen [ConMeyLes86]. Die Überwachung eines Netzes findet in Knoten statt, die für einen bestimmten Teilbereich des Netzes zuständig sind. Kein Knoten hat den Überblick über das Gesamtnetz. In einem Störungsfall werden Ziele, Intentionen und Lösungsalternativen ausgetauscht. In einem Verhandlungsprozess wird nach einer konsistenten Gesamtlösung gesucht, so daß die gestörten Verbindungen umgegangen werden.

Analog zu Multistage Negotiation gibt es auch im Flugplanungsprozess lokale Ziele der einzelnen Planer, die in einem Verhandlungsprozess zu einer konsistenten, von allen akzeptierten Lösung zusammengeführt werden müssen. Weil eine Änderung eines bestimmten Planers aufgrund der gemeinsamen Ressourcennutzung weitgehende Folgen im Gesamtplan haben kann, wird eine zentrale Kontrollkomponente wahrscheinlich sinnvoll sein.

Partial Global Planning

Durch Partial Global Plans (PGP's) können mehrere Arten der kooperativen Problemlösung modelliert werden [DurLes87]. Kommunikation bei der Problemlösung kann durch Verhandlungen, Tauschen von Teillösungen und gegenseitige Hilfen erfolgen. In jedem Knoten eines verteilten Planungssystems wird ein Knotenplan (node plan) definiert. Im Knotenplan sind die lokalen Ziele eines Knotens sowie die vom Knoten geplanten Aktivitäten und deren Dauer kurz zusammengefaßt. Somit enthält ein Knotenplan nur die not-

wendigen Informationen und kann preiswert im Netz verteilt werden. Aus einer Teilmenge der Knotenpläne, deren lokale Ziele kompatible sind, werden Partial Global Plans generiert. Ein Partial Global Plan ist global in dem Sinne, daß alle Ziele und Aktivitäten der beteiligten Knoten berücksichtigt sind, aber partiell in dem Sinne, daß nicht alle Knoten beteiligt sind.

Es wäre im Prinzip denkbar, bei der Flugplanung aus den einzelnen Teilplänen Partial Global Plans für solche Teile zu generieren, die wenig Ressourcenkonflikte enthalten. Das größere Problem, wie die einzelnen PGP's zu einem konsistenten Gesamtplan zusammengefügt werden, bleibt jedoch ungelöst.

Constraint-directed Negotiation

Das Prinzip der Constraint-directed Negotiation stellt einen automatisierten Verhandlungsprozeß zwischen mehreren Agenten dar [SatFox89]. Die gewünschten Aktivitäten jedes Agenten werden in Form von Constraints mit den benötigten Ressourcen repräsentiert. In einem Konfliktfall werden Verhandlungen durchgeführt, die zu einer Änderung der einzelnen Constraints, Aktivitäten oder Ziele führen.

Weil die Ressourcenzuordnung als Fragestellung sehr nahe der Flugplanungsproblematik ist, können viele Aspekte der Constraint-directed Negotiation übernommen werden. Es ist jedoch zu beachten, daß das Flugplanungsproblem nicht völlig formalisierbar und automatisierbar ist, so daß die eigentliche Entscheidungsfindung der menschlichen Planer nicht eingeschränkt werden darf.

Relationship Driven Plan Coordination

Die Plankoordination, die auf Beziehungen zwischen Plänen basiert, geht von zwei Beziehungstypen aus [Mar91]. Die Beziehung zwischen zwei Teilplänen ist *negativ*, wenn die Pläne sich gegenseitig bei der Umsetzung beschränken. Zum Beispiel fordern zwei Pläne gleichzeitig dieselbe Ressource an. Die Beziehung ist *positiv*, wenn die gemeinsame Durchführung beider Pläne einem oder beiden Agenten einen Vorteil bringt. Zum Beispiel eine Aktivität, die von beiden Agenten vorgeplant war, muß nur einmal durchgeführt werden. Natürlich gibt es auch Teilpläne, die gegenseitig neutral sind, also keine positiven oder negativen Auswirkungen haben.

Die einzelnen Teilpläne werden an alle anderen Planer übersendet und nach Beziehungsarten überprüft. Die negativen Beziehungen werden soweit wie möglich in einem Verhandlungsprozeß beseitigt. Dieser Prozeß kann mit Hilfe eines Koordinationsagenten oder eines Blackboards gesteuert werden.

Beim Flugplanungsprozeß wird die Untersuchung der Beziehungen einzelner Teilpläne vor dem Anfang des Verhandlungsprozesses von einer großen Hilfe sein.

5 Erste Lösungsansätze in GroupScheduler

Im Forschungsprojekt GroupScheduling wird von zwei Seiten vorgegangen: Auf der einen Seite werden alternative Konzepte zur Organisation des kooperativen Scheduling entwickelt und getestet. Auf der anderen Seite wurde die Entwicklung eines prototypischen, modularen Systems GroupScheduler früh angefangen. Aus den Komponenten des GroupScheduler können Anwendungssysteme für unterschiedliche Problem- und Organisationsstrukturen zusammengestellt werden.

Ein wesentliches Merkmal der im Forschungsprojekt betrachteten kooperativen Scheduling-Probleme ist, daß jeder Planer für einen Teilplan zuständig ist, muß aber die Benutzung von gemeinsamen Ressourcen mit anderen Planern abstimmen. Wie hoch die Motivation zur Zusammenarbeit ist, hängt von der Organisationsstruktur ab. Wenn jeder Planer als ein Profit Center funktioniert, so daß das finanzielle Ergebnis von der Güte des Planes abhängt, ist er kaum motiviert, bei seiner Zielsetzung nachzugeben. In dem Fall ein Koordinator mit Entscheidungsbefugnissen unvermeidbar sein. Wenn aber der Wunsch nach einem konsistenten Schedule im Vordergrund steht, und durch das Nachgeben bei lokalen Zielen keine Nachteile entstehen, wird das System viel eher auch demokratisch funktionieren.

In einer Literaturstudie wurde für diese Art von Systemen keine Lösung gefunden, die direkt übernommen werden kann. Viele Komponenten der existierenden Ansätze können jedoch sinnvoll eingesetzt werden. Innerhalb des Systems GroupScheduler wird es möglich sein, mehrere Organisationstypen zu modellieren, aber im Prinzip läuft der kooperative Schedulingprozeß folgendermaßen ab:

1. In einem Teil-Sollprogramm definiert jeder Planer die gewünschten Aktivitäten, deren Dauer und gewünschte Anfangszeiten bzw. Anfangszeitfenster innerhalb seines Verantwortungsbereichs.

2. Vom Koordinator werden die Teil-Sollprogramme geprüft und mit einem Algorithmus zu einer Anfangslösung zusammengeführt. Diese Lösung kann unzulässig sein, weil z.B. mehrere Ressourcen gleichzeitig angefordert werden als zur Verfügung stehen.

3. Die Anfangslösung wird vom Koordinator nach Inkonsistenzen zwischen den einzelnen Teilplänen geprüft. Jeder Planer wird über die Inkonsistenzen, die seinen

Teilplan betreffen, informiert, und vom System wird, soweit möglich, ein Lösungsvorschlag gemacht.

4. Die Teilnehmer akzeptieren die Systemvorschläge oder lehnen sie ab, und bestätigen ggf. die Planänderungen. Dieser Vorgang wird wiederholt, bis sich nichts mehr ändert oder der gegebene Zeitrahmen überschritten ist.

5. Die Inkonsistenzen, die noch nicht gelöst wurden, werden in menschlichen Verhandlungen zwischen dem Koordinator und den beteiligten Planern einzeln gelöst.

Im diesem Konzept ist ein menschlicher Koordinator vorgesehen, der rechnergestützt arbeitet. Eine andere Variante wäre ein maschineller Koordinator, wobei die menschlichen Planer gemeinsam ohne eine höhere Hierarchieebene zu einer konsistenten Lösung kommen. Weiterhin arbeiten die Planer im vorgeschlagenen System zeitlich und örtlich getrennt, wodurch der Verhandlungsprozeß unnötig lange dauern kann. Eine effizientere Variante bei der Verhandlungsphase könnte die Benutzung eines spezialisierten Sitzungsunterstützungssystems sein, so daß alle Planer in einer Sitzung (oder mehreren Sitzungen) die Konflikte gemeinsam rechnergestützt lösen. Bei Scheduling-Problemen mit dieser Komplexität ist es kaum möglich, daß ein Mensch alle Auswirkungen einer Änderung überblickt. Daher ist die Computerunterstützung der Verhandlungen unvermeidbar.

6 Zusammenfassung

In diesem Vortrag wurde der Einsatz von Methoden der Verteilten Künstlichen Intelligenz beim kooperativen Scheduling von betrieblichen Ressourcen diskutiert. Ideen aus mehreren Ansätzen können übernommen werden, aber kein Ansatz ist direkt einsetzbar. Der größte Grund dafür ist, daß die betrachteten Scheduling-Prozesse nicht völlig formalisierbar sind, wodurch die Beteiligung von menschlichen Planern notwendig ist. Eine prinzipielle Vorgehensweise im prototypischen System GroupScheduler wurde vorgestellt. Bei der Umsetzung dieses Ansatzes sind jedoch noch viele offene Fragen zu lösen.

Literatur

[ConMeyLes86]
Conry S., Meyer R., Lesser V., Multistage negotiations in distributed planning. COINS TR86-87, pp. 1-17, University of Massachusetts, 1986.

[DavSmi83]
Davis R., Smith R.G., Negotiations as a metaphor for distributed problem solving. Artificial Intelligence, Vol. 20, pp. 63-109, 1983.

[DurLes87]
Durfee E., Lesser V., Using partial global plans to coordinate distributed problem solver Proceedings of the 1987 International Joint Conference on Artificial Intelligence, pp. 875 883, 1987.

[Fra91]
Franken R., Designkonzept eines wissensbasierten Planungsunterstützungssystems. ist Intelligente Software-Technologien, Vol. 1, No. 4, 1991.

[FraSu91]
Franken R., Suhl L., Methoden der Planungsunterstützung bei der Produktionsplanung dargestellt am Beispiel der Deutschen Lufthansa AG. Operations Research Proceeding 1991, Springer-Verlag 1992.

[Hay85]
Hayes-Roth B., A blackboard architecture for control. Artificial Intelligence Journal, Vol 26, pp. 251-321, 1985.

[KorHew81]
Kornfeld W., Hewitt C., The scientific community metaphor. IEEE Transactions on Systems, Man and Cybernetics, Vol. 11, No. 1, pp. 24-33, 1981.

[LesCor81]
Lesser V., Corkill D., Functionally accurate, cooperative distributed systems. IEEE Transactions on Systems, Man and Cybernetics, Vol. 11, No. 1, pp. 81-96, 1981.

[Mar90]
Martial F. v., Activity coordination via multiagent and distributed planning. In Brauer W. and Hernández D. (Eds.), Verteilte Künstliche Intelligenz und kooperatives Arbeiten. Informatik-Fachberichte No. 291, Springer-Verlag 1991.

[SatFox89]
Sathi A., Fox M., Constraint-directed negotiation of resource reallocations. In Gasser L., and Huhns M. (Eds.), Distributed Artificial Intelligence Vol. 2, Morgan Kaufmann Publishers, Los Altos, CA, pp. 129-162.

[Smi80]
Smith R., The Contract Net Protocol: High-level communication and control in a distributed problem solver. IEEE Transactions on Computers, Vol. 29, No. 12, pp. 1104-1113, 1980.

[Su93]
Suhl L., Entscheidungsunterstützende Systeme bei der Produktionsplanung in Fluggesellschaften. Erscheint in Wirtschaftsinformatik.

Gründungsworkshop der Fachgruppe 1.1.6 *Verteilte Künstliche Intelligenz* der GI. Saarbrücken, 29.-30. April 1993

CAP II & RAP: Lernfähige, verhandelnde Agenten für die Büroautomatisierung

Siegfried Bocionek*

Siemens AG, ZFE ST SN 33
Otto-Hahn-Ring 6
8000 München 83
bocionek@ztivax.zfe.siemens.de OR bocionek@cs.cmu.edu

Abstract

Agentensysteme für die Büroautomatisierung sind als Erweiterung heutiger Anwendersoftware (Dokumenterstellung, Tabellenalkulation, Abrechnungen, Planung etc.) zu sehen, die nicht nur für einen einzelnen Arbeitsgang eines einzelnen Benutzers da sind, sondern auf die Unterstützung von *gesamten Arbeitsabläufen* zwischen *allen beteiligten Personen* abzielen. Dazu benötigen die Agenten zwei wesentliche "Fähigkeiten": Verhandeln und Lernen. Das Verhandeln – mit anderen Agenten ebenso wie mit menschlichen Rechnerbenutzern – ist notwendig, damit der Agent für seinen Besitzer im kooperativen Arbeitsablauf soweit wie möglich Arbeitsschritte *autonom* ausführen kann. Lernen durch "Beobachten" des Vorgehens seines Besitzers ermöglicht dem Agent, sich mit der Zeit immer besser an dessen "Arbeitsstil" anzupassen und immer mehr Aufgaben übernehmen zu können.

Im vorliegenden Artikel wird beschrieben, welche Mechanismen zum Verhandeln und Lernen in zwei Agenten für typische, i.d.R. sehr zeitaufwendige Alltagsaufgaben im Büro integriert wurden: Terminmanagement und Raumverwaltung. Dabei werden Architektur, Funktionalität und Besonderheiten des Kalenderagenten CAP II sowie des Raumreservierungsagenten RAP im Detail vorgestellt. CAP II und RAP sind Beispiele für DAI Systeme, die der Bürosoftware von morgen zugrunde liegen werden. Wenn jedermann einen "Personal Agent" auf einem "Hand-held Computer" bei sich trägt, dann sind solche Systeme ein *Muß* zur Organisation der eigenen Arbeit sowie zur Kooperation mit Kollegen.

*Zur Zeit Visiting Scientist an der Carnegie Mellon University, School of Computer Science, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA

1 Einleitung

An der Carnegie Mellon University wird unter dem Arbeitstitel *Software Secretary* an Büroautomatisierungssystemen gearbeitet, die den "Büromenschen" bei seiner täglichen Arbeit unterstützen sollen: Beim Vereinbaren von Terminen, beim Verwalten von Ressourcen (wie Besprechungs- oder Vortragsräumen), später auch bei Budgetplanung oder Abrechnungs- und Bestellwesen. Die Idee des *Software Secretary* klingt attraktiv, da eine derartige Software sekretariatsähnliche Leistungen *für jedermann/frau* bereitstellen könnte, der/die heute noch nicht das "Privileg" einer eigenen Sekretärin besitzt.

Drei Eigenschaften bestimmen Erfolg oder Mißerfolg solcher Systeme: Der Grad an *Autonomie*, die *Flexibilität*, mit verschiedenen Benutzern in verschiedenen Umgebungen verschiedene Aufgaben lösen zu können, und die *Benutzerschnittstelle*. Auf die Benutzerschnittstelle wird in diesem Artikel nur soweit eingegangen wie sich die Anzahl der notwendigen Interaktionen oder der Bedienungskomfort durch wachsendes Wissen und Autonomie der Agenten verbessern läßt. Fragen z.B. der Graphikoberfläche oder geeigneter Interaktionsgeräte werden nicht diskutiert. Die zentralen Punkte im Artikel sind Autonomie und Flexibilität.

Autonomie bedeutet, daß ein Büroagent Aufgaben *ohne* (oder mit ganz wenig) Benutzerinteraktion bearbeitet. Insbesondere sollte der "Büromensch" von regelmäßigen Arbeiten mit stupidem Charakter befreit werden. Da viele Aufgaben im Büro kooperativ von mehreren Personen bearbeitet werden, müssen die Agenten auch Modelle gängiger Arbeitsabläufe (workflows) besitzen. Diese sind oft nicht nur einfache Aktionssequenzen, sondern können komplexe *Verhandlungsstrukturen* beinhalten, die beliebige Zyklen sowie Planen und Beurteilen von Alternativlösungen und Ersatzressourcen erfordern. Außerdem ist daran zu denken, daß (heute) wahrscheinlich noch nicht alle beteiligten "Büromenschen" die notwendige Agentensoftware besitzen. Dennoch sind sie Teil des Arbeitsablaufs, und die Agenten müssen auch mit ihnen interagieren können.

Allgemein einsetzbare Agenten mit der genannten Verhandlungsfähigkeit zu implementieren, ist – wegen der Komplexität der Domäne, den vielen speziellen Aufgaben und den Unterschieden innerhalb verschiedener Organisationen – praktisch unmöglich. Jede Person hat Präferenzen, wann und wie sie eine Aufgabe angeht. In jeder Organisation sind Formblätter bzw. die Regeln zum Ausfüllen anders, usw. Daher sollten die Agenten genügend *Flexibilität* besitzen, sich auf neue Situationen, Umgebungen und evtl. neue Aufgaben "einzustellen". Dazu schlagen wir neben induktiven Verfahren zur Detektion von Benutzerpräferenzen den Ansatz des *dialogbasierten Lernens (DBL)* vor [2], bei dem der Agent mit Hilfe von Dialogen *bei Bedarf*, d.h. in unbekannten Situationen, vom Benutzer neues Wissen akquiriert und für den späteren Gebrauch speichert oder – im Falle neuartiger Aktionen – sogar operationalisiert.

Autonomie, Verhandlungsfähigkeit und Flexibilität durch integrierte Lernmechanismen werden an unseren Implementierungen CAP II[1], einem Terminplanungsagenten, und RAP[2], einem Raumreservierungsagenten, diskutiert.

[1] CAP II steht für Calendar APprentice, Version II; zum "Verhältnis" von Agenten zu Apprentices vgl. [1].

[2] RAP steht für Room APprentice; vgl. auch [2].

Der Rest des Artikels ist wie folgt gegliedert. In Kapitel 2 werden verwandte Arbeiten und ihr Verhältnis zu unserem Ansatz diskutiert. Die Kapitel 3 und 4 beschreiben im Detail die Büroagenten CAP II und RAP. Kapitel 5 enthält die Zusammenfassung unserer Ergebnisse und Kapitel 6 gibt einen Ausblick auf zukünftige geplante Forschungsaktivitäten.

2 Verwandte Arbeiten

Verhandlungen zwischen Kalenderagenten werden von Sen und Durfee beschrieben [9]. Ihre Experimente wurden bisher nur in einer Simulationsumgebung für Kontraktnetzapplikationen durchgeführt. Dort evaluieren sie von ihnen vorhergesagte Formeln über den Zusammenhang zwischen der Anzahl von Verhandlungszyklen und Kommunikationskosten bezogen auf verschiedene Ansage-, Bieter- und Zusagestrategien. Die Verhandlung in unserem CAP II System ist eine Erweiterung des von Sen und Durfee verwendeten Modells kommunizierender, endlicher Automaten [1].

Eine interessante Implementierung von Terminverhandlungsagenten wird von Lux beschrieben [6]. Auf der Basis einer allgemeinen "Sprache" von Verhandlungsprimitiven wurden Kalenderagenten modelliert und an existierende Tools (den EMACS Kalender und Sun's Xcalentool) als Interface gekoppelt. Der Ansatz ist deswegen interessant, weil er einen praktischen Weg aufzeigt, wie man beliebige Kalendersoftware seiner Wahl systematisch zu verhandelnden Agenten erweitern kann.

Als Anwendung für seine Theorie strukturierter Konversationen wurde von Woitass das System TVS implementiert [11]. Zentrale Instanz bei ihm ist ein *Mediatoragent*, der die Terminkoordination vornimmt. Im Unterschied zu CAP II verhandeln Mediatoragenten jedoch (noch) nicht mit anderen "Rechneragenten" (obwohl sie das prinzipiell könnten), sondern mit Menschen, die am TVS-Benutzerinterface Eingaben machen müssen. Die möglichen Nachrichtentypen (vgl. [10]) sind mit denen in CAP II vergleichbar, jedoch erfolgt das Versenden über eine selbstgeschriebene X.400 Schnittstelle und nicht über Email wie in CAP II. Das macht das Einbeziehen von Personen ohne TVS unmöglich.

Noch allgemeiner als Woitass' Modell für strukturierte Konversationen ist das *Action Workflow Loop* Konzept von Medina-Mora, Winograd und Flores [7], das Vorschlag des Initiators (proposal), Übereinstimmung finden (agreement; die eigentliche Verhandlung), Zustimmung an Initiator senden (performance; in CAP II das Senden eines "yes-bids") und Bestätigung vom Initiator an alle senden (satisfaction; in CAP II confirmation genannt) beinhaltet. In CAP II kommt noch der Schritt Rückbestätigung (validation) dazu. Er ist bei autonom miteinander verhandelnden Agenten nötig, wenn sie eine "bedingte Zustimmungsstrategie" benutzen, d.h. trotz erster Zusage einen Termin überlagern können, solange vom Initiator die Bestäting (confirmation) noch fehlt [1].

Ein verhandelnder Kalender, der im Gegensatz zu allen bisher genannten Systemen auch *lernfähig* ist, wurde von Kozierok und Maes [5] implementiert. Er verwendet *memory-based learning*, während CAP II explizit generalisierende Regeln induziert (mit ID3 und Backpropoagation). Der Unterschied bzgl. des Verhandelns ist darin zu sehen, daß ein Scheduler verwendet wird, der von allen gewünschten Teilnehmern die freien Zeitschlitze sammelt und dann, auf der Basis der durch Lernen ständig verbesserten Kriterien, den

Der Rest des Artikels ist wie folgt gegliedert. In Kapitel 2 werden verwandte Arbeiten und ihr Verhältnis zu unserem Ansatz diskutiert. Die Kapitel 3 und 4 beschreiben im Detail die Büroagenten CAP II und RAP. Kapitel 5 enthält die Zusammenfassung unserer Ergebnisse und Kapitel 6 gibt einen Ausblick auf zukünftige geplante Forschungsaktivitäten.

2 Verwandte Arbeiten

Verhandlungen zwischen Kalenderagenten werden von Sen und Durfee beschrieben [9]. Ihre Experimente wurden bisher nur in einer Simulationsumgebung für Kontraktnetzapplikationen durchgeführt. Dort evaluieren sie von ihnen vorhergesagte Formeln über den Zusammenhang zwischen der Anzahl von Verhandlungszyklen und Kommunikationskosten bezogen auf verschiedene Ansage-, Bieter- und Zusagestrategien. Die Verhandlung in unserem CAP II System ist eine Erweiterung des von Sen und Durfee verwendeten Modells kommunizierender, endlicher Automaten [1].

Eine interessante Implementierung von Terminverhandlungsagenten wird von Lux beschrieben [6]. Auf der Basis einer allgemeinen "Sprache" von Verhandlungsprimitiven wurden Kalenderagenten modelliert und an existierende Tools (den EMACS Kalender und Sun's Xcalentool) als Interface gekoppelt. Der Ansatz ist deswegen interessant, weil er einen praktischen Weg aufzeigt, wie man beliebige Kalendersoftware seiner Wahl systematisch zu verhandelnden Agenten erweitern kann.

Als Anwendung für seine Theorie strukturierter Konversationen wurde von Woitass das System TVS implementiert [11]. Zentrale Instanz bei ihm ist ein *Mediatoragent*, der die Terminkoordination vornimmt. Im Unterschied zu CAP II verhandeln Mediatoragenten jedoch (noch) nicht mit anderen "Rechneragenten" (obwohl sie das prinzipiell könnten), sondern mit Menschen, die am TVS-Benutzerinterface Eingaben machen müssen. Die möglichen Nachrichtentypen (vgl. [10]) sind mit denen in CAP II vergleichbar, jedoch erfolgt das Versenden über eine selbstgeschriebene X.400 Schnittstelle und nicht über Email wie in CAP II. Das macht das Einbeziehen von Personen ohne TVS unmöglich.

Noch allgemeiner als Woitass' Modell für strukturierte Konversationen ist das *ActionWorkflow Loop* Konzept von Medina-Mora, Winograd und Flores [7], das Vorschlag des Initiators (proposal), Übereinstimmung finden (agreement; die eigentliche Verhandlung), Zustimmung an Initiator senden (performance; in CAP II das Senden eines "yes-bids") und Bestätigung vom Initiator an alle senden (satisfaction; in CAP II confirmation genannt) beinhaltet. In CAP II kommt noch der Schritt Rückbestätigung (validation) dazu. Er ist bei autonom miteinander verhandelnden Agenten nötig, wenn sie eine "bedingte Zustimmungsstrategie" benutzen, d.h. trotz erster Zusage einen Termin überlagern können, solange vom Initiator die Bestäting (confirmation) noch fehlt [1].

Ein verhandelnder Kalender, der im Gegensatz zu allen bisher genannten Systemen auch *lernfähig* ist, wurde von Kozierok und Maes [5] implementiert. Er verwendet *memory-based learning*, während CAP II explizit generalisierende Regeln induziert (mit ID3 und Backpropoagation). Der Unterschied bzgl. des Verhandelns ist darin zu sehen, daß ein Scheduler verwendet wird, der von allen gewünschten Teilnehmern die freien Zeitschlitze sammelt und dann, auf der Basis der durch Lernen ständig verbesserten Kriterien, den

1 Einleitung

An der Carnegie Mellon University wird unter dem Arbeitstitel *Software Secretary* Büroautomatisierungssystemen gearbeitet, die den "Büromenschen" bei seiner täglich Arbeit unterstützen sollen: Beim Vereinbaren von Terminen, beim Verwalten von Ressourcen (wie Besprechungs- oder Vortragsräumen), später auch bei Budgetplanung oder Abrechnungs- und Bestellwesen. Die Idee des *Software Secretary* klingt attraktiv, da eine derartige Software sekretariatsähnliche Leistungen *für jedermann/frau* bereitstellen könnte, der/die heute noch nicht das "Privileg" einer eigenen Sekretärin besitzt.

Drei Eigenschaften bestimmen Erfolg oder Mißerfolg solcher Systeme: Der Grad der *Autonomie*, die *Flexibilität*, mit verschiedenen Benutzern in verschiedenen Umgebungen verschiedene Aufgaben lösen zu können, und die *Benutzerschnittstelle*. Auf die Benutzerschnittstelle wird in diesem Artikel nur soweit eingegangen wie sich die Anzahl der notwendigen Interaktionen oder der Bedienungskomfort durch wachsendes Wissen und Autonomie der Agenten verbessern läßt. Fragen z.B. der Graphikoberfläche oder geeigneter Interaktionsgeräte werden nicht diskutiert. Die zentralen Punkte im Artikel sind Autonomie und Flexibilität.

Autonomie bedeutet, daß ein Büroagent Aufgaben *ohne* (oder mit ganz wenig) Benutzerinteraktion bearbeitet. Insbesondere sollte der "Büromensch" von regelmäßigen Arbeiten mit stupidem Charakter befreit werden. Da viele Aufgaben im Büro kooperativ von mehreren Personen bearbeitet werden, müssen die Agenten auch Modelle ganzer Arbeitsabläufe (workflows) besitzen. Diese sind oft nicht nur einfache Aktionssequenzen, sondern können komplexe *Verhandlungsstrukturen* beinhalten, die beliebige Zyklen sowie Planen und Beurteilen von Alternativlösungen und Ersatzressourcen erfordern. Außerdem ist daran zu denken, daß (heute) wahrscheinlich noch nicht alle beteiligten "Büromenschen" die notwendige Agentensoftware besitzen. Dennoch sind sie Teil des Arbeitsablaufs, und die Agenten müssen auch mit ihnen interagieren können.

Allgemein einsetzbare Agenten mit der genannten Verhandlungsfähigkeit zu implementieren, ist – wegen der Komplexität der Domäne, den vielen speziellen Aufgaben und den Unterschieden innerhalb verschiedener Organisationen – praktisch unmöglich. Jede Person hat Präferenzen, wann und wie sie eine Aufgabe angeht. In jeder Organisation sind Formblätter bzw. die Regeln zum Ausfüllen anders, usw. Daher sollten die Agenten genügend *Flexibilität* besitzen, sich auf neue Situationen, Umgebungen und evtl. auch Aufgaben "einzustellen". Dazu schlagen wir neben induktiven Verfahren zur Detektion von Benutzerpräferenzen den Ansatz des *dialogbasierten Lernens (DBL)* vor [2], bei dem der Agent mit Hilfe von Dialogen *bei Bedarf*, d.h. in unbekannten Situationen, vom Benutzer neues Wissen akquiriert und für den späteren Gebrauch speichert oder – im Falle neuartiger Aktionen – sogar operationalisiert.

Autonomie, Verhandlungsfähigkeit und Flexibilität durch integrierte Lernmechanismen werden an unseren Implementierungen CAP II[1], einem Terminplanungsagenten, und RAP[2], einem Raumreservierungsagenten, diskutiert.

[1] CAP II steht für Calendar APprentice, Version II; zum "Verhältnis" von Agenten zu Apprentice-Systemen vgl. [1].

[2] RAP steht für Room APprentice; vgl. auch [2].

Der Rest des Artikels ist wie folgt gegliedert. In Kapitel 2 werden verwandte Arbeiten und ihr Verhältnis zu unserem Ansatz diskutiert. Die Kapitel 3 und 4 beschreiben im Detail die Büroagenten CAP II und RAP. Kapitel 5 enthält die Zusammenfassung unserer Ergebnisse und Kapitel 6 gibt einen Ausblick auf zukünftige geplante Forschungsaktivitäten.

2 Verwandte Arbeiten

Verhandlungen zwischen Kalenderagenten werden von Sen und Durfee beschrieben [9]. Ihre Experimente wurden bisher nur in einer Simulationsumgebung für Kontraktnetzapplikationen durchgeführt. Dort evaluieren sie von ihnen vorhergesagte Formeln über den Zusammenhang zwischen der Anzahl von Verhandlungszyklen und Kommunikationskosten bezogen auf verschiedene Ansage-, Bieter- und Zusagestrategien. Die Verhandlung in unserem CAP II System ist eine Erweiterung des von Sen und Durfee verwendeten Modells kommunizierender, endlicher Automaten [1].

Eine interessante Implementierung von Terminverhandlungsagenten wird von Lux beschrieben [6]. Auf der Basis einer allgemeinen "Sprache" von Verhandlungsprimitiven wurden Kalenderagenten modelliert und an existierende Tools (den EMACS Kalender und Sun's Xcalentool) als Interface gekoppelt. Der Ansatz ist deswegen interessant, weil er einen praktischen Weg aufzeigt, wie man beliebige Kalendersoftware seiner Wahl systematisch zu verhandelnden Agenten erweitern kann.

Als Anwendung für seine Theorie strukturierter Konversationen wurde von Woitass das System TVS implementiert [11]. Zentrale Instanz bei ihm ist ein *Mediatoragent*, der die Terminkoordination vornimmt. Im Unterschied zu CAP II verhandeln Mediatoragenten jedoch (noch) nicht mit anderen "Rechneragenten" (obwohl sie das prinzipiell könnten), sondern mit Menschen, die am TVS-Benutzerinterface Eingaben machen müssen. Die möglichen Nachrichtentypen (vgl. [10]) sind mit denen in CAP II vergleichbar, jedoch erfolgt das Versenden über eine selbstgeschriebene X.400 Schnittstelle und nicht über Email wie in CAP II. Das macht das Einbeziehen von Personen ohne TVS unmöglich.

Noch allgemeiner als Woitass' Modell für strukturierte Konversationen ist das *ActionWorkflow Loop* Konzept von Medina-Mora, Winograd und Flores [7], das Vorschlag des Initiators (proposal), Übereinstimmung finden (agreement; die eigentliche Verhandlung), Zustimmung an Initiator senden (performance; in CAP II das Senden eines "yes-bids") und Bestätigung vom Initiator an alle senden (satisfaction; in CAP II confirmation genannt) beinhaltet. In CAP II kommt noch der Schritt Rückbestätigung (validation) dazu. Er ist bei autonom miteinander verhandelnden Agenten nötig, wenn sie eine "bedingte Zustimmungsstrategie" benutzen, d.h. trotz erster Zusage einen Termin überlagern können, solange vom Initiator die Bestäting (confirmation) noch fehlt [1].

Ein verhandelnder Kalender, der im Gegensatz zu allen bisher genannten Systemen auch *lernfähig* ist, wurde von Kozierok und Maes [5] implementiert. Er verwendet *memory-based learning*, während CAP II explizit generalisierende Regeln induziert (mit ID3 und Backpropoagation). Der Unterschied bzgl. des Verhandelns ist darin zu sehen, daß ein Scheduler verwendet wird, der von allen gewünschten Teilnehmern die freien Zeitschlitze sammelt und dann, auf der Basis der durch Lernen ständig verbesserten Kriterien, den

optimalen Zeitpunkt für das Treffen bestimmt. Ein CAP II Agent besitzt (bisher) Schedulingfunktionalität. Angebote und Entscheidungen werden rein lokal getr unter Verwendung des Benutzerkalenders sowie der eintreffenden Email-Nachrichte

3 Das Terminverhandlungssystem CAP II

CAP II ist die Erweiterung des "Einplatzkalenders" CAP [3], der beim Eintragen Ereignissen Werte für Zeitdauer, Besprechungsraum etc. vorschlägt[3]. Wenn der Ben die Werte nicht akzeptiert, kann er sie einfach überschreiben. Die vorgeschlagenen V werden durch Regeln bestimmt, die aus allen Kalendereinträgen *gelernt* werden. Beispiel ist die Regel

```
IF AttendeeStatus = Undergrad
AND AttendeeDepartment = ComputerScience
THEN duration = 30
```

die den Ratschlag gibt, Besprechungen mit "Undergrad-Studenten" des Computer Sc Department nicht länger als 30 Minuten dauern zu lassen. Das Lernprogramm automatisch jede Nacht gestartet[4] und generiert - nach ein bis zwei Monaten - die e *verläßlichen* Regeln dieser Art, die die Vorlieben des Benutzers modellieren.

CAP II erweitert CAP um Mechanismen zum autonomen Vereinbaren von Te nen über Email. In den nächsten Abschnitten werden die Anforderungen an CA das Verhandlungsprotokoll, das Nachrichtenformat sowie die Gesamtarchitektur de liert vorgestellt.

3.1 Anforderungen an CAP II

Aus den beiden Zielvorstellungen für CAP II, nämlich *autonomes Verhandeln* und *bezug von Personen auch ohne CAP II Agent*, wurden die folgenden Anforderu abgeleitet:

- Das Terminverhandeln sollte die Arbeit des Benutzers am Rechner so wenig möglich unterbrechen.
- Der Benutzer muß Entscheidungen des Agenten zu jeder Zeit zurücksetzen kön
- Ein "Time-out Mechanismus" muß bereitgestellt werden, um bei ausbleibe Antworten geeignet zu reagieren.
- Um Benutzer ohne CAP II Agent an Verhandlungen teilnehmen zu lassen, das Nachrichtenformat "natürlich" (-sprachlich) sein.
- Es müssen einfache Filter- und Nachrichtenzuordnungsmechanismen bereitge werden, um aufwendige Textanalysen der Email zu vermeiden.

[3] Das Vorschlagen von Tag und Uhrzeit ist noch nicht zufriedenstellend.

[4] Es verwendet ID3 und Backpropagation aus der "ML-Toolbox" THEO [8].

- Ebenso sollte der CAP II Agent den Inhalt einer Nachricht schnell und ohne teure sematische Analyse "verstehen" können.
- Zusätzliche Unterstützung für Personen ohne CAP II Agent soll vorgesehen werden.

In den folgenden Abschnitten wird gezeigt, daß unter einer sogenannten *Gutwilligkeitsannahme* alle Anforderungen an CAP II erfüllt werden konnten.

3.2 Das Verhandeln in CAP II

CAP II ist ein System kommunizierender, endlicher Automaten, die beim Verhandeln einem Kontraktnetzprotokoll folgen, das eine Erweiterung von [9] darstellt. Die Agenten entscheiden rein reaktiv und lokal; es ist keine Planung nötig. Die Synchronisation erfolgt nur durch Kommunikation und Verhandlung.

Sobald ein Benutzer den Wunsch zu einem Treffen in CAP II eingetragen hat (per `add-meeting` mit Flag `not-confirmed`), wird vom Agenten eine Nachricht mit Einladung generiert und per Email an alle spezifizierten Teilnehmer gesandt. Besitzen die Empfänger CAP II Agenten, verhandeln diese zunächst autonom mit dem einladenden Agent, bis ein für alle geeigneter Zeitpunkt gefunden wurde (oder auch nicht). Sollte ein Empfänger keinen CAP II Agenten haben, muß er die Nachricht persönlich wie jede andere Email beantworten (s.u.). Als Antwort können die Angebote "yes", "not-then", "maybe" oder "no" zurückgeschickt werden (zur genauen Semantik vgl. [1][5]). Wurde eine Einigung erzielt, folgt eine Art "handshake" Prozedur von Bestätigung (confirmation) durch den einladenden Agenten und Rückbestätigungen (validations) durch die Eingeladenen. Falls Antworten ausbleiben, generiert und verschickt ein Ausnahmebehandlungsprozeß Mahnungen (reminders), bzw. benachrichtigt, falls die Mahnungen nicht helfen, den Benutzer, so daß dieser den "Unerreichbaren" per Telefon oder Fax kontaktieren kann. Die Eigenschaft des *asynchronen* und – bis zu einem gewissen Grad – *benutzerunabhängigen* Verhandelns macht aus dem Apprenticesystem CAP den *Agenten* CAP II [1].

Idealerweise verhandeln die Agenten untereinander bis zur Einigung, ohne daß die Benutzer gestört werden – außer wenn Probleme auftauchen. Nachdem Erfolg (oder Mißerfolg) feststehen, werden sie benachrichtigt[6]. Sollten sie mit dem Termin nicht einverstanden sein, können sie sofort Dienste wie "cancel-participation", "move-request" etc. aufrufen. Das kann dann zu weiteren Verhandlungsschleifen führen. Der Einladende hat zusätzlich noch die Möglichkeit, ein Treffen komplett abzusagen, ohne daß Verhandlungen nötig werden.

3.3 Das Nachrichtenformat in CAP II

Das Nachrichtenformat in CAP II wird vor allem durch die Vorgabe bestimmt, auch Personen ohne solche Agenten in Verhandlungen mit einbeziehen zu können. Ohne diese

[5] Das Zurückschicken von Alternativvorschlägen ist geplant, aber noch nicht implementiert.

[6] Bisher sehen sie nur eine textuelle Nachricht; in Kürze wird ebenfalls ein blinkender Eintrag im Kalender erscheinen (sofern er angezeigt ist), um die Aufmerksamkeit des Benutzers zu erregen. Denkbar ist auch ein Icon wie die "xbiff" Mailbox, die ihre Farbe wechselt und eine Fahne hochgehen läßt, wenn neue Email angekommen ist.

Vorgabe würde jeder gepackte Bitstring ausreichen. Die Lösung in CAP II sind strukturierte Nachrichten sowie zusätzliche Hinweise am Textende, wie ein "CAP-lose" Benutzer antworten soll.

```
To:bocionek@cs
From:mitchell@cs
Subject:Meeting on dec-24-1992 at 2:30pm?  CAP-Requestxx123

This message is to suggest a meeting with Tom Mitchell
as follows.  Please let us know if such a meeting is
acceptable to you.

  Date:  dec-24-1992
  Time:  2:30 pm
  Duration:  30
  Location:  Weh5409
  Attendees:  Tom, Siegfried, Stork
  Topics:  Cancel Messages, Parsing in CAP II, NL Interface

You can make your reply computer-understandable by
beginning your message with one of the following words:
Yes, Not-then, Maybe, No
```

Abbildung 1: Eine vom CAP II Agenten generierte Einladung

Abb. 1 zeigt eine Einladung, wie sie von CAP II Agenten verschickt wird. Das Wort **CAP-Request** am Ende der Subject-Zeile im Nachrichtenkopf wird benutzt, um die Email für CAP II aus allen anderen herauszufiltern[7]. Die eindeutige Vorgangsnummer (event identifier) **xx123**, die an **CAP-Request** angehängt ist, ermöglicht die sofortige Zuordnung der Nachricht zu einer bestimmten Verhandlung. Wenn Agenten Nachrichten verschicken ist diese Nummer immer Bestandteil der Subject-Zeile. Bei Benutzern ohne CAP II Agenten *nehmen wir an*, daß sie eine Antworttaste oder -option benutzen, so daß die Subject-Zeile automatisch in die Antwort hineinkopiert wird[8]. Wegen unserer Gutwilligkeitsannahme (vgl. Abschnitt 3.5) gehen wir davon aus, daß Benutzer nicht böswillig falsche Vorgangsnummern verwenden.

Bei der Analyse des Nachrichtentexts *sucht* der CAP II Agent lediglich nach einer Schlüsselphrase, oft nur nach einem Schlüsselwort im Text (in Abb. 1 ist das "**suggest meeting**"). Diese Schlüsselphrase bestimmt den Typ der Nachricht (proposal, bid, con-

[7]Dazu wird die lokale **.maildelivery** Datei der MMDF Software unter Mach und Unix verwendet, in die man einfache Regeln der Form "**if CAP-Request is in subject line, then append message to file .my-newmail**" eintragen kann.

[8]Sollte so ein Benutzer aus irgendeinem Grund die Antwort mit einer Subject-Zeile ohne Vorgangsnummer verschicken, wird sie vom empfangenden Agenten ignoriert; der Benutzer wird aber später durch automatische Mahnungen des Ausnahmebehandlungsprozesses zur erneuten Antwort aufgefordert. Ist der "CAP-lose" Benutzer der Einladende, und verwendet er **CAP-request** ohne eine Vorgangsnummer, erzeugen die empfangenden Agenten eine. Trotzdem diese Nummer für alle Agenten verschieden sein wird, ist eine korrekte Terminverhandlung möglich [1].

mation, validation, ...) und damit die Reaktion des Agenten sowie den Zustandsübergang im Automat.

Der wichtigste Hilfemechanismus für "CAP-lose" Benutzer ist ein Hinweis am Nachrichtenende, wie geantwortet werden soll (in Abb. 1 "`You can make your reply computerunderstandable ...`"). Weitere Unterstützung bietet das automatisches Ergänzen von Nachrichten, flexibles Matching und – im Fall *unverständlicher* Antworten – automatisches Zurückschicken von Beispielantworten (die Details dazu sind in [1] beschrieben).

3.4 Die Architektur von CAP II

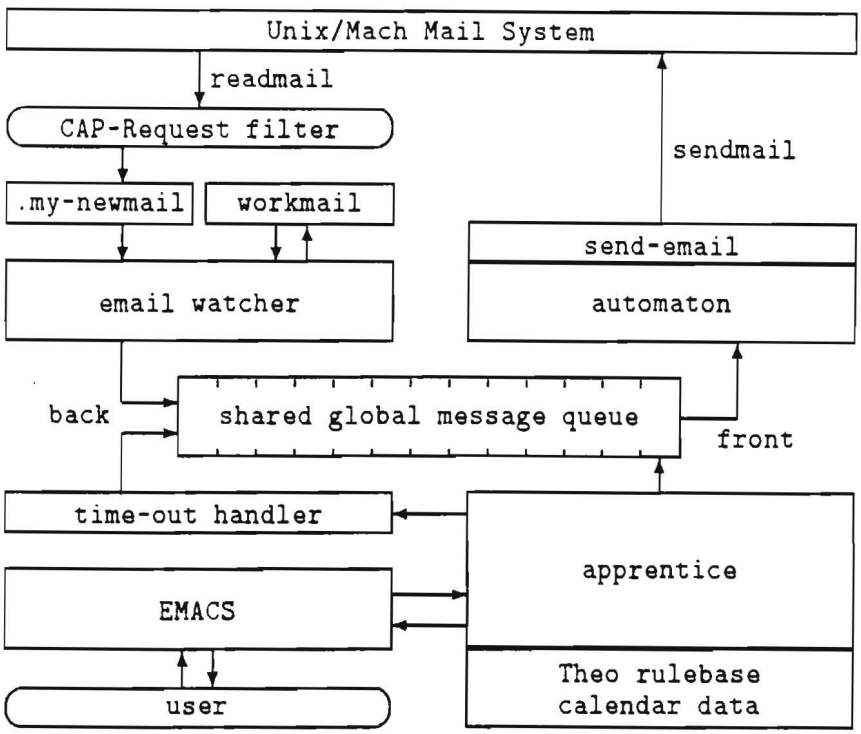


Abbildung 2: Die Prozeßstruktur von CAP II

CAP II ist in Common LISP auf einer Mach/Unix Plattform implementiert. Das Kalenderinterface – in EL-LISP – ist ein EMACS-Prozeß, der alle übrigen Subprozesse aktiviert und per Remote Procedure Call (RPC) Zugriff zum `apprentice` Prozeß hat (vgl. Abb. 2). Die Subprozesse `automaton` und `time-out handler` können darüberhinaus über Named Pipes mit dem EMACS Interface kommunizieren. `email watcher` kommuniziert mit keinem anderen Subprozeß.

`apprentice` stellt den Zugang zum Kalender und zu den gelernten Regeln in der THEO Regelbasis [8] bereit. Dieser Subprozeß arbeitet wie in der originalen CAP version [4]. Nach Aufruf eines Macros (wie `add-event`) vom EMACS-Prozeß, erzeugt `apprentice`

einen Deskriptor für den Vorgang und trägt ihn am Anfang der Warteschlange für v Automat zu bearbeitende Nachrichten ein[9]. Aus der Warteschlange entnimmt der S prozeß `automaton` die Nachrichtendeskriptoren und bearbeitet sie wie in Abschnitt beschrieben. Falls Email zu senden ist, ruft `automaton` eine Funktion `send-email` ein eigener Prozeß ist dafür nicht nötig.

Der zweite Subprozeß, der `automaton` mit Nachrichtendeskriptoren versorgt, ist em `watcher`. Er pollt die Datei `.my-newmail`, in die der MMDF Filter (s.o.) Nachrichten CAP II Agenten ablegt. Sobald eine neue Nachricht eingetroffen ist, wird sie analysiert, einen Nachrichtendeskriptor umgewandelt und am Ende der Warteschlange eingetrag Falls mehr als eine Nachricht in `.my-newmail` vorliegt, wird die Datei `workmail` z Puffern benutzt.

Der dritte Subprozeß, der `automaton` mit Nachrichtendeskriptoren versorgt, ist `time-out handler`. Er hat Zugriff auf Internstrukturen von `automaton` und `apprenti` (in Abb. 2 nicht gezeigt) und erzeugt Mahnungen, wenn Angebote oder Rückbestätig gen nicht in einem bestimmten (konfigurierbaren) Zeitraum eintreffen. Zudem kann über den Nachrichtenmechanismus, Information an das EMACS Interface zur Anze beim Benutzer absetzen, falls trotz Mahnungen keine Antworten empfangen werden.

3.5 Diskussion

Die CAP II Implementierung erfüllt alle in Abschnitt 3.1 aufgestellten Anforderung Autonomes Verhandeln nimmt dem Benutzer häufiges Versenden von Email, bzw. vi Telefonanrufe ab, wenn er ein Treffen organisieren will. Seine Aufmerksamkeit – außer Fehlerfall – wird nur am Ende einer Verhandlung verlangt. Dienste wie `cancel-requ` oder `move-request` ermöglichen ihm, jede Agentenentscheidung sofort rückzusetzen. Ausnahmebehandlungsprozeß sichert zu, daß der Benutzer stets benachrichtigt wird, we ein Adressat unerreichbar ist (z.B. weil Rechner ausgefallen oder die Person verreist is Dann kann er diese Person per Telefon oder Fax zu erreichen versuchen. Das Mo der kommunizierenden Automaten ist adäquat, da es genau das Verhalten von Mensch modelliert, wenn sie Termine vereinbaren.

Durch das semistrukturierte Nachrichtenformat wird ermöglicht, daß auch Perso ohne CAP II Agenten in Terminverhandlungen einbeziehbar sind. Das Filtern und Zuo nen von Emails zu Vorgängen ist einfach wie gefordert. Der Inhalt der Nachrichten ka allein durch Matching von Schlüsselphrasen bestimmt werden. In keinem Fall ist e aufwendige, natürlichsprachliche Analyse notwendig. Hinweise am Ende jeder Nachri sowie zusätzliche Hilfemechanismen unterstützen "CAP-lose" Email-Benutzer. Die diese Weise gewonnene Flexibilität, nämlich sowohl mit Agenten wie mit Menschen v handeln zu können, ist ein *Muß*, solange noch keine standardisierten Terminplaner gramme in jedem Büro vorhanden sind.

Eine wichtige *Gutwilligkeitsannahme* liegt jedoch der von uns vorgeschlagenen In gration von Agenten und Menschen zugrunde. Alle Teilnehmer *müssen* kooperativ s wollen, oder dürfen zumindest die Verhandlungen nicht mutwillig "sabotieren". Da z.B. keine natürlichsprachliche Analyse der Nachrichten vornehmen – und mit Absi

[9] Am Anfang, da der EMACS-Aufruf ein RPC ist und der Benutzer nicht unnötig warten soll.

auch nicht wollen - könnte man durch Einfügen von Negationen in Antworten deren Sinn ins Gegenteil verkehren, ohne daß CAP II Agenten das merken würden. Unserer Meinung nach werden Probleme i.d.R. nicht auftreten, jedenfalls nicht öfter als auch jetzt schon bei telefonischen Vereinbarungen.

4 Das Raumreservierungssystem RAP

Obwohl das Kalendersystem CAP II einige Flexibilität besitzt, sich an die Präferenzen seiner Benutzer zu adaptieren, verwendet es dennoch eine feste, vorgegebene Verhandlungsstrategie. Daher kann es nicht einfach an Besonderheiten z.B. in verschiedenen Organisationen angepaßt werden. In diesem Kapitel wird mit dem Raumreservierungsagenten RAP ein System vorgeschlagen, daß Verhandlungsabläufe inkrementell akquirieren kann, indem ein- und ausgehende Email analysiert und der Benutzer, wenn nötig, über den Inhalt befragt wird. RAP ist ein erstes Beispiel, an dem die Mächtigkeit von *dialogbasiertem Lernen (DBL)* zur Synthese von neuen Workflow-Strukturen erforscht werden soll [2].

4.1 Anforderungen an RAP

Die Aufgabe eines RAP Agenten ist das Reservieren von Besprechungs- und Vortragsräumen über Email. RAP lernt dabei, die für die Verteilung zuständigen Sekretärinnen der verschiedenen Fakultäten der CMU solange zu fragen, bis ein freier Raum gefunden ist. Hat RAP den Vorgang gelernt, führt der Agent die *komplette Suche* nach einem Raum vollständig autonom durch.

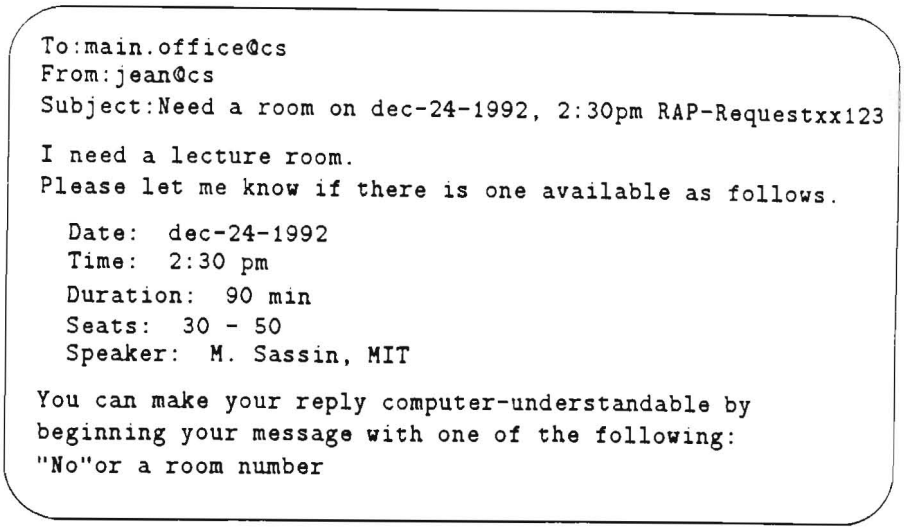
Im einzelnen ergeben sich daher - neben den auch schon für CAP II in Abschnitt 3.1 genannten - folgende zusätzliche Anforderungen:

- Der endliche Automat, der die Verhandlung in RAP treibt, soll komplett synthetisiert werden.
- Es sollen die verschiedenen Nachrichtenmuster akquiriert werden, die per Email hin- und herzuschicken sind.
- Eine kleine Datenbank mit Information über Räume und RaumverwalterInnen soll aufgebaut werden.

Im folgenden wird nur das Lernen des Automaten beschrieben. Details über die anderen Teilaufgaben sind in [2] zu finden.

4.2 Das Nachrichtenformat in RAP

Für das Nachrichtenformat in RAP gilt bzgl. Aufbau, Filtermechanismus, Hilfestellung für die Empfänger der Nachrichten usw. dasselbe wie für CAP II. Eine Anfrage ist in Abb. 3 zu sehen. Die Schlüsselphrase ist hier `"need a lecture room"`.



```
To:main.office@cs
From:jean@cs
Subject:Need a room on dec-24-1992, 2:30pm RAP-Requestxx123

I need a lecture room.
Please let me know if there is one available as follows.

  Date:  dec-24-1992
  Time:  2:30 pm
  Duration:  90 min
  Seats:  30 - 50
  Speaker:  M. Sassin, MIT

You can make your reply computer-understandable by
beginning your message with one of the following:
"No"or a room number
```

Abbildung 3: Von einem RAP Agent generierte Bestellung eines Vortragsraums

4.3 Dialogbasiertes Lernen in RAP

Sieht ein RAP Agent eine Email zum ersten Mal (und kann den Inhalt nicht "erschließen") stellt er dem Benutzer Fragen wie

> `To which type of interaction does this message belong? (room-request positive-answer, negative-answer, alternative-proposal, other)`
>
> `Which is the meaning-carrying word or phrase in that message?`
>
> `Is this the last action of the task?`

Aus den Antworten auf diese Fragen wird ein *Thesaurus* aus Nachrichtentypen und Schlüsselphrasen aufgebaut. Alle Phrasen, die zu einem Nachrichtentyp gehören, werden als *Synonyme* behandelt.

Bei jedem neu erkannten Nachrichtentyp wird der endliche Automat erweitert, der die RAP Agenten treibt. An vereinfachenden Annahmen wurden u.a. getroffen (siehe auch [2]), daß stets nur eine Person nach einem Raum für dasselbe Ereignis fragt, und daß die einzelnen RaumverwalterInnen hintereinander kontaktiert werden. Die Strategie, wann und wie der Automat zu erweitern ist, ist ebenfalls fest vorgegeben.

Tabelle 1 zeigt den endlichen RAP Automat (die Zeilenindizes bezeichnen die Zustände, die Spaltenindizes die akzeptierten Nachrichtentypen), nachdem die Typen `room-request` und `positive-answer`, sowie der Zustand `s1` gelernt wurden. Beim Start von RAP war nur der "leere" Automat mit Zuständen `global-start` und `global-stop` sowie Nachrichtentypen `stop-RAP` und (aus technischen Gründen) `EOF` existent. Eine neue Spalte entsteht immer, wenn ein neuer Nachrichtentyp eingeführt wird. Neue Zeilen kommen hinzu, wenn ein neuer Nachrichtentyp *nicht* den Abschluß eines Vorgangs bezeich-

(d.h. nicht in den Anfangszustand `global-start` zurückführt). Diese Strategie konstruiert nicht den minimalen Automaten, aber sie lernt die Aufgabe[10].

Man beachte auch die leeren Felder in der Übergangstafel. Ihr Inhalt wird erst synthetisiert, wenn so ein Fall wirklich eintritt, also z.B. eine `positive-answer` trifft im Zustand `global-start` ein, d.h. ohne daß (in `s1`) auf so eine Nachricht gewartet wird. Als Fehleraktion in diesem Fall könnte per DBL eine Benachrichtigung des Benutzers oder das "stillschweigende" Löschen der Nachricht eingetragen werden.

	EOF	stop-RAP	room-request	positive-answer
global-start	–	goto global-stop	send-email goto s1	
global-stop	clean-up process-exit	–	–	–
s1	–	goto global-stop		notify-user(success) goto global-start

Tabelle 1: Ein Teil des endlichen Automaten von RAP

Eine detaillierte Beschreibung des vollständigen Lernalgorithmus von RAP ist in [2] zu finden.

4.4 Diskussion

RAP soll dazu verwendet werden, die Mächtigkeit des dialogbasierten Lernens zur Erweiterung von Bürosoftware *während ihres Gebrauchs* zu erforschen. Ohne Einbezug des Benutzers, nur durch maschinelle Lernalgorithmen ist das (heute; nur heute?) nicht möglich. Das Ergebnis sind Agenten, die sich den Präferenzen ihrer Benutzer sowie den speziellen Abläufen in unterschiedlichen Organisationen anpassen können. Diese Eigenschaft wird die Akzeptanz solcher Software drastisch erhöhen. Außerdem läßt sich die Software schneller und billiger entwickeln, da nicht Hunderte von speziellen Funktionen vorab programmiert werden müssen.

Trotz RAP sind eine ganze Reihe von Fragen noch offen. Erstens, ist Matching von Schlüsselphrasen auch bei komplizierteren Aufgaben wirklich ausreichend? Oder wo und wie sollte man es mit natürlichsprachlichem Parsing mischen? Kann man auch schwierigere Aufgabenflüsse mit DBL (allein) synthetisieren? Ist das Automatenmodell immer ausreichend, oder muß man auch Petrinetze vorsehen, z.B. bei der Synchronisation von Bestellwesen und Lagerverwaltung? RAP kann nur als ein erster Versuch gesehen werden, am Beispiel einer einfachen aber *realistischen Aufgabe* aus dem Bürobereich diesen Fragen nachzugehen und die Anforderungen für komplexe Vorgänge herauszuarbeiten.

[10]Die Optimierung dieser Strategie durch ein "Merge" von Zuständen ist angedacht, aber nocht nicht realisiert, da eine wirkliche Notwendigkeit von uns für die RAP-Domäne bisher nicht erkannt wurde. Hier sind keine allzu großen Automaten zu erwarten.

5 Zusammenfassung

Im vorliegenden Artikel wurden zwei Agentensysteme vorgestellt, die – so oder ähnl – auf jeden Fall Bestandteil zukünftiger Büroautomatisierungssoftware sein werden: D Kalendermanager CAP II und der Raumreservierungsagent RAP. CAP II zeigt die Vorte von Agenten, die sich an die Vorlieben ihrer Benutzer anpassen und stellvertretend sie Aufgaben autonom abwickeln können. Eine wichtige Eigenschaft von solchen Agent muß sein, daß sie auch mit Personen verhandeln können, die das Tool nicht besitzen, ab über ein Rechnernetzwerk erreichbar sind. CAP II ermöglicht das durch das semistrukt erte Nachrichtenformat der mit beliebigen Agenten oder Personen ausgetauschten Em

RAP ist ein Experiment zur Erforschung der Mächtigkeit von dialogbasiertem Lern (DBL) bei der Synthese von operationellen Strukturen wie sie für computerunterstütz kooperatives Arbeiten notwendig sind. Drei Gründe sprechen für DBL. Erstens komm in der Bürodomäne nicht schnell genug ausreichend viele Trainingsbeispiele zusamm um die klassischen induktiven Machine Learning Algorithmen einzusetzen. Erklärun basiertes Lernen (EBL) – gern eingesetzt bei wenigen Beispielen – ist, zweitens, au keine Lösung, da die Domäne nicht (ausreichend) vollständig modelliert werden kann bei EBL erforderlich. Der dritte Grund ist, daß bei Bürosoftware der Benutzer die me Zeit anwesend ist; also sollte er auch in das Lernen interaktiv einbezogen werden. Da lassen sich induktives wie deduktives Lernen massiv erleichtern [2].

Die wirtschaftlichen Vorteile unseres Ansatzes sind offensichtlich. Da die Büros ware sich flexibel den Anforderungen verschiedener Benutzer und Organisationen anp wird die Akzeptanz (und die Verkaufszahlen) steigen. Ein weiteres Plus ist die Überna me zeitaufwendiger Routinearbeiten, was dem Benutzer mehr Zeit für kreatives Arbei läßt. Für den Entwickler solcher Bürosoftware werden sich außerdem geringere Kosten u kürzere Time-to-Market Intervalle ergeben, da eine Vielzahl von Optionen und Spez funktionen für jeden denkbaren Benutzer nicht mehr von vornherein einprogramm sein muß. Die heutige Hardware- und Softwaretechnologie im Verbund mit den aktue Ergebnissen von DAI und Maschinellem Lernen stellen die notwendige Basis für so Systeme inzwischen bereit.

An der Carnegie Mellon University läuft die Forschung an CAP II und RAP un dem Arbeitstitel "Software Secretary". Ziel ist, (einen Teil der) SekretärInnenfunkt alität auf einem Rechner verfügbar zu machen, so daß der Eigentümer von langweil und/oder zeitaufwendigen Routinearbeiten verschont wird. Zur Zeit benutzen dort e ein Dutzend Personen den lernfähigen Einplatzkalender CAP. CAP gibt nach circa 2 Monaten zufriedenstellende Ratschläge (bei einer Anzahl von Terminen, wie sie für ei Professor anfallen). CAP II wird zur Zeit nur vom Autor benutzt; alle anderen M glieder der CAP-Gruppe agieren als menschliche Agenten und beantworten die vom C II Agent verschickten Einladungen zu Gruppenbesprechungen "von Hand". Die Freig von CAP II an alle Gruppenmitglieder ist für Ende April 93 geplant.

RAP wird gerade fertiggestellt und soll im Mai 93 zum ersten Mal auf nicht vor warnte Sekretärinnen der CMU "losgelassen" werden.

6 Zukünftige Forschungsaktivitäten

Eine alternative Benutzerschnittstelle auf der Basis von *Electronic Forms (EFORMS)* soll für CAP II implementiert werden. Dabei sollen beim Empfänger von CAP II - Nachrichten automatisch Bildschirmfenster mit Formularstruktur geöffnet werden (remote pop-up windows), in denen Felder angekreuzt bzw. einfach ausgefüllt werden können. Die Frage ist, ob der Benutzer so ein komfortables Interface eher akzeptiert als die (verborgenen) Terminverhandlungen der Agenten im Hintergrund.

Eine anderer Forschungsgegenstand ist das Einbeziehen von *externen Wissensquellen* wie Personaldatenbanken oder auch das `finger` Kommando in Unix, die an CAP II gekoppelt werden sollen. Durch solche Zugriffsmöglichkeiten ließe sich die Zeit für den Aufbau des initial notwendigen Wissens stark verkürzen, was wesentlich weniger Interaktionen (also Störungen) mit dem Benutzer bedeuten würde.

Als weitere Anwendung von DBL soll das induktive Regellernen in CAP beschleunigt werden. Der Benutzer soll die erzeugten Regeln mit dem Agenten diskutieren, ob sie seiner Intention entsprechen. Wenn der Benutzer die Regeln akzeptiert, bekommen sie einen hohen Vertrauenskoeffizient schon nach ganz wenigen Beispielen.

Liegen erst mehr Daten über die Anwendbarkeit von RAP vor, soll auf dieselbe Weise der endliche Automat von CAP II per DBL akquiriert und mit der ersten, "handcodierten" Version verglichen werden.

Zusätzlich zur Erforschung der grundsätzlichen Fragen sollen weitere nützliche Agenten implementiert werden. Beispielsweise RAP-ADMIN, der Raumverwaltungsagent auf der Administratorseite. Neben der Verhandlungsfunktionalität wie in RAP auf der Anfrageseite muß RAP-ADMIN auch das Verteilen der Räume lernen. Ob und wie das gehen soll, ist bis jetzt unklar.

Des weiteren sollen CAP II und RAP zum Agenda APprentice AAP kombiniert werden, der erste Vorschläge für die Zeitpläne von CMU-Besuchern ausarbeiten soll, d.h. ihre Treffen mit CMU-Mitarbeitern koordiniert und einen Raum für ihren Gastvortrag belegt.

Andere Agenten für Sekretariatsaufgaben wie Bestellwesen oder Reiseabrechnung sind in der Diskussion.

Danksagung

Ich möchte mich für eine Reihe wichtiger Diskussionen und Anregungen bei Tom Mitchell, Ryusuke Masuoka, Dayne Freitag und Michael Sassin bedanken. David Zabowski half mir bei der Einarbeitung in die CAP-Software und bei der Bereitstellung des Kommunikationsinterface zum Gnu-Emacs-Prozeß. Anand Chandani und Ratul Puri haben die Email-Parser von CAP II implementiert, Dan Ferrell das Lernen in RAP. Weiterhin möchte ich mich auch bei Sybille Rosenmüller, Gisbert Lawitzky und Wolfgang Rencken in München für ihre Unterstützung der bürokratischen Notwendigkeiten bei Veröffentlichungen in unser Firma bedanken.

Literatur

[1] S. Bocionek, A. Chandani, R. Puri, D. Freitag, R. Masuoka, and T.M. Mitche CAP II: Making the Calendar Apprentice an Agent. Technical Report CMU-CS- to-appear, Carnegie Mellon University, School of Computer Science, Pittsburgh, P 1993.

[2] S. Bocionek and M. Sassin. Dialog-Based Learning (DBL) for Adaptive Interfa Agents and Programming-by-Demonstration Systems. Technical Report CMU-C 93-to-appear, Carnegie Mellon University, School of Computer Science, Pittsburg PA, 1993.

[3] L. Dent, J. Boticario, J. McDermott, T.M. Mitchell, and D. Zabowski. A Person Learning Apprentice. In *National Conference on Artificial Intelligence (AAAI-9* August 1992.

[4] J. Jourdan, L. Dent, J. McDermott, T.M. Mitchell, and D. Zabowski. Interfaces th learn: A learning apprentice for calendar management. Technical Report CMU-C 91-135, Carnegie Mellon University, School of Computer Science, Pittsburgh, P April 1991.

[5] R. Kozierok and P. Maes. A learning interface agent for scheduling meetings. In *I Workshop on Intelligent User Interfaces*, Orlando, Fl, January 1993. ACM-SIGCH ACM Press.

[6] A. Lux. A Multi-Agent Approach towards Group Scheduling. Technical Report Siemens within ESPRIT Project 5362: Imagine, August 1992.

[7] R. Medina-Mora, T. Winograd, R. Flores, and F. Flores. The ActionWorkflow ap proach to workflow management technology. In J. Turner and R. Kraut, editor *CSCW '92 (Sharing Perspectives)*, pages 281 - 288, Toronta, Canada, Novemb 1992. ACM Press.

[8] T.M. Mitchell, J. Allen, P. Chalasani, J. Cheng, O. Etzioni, M. Ringuette, a J. Schlimmer. THEO: A framework for self-improving systems. In K. VanLeh editor, *Architectures for Intelligence*. Erlbaum, 1991.

[9] S. Sen and E.H. Durfee. A formal study of distributed meeting scheduling: Preli nary results. In *Conf. on Organizational Computing Systems*, pages 55 - 68. AC November 1991.

[10] M. Woitass. Coordination of Intelligent Office Agents - Applied to Meeting Sched ing. In S. Gibbs and A.A. Verrijn-Stuart, editors, *Multi-User Interfaces and Ap cations*, pages 371 - 387, Heraklion, Greece, September 1990. IFIP, Elsevier Scien Publishers B.V.

[11] M. Woitass. *Koordination in strukturierten Konversationen*, volume 190 of *GM Berichte*. Oldenbourg Verlag, 1991.

COSMA: Ein verteilter Terminplaner*

Achim Schupeta
Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI)
Stuhlsatzenhausweg 3, D-6600 Saarbrücken, Deutschland
Tel.: ++49 681 302 5320
Fax : ++49 681 302 5341
E-Mail: schupeta@dfki.uni-sb.de

Zusammenfassung

Wie können Aktionen (autonom) geplant und ausgeführt werden, die das gemeinsame gleichzeitige oder zeitlich eng verzahnte Handeln von mehreren Agenten inhärent erfordern? Eine Hauptvoraussetzung zur Lösung dieser Frage sind sicherlich die kommunikativen Fähigkeiten der Agenten [MS91]. Am Beispiel des Problems der Terminvereinbarung versuchen wir ein mehrschichtiges Verhandlungsmodell zu entwickeln. Die *Verhandlungsführung* zwischen den Agenten wird durch ein Verhandlunsprotokoll festgelegt. Das heißt, jeder Agent verfügt über eine Menge von Kommunikationsprimitiven wie sie in der Sprechakttheorie beschrieben werden [Sea69]. Diese werden dann zu einem Verhandlungsprotokoll kombiniert, dessen Ablauf häufig durch endliche Automaten beschrieben wird [CW91]. Entscheidungen des Agenten innerhalb einer Verhandlung werden aufgrund einer Bewertung von Vorschlägen gefällt. Dabei determiniert die Verwendung der Bewertungskriterien die Verhandlungsstrategie. Auf der Ebene der Agentengesellschaft kann man schließlich die resultierenden Kooperationsmechanismen ausmachen. Wir beschreiben im folgenden die Verwendung dieses Modells in dem verteilten Terminplanungssystem COSMA (Cooperative Schedule Management Agent).

1 Die Idee von COSMA

Der verteilte Terminplaner COSMA simuliert das Problem der Terminvereinbarung für mehrere Personen. Die Grundidee ist die eines persönlichen Assistenten, der den Terminkalender seines Chefs verwaltet, ihn an Termine erinnert und die Kompetenz hat, über die Zeitpunkte neuer Termine zu verhandeln. Das heißt ein Auftrag wie "Machen sie mir nächste Woche einen Termin mit Schmidt und Schulze" führt dazu, daß der Assistent aufgrund seines Wissens über den bisherigen Terminkalender und die Präferenzen seines Chefs mit den Assistenten von Schmidt und Schulze über einen konkreten Termin in der nächsten Woche verhandelt. Diese Verhandlung mag mehrere Verhandlungsrunden von Vorschlägen und Gegenvorschlägen umfassen. Einigen sich die Partner schließlich auf einen Termin, so wird dieser jeweils im Terminkalender fixiert. Natürlich können auch bereits festgelegte Termine erneut verschoben oder abgesagt werden. COSMA übernimmt in diesem Szenario die Rolle des Assistenten.

*Diese Arbeit wurde unterstützt vom Bundesministerium für Forschung und Technologie unter der Kennung ITW9104

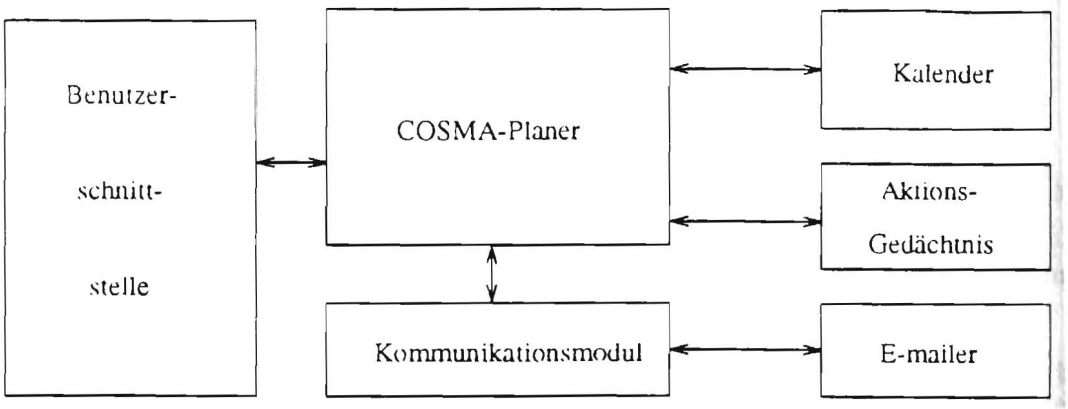


Abbildung 1: Die COSMA Architektur

Ein wesentlicher Unterschied zu den meisten anderen Terminplanern besteht in dem grundlegend verteilten Ansatz: Es gibt keinen zentralen Terminkalender, in dem alle Termine der Firma oder Abteilung gespeichert sind. Jedes COSMA-System verwaltet nur lokale Daten seines Benutzers, die Koordination erfolgt ausschließlich über die Kommunikation. Die im Grunde genommen sehr begrenzte Aufgabe, nämlich Einigung auf einen konkreken Termin für ein Treffen, erweist sich als gutes Fallbeispiel für viele grundsätzliche Fragestellungen in der verteilten KI. Wir wollen hier besonders den Aspekt der Verhandlungsführung herausstellen und ein mehrstufiges Verhandlungsmodell am Beispiel von COSMA darstellen.

2 Die COSMA Architektur

Die Abbildung 1 zeigt die Grobstruktur eines COSMA-Agenten. Der *Kalender* enthält alle Daten über fest vereinbarte Termine. Ein Termin ist beschrieben durch:

- die Teilnehmer
- den Initiator
- den Typ des Treffens
- die Priorität. (Zahl aus [0, 1] mit 0 = unwichtig, 1 = sehr wichtig)
- die Dauer des Treffens
- den Ort
- das Thema
- die Zeit

Der Initiator des Treffens initialisiert alle diese Werte. Die Zeit wird dabei zunächst als ein maximal mögliches Intervall spezifiziert, wobei jedem Teilintervall eine Präferenz

zugeordnet ist. Wie sich die Präferenzstruktur der Zeit ergibt, wird später noch genauer beschrieben. In der momentanen Version ist die Zeit der einzige Gegenstand der Verhandlung. Der Kalender enthält außerdem Weltwissen in Form von allgemein akzeptierten Präferenzen, also z.B. daß es im allgemeinen nicht üblich ist, Termine in der Nacht oder zur Mittagszeit zu vereinbaren. Dazu kommen Benutzer-spezifische Präferenzen, die mit den allgemeinen Präferenzen, den fixierten Termindaten und den Daten aus dem Aktionsgedächtnis zu einem Präferenzfile gemischt werden (vergl. 3.3).

Das *Aktionsgedächtnis* enthält Zeitconstraints, die sich aus parallel laufenden Verhandlungen ergeben. Zeitintervalle, die sich aktuell in Verhandlung befinden, können abhängig von der gewählten Verhandlungsstrategie als für andere Verhandlungen gesperrt markiert sein. Im Aktions-Gedächtnis wird außerdem der aktuelle Zustand, sowie die Historie einer Verhandlung gespeichert.

Das *Kommunikationsmodul* transformiert alle Nachrichten vom internen Format (vergl. 3.1) in natürliche Sprache, die per elektronischer Post verschickt wird. Umgekehrt werden alle eingehenden Nachrichten in das interne Format umgesetzt und an den Planer weitergegeben. Die Kommunikation in nat. Sprache macht COSMA zu einem offenen System, d.h. auch Partner ohne COSMA können an Terminverhandlungen teilnehmen, sofern sie an das E-mail System Zugang haben. Im folgenden abstrahieren wir jedoch von diesem Modul und beschreiben Nachrichten nur bis zur Schnittstelle zwischen Planer und Kommunikationsmodul, da wir hier nicht in den Bereich der nat. Sprachverarbeitung eindringen möchten.

Der *COSMA-Planer* kontrolliert unter Zugriff auf die anderen Module die Verhandlungsführung des Agenten. Es werden Nachrichten im jeweiligen Verhandlungskontext verschickt und empfangen, bewertet und verglichen und Entscheidungen gemäß der Verhandlungsstrategie gefällt. Dies wird im Abschnitt 3 genauer beschrieben.

Zur komfortableren Handhabung für den Benutzer ist zusätlich eine window-orientierte *Benutzer-Schnittstelle* vorgesehen.

3 Das Verhandlungmodell

Die Verhandlungsführung steht im Mittelpunkt unseres Interesses. Wir strukturieren die zur Verhandlungsführung nötigen Fähigkeiten des Agenten in Ebenen unterschiedlicher Komplexität. Wir unterscheiden hier 5 Ebenen, die auf der durch die Anwendung vorgegebenen Problemstellung aufbauen, wobei sich diese Zahl jedoch durch Abstraktion oder Verfeinerung verringern oder vergrößern kann. Auf der untersten Ebene stehen die ausgetauschten *Nachrichten*. Alle möglichen Typen von Nachrichten, ihr Format und ihre Bedeutung müssen festgelegt werden. Darüber liegt die *Protokollebene*, die festlegt welche Sequenzen von Nachrichten in einem Dialog möglich sind. Ein Protokoll kann aufgefaßt werden als Einschränkung aller Nachrichtenfolgen und bestimmt in jeder Situation einen Dialogkontext. Die Menge aller möglichen Nachrichtenseqeunzen wird weiter eingeschränkt auf die tatsächliche Abfolge indem dort, wo der Kontext mehrere Alternativen zuläßt, eine Entscheidung für eine von ihnen getroffen wird. Dies geschieht auf der *Verhandlungsebene*, wobei zur Fällung der Entscheidung auch auf die Bewertungsebene zurückgegriffen werden muß, denn die Grundlage für Entscheidungen sind Bewertungen von vorgeschlagenen oder selbst generierten (Teil-)Lösungen. Die *Bewertungsebene* muß dafür eine oder mehrere Kriterien zur Verfügung stellen. Die Verhandlungsebene legt dann fest, auf welche Weise diese Kriterien für eine Entscheidung verwendet werden und determiniert damit die Verhandlungsstrategie. Welche Effekte solche Verhandlungsstrate-

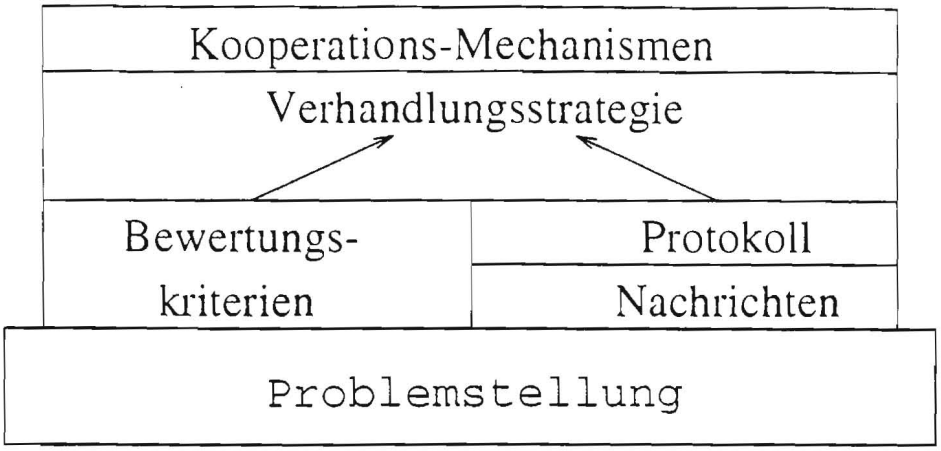


Abbildung 2: Ebenen des Verhandlungsmodells

gien dann auf der Ebene der Agentengesellschaft haben wird auf der *Kooperationsebene* beschrieben.

Die Abbildung 2 zeigt schematisch dieses Modell, das wir in den folgender Abschnitten anhand des COSMA-Systems exemplarisch in seinen einzelnen Ebenen beschreiben werden.

3.1 Die Nachrichtenebene

Auf dieser Ebene wird festgelegt welche Nachrichtentypen es geben soll und wie eine Nachricht aufgebaut ist. Der Aufbau ist natürlich durch die Informationen, die mit der Nachricht übertragen werden sollen bestimmt und somit sehr stark von dem jeweiligen Verhandlungsgegenstand abhängig. Die folgende Liste enthält die in COSMA verwendeten Nachrichtentypen und deren Bedeutung:

- **ARRANGE**: Initiierung 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.
- **REJECT**: Endgültige Ablehnung einer Anfrage nach Terminvereinbarung.
- **CANCEL**: Absage eines bereits bestehenden Termins.
- **PERSUADE**: Spezielle Nachfrage zu einem bestimmten Terminvorschlag bei gleichzeitiger Übermittlung der eigenen Bewertung desselben.
- **COND-ACCEPT**: Bedingte Zustimmung in Abhängigkeit von der Bewertung durch die anderen Teilnehmer, die dem Agenten noch nicht bekannt ist.

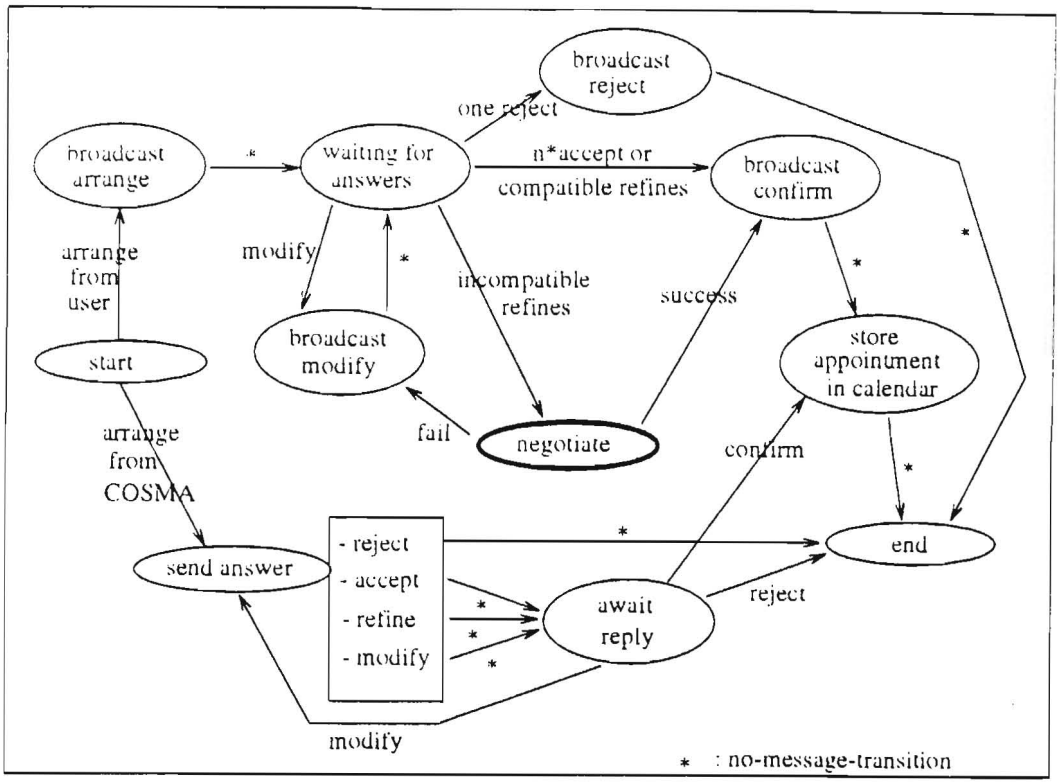


Abbildung 3: Darstellung des Protokolls als Graph (Ausschnitt)

3.2 Die Protokollebene

Sie legt fest, welche Abfolgen von Nachrichten möglich sind. Eine Antwort z.B. ist nur sinnvoll im Kontext einer Frage. Wir modellieren das Protokoll als Menge möglicher Aktionen innerhalb eines Verhandlungskontextes. Die Ausführung von Aktionen führt jeweils zu einem neuen Kontext. Gemäß der Grundannahme der Sprechakt-Theorie [Sea69] ist dabei das Verschicken und Empfangen von Nachrichten ebenfalls als Aktion anzusehen.

Abbildung 3 zeigt ausschnittsweise eine graphische Repräsentation unseres Protokolls für COSMA. Die Knoten stehen dabei für Aktionen, die der Agent ausführt oder - wie im Falle des mit "nagotiate" markierten Knotens - für Subgraphen, die eine Unterverhandlung über ein spezielles Teilproblem (hier: ein bestimmtes Zeitintervall) darstellen. Die Pfeile stehen für die externen Aktionen, d.h. konkret für die von anderen Agenten gesandten Nachrichten.

Die Grundidee des Protokolls für COSMA ist sehr einfach: Der Initiator eines Treffens verschickt eine ARRANGE-Nachricht an die Teilnehmer. Diese reagieren mit Ablehnung (REJECT), Zustimmung (ACCEPT), einer Verfeinerung des vorgeschlagenn Zeitraumes (REFINE) oder einem Gegenvorschlag (MODIFY). Falls alle Partner mit Zustimmung reagieren oder die jeweiligen Verfeinerungen unter einen Hut zu bringen, sind fixiert der Initiator den Zeitpunkt des Treffens durch eine CONFIRM-Nachricht an alle. Die Ablehnung eines Partners reicht aus um das Treffen unmöglich zu machen (REJECT). (Der Initiator muß in diesem Fall entscheiden, ob er das Treffen mit einer modifizierten Teilnehmerliste nochmals zur Diskussion stellen will.) Bei Gegenvorschlägen muß der Initiator

entweder den gemachten, oder einen selber errechneten Gegenvorschlag an alle Partner weitergeben und wie am Anfang deren Antworten abwarten. Bei mehreren unvereinbaren REFINE-Nachrichten könnte der Initiator wie bei MODIFY verfahren, aber wir haben für diesen Fall eine Unterverhandlung ("negotiate") eingerichtet. Dabei werden für die einzelnen Teilintervalle des Vorschlages die Partner nochmals befragt, wobei ihnen die Bewertungen des Vorschlages durch die anderen Partner offengelegt werden. Aufgrund dieser Informationen können die Agenten in Abhängigkeit von ihrer Verhandlungsstrategie (vergl. 3.4.) ihre zu Anfang gemachte Bewertung verändern, und es kann doch noch zu einer Einigung kommen. Grundlage für die Bewertung von Vorschlägen sind die Priorität des Treffens und die Präferenzen, die ein Agent für unterschiedliche Zeitintervalle hat. Um die durch das Protokoll offengelassenen Entscheidungen in einer Verhandlung zu fällen wird also auf die Bewertungsebene zurückgegriffen, die im folgenden Abschnitt näher erläutert werden soll.

3.3 Die Bewertungsebene

Die Bewertungsebene ist sehr stark von der jeweiligen Anwendung abhängig. In unserem Beispiel liefert sie der Verhandlungsebene 2 Kriterien, die für die dortige Entscheidung verwendet werden: Erstens den *Nutzen*, den ein Agent einem Terminvorschlag zuordnet und zweitens den *Faktor der Kompromissbereitschaft* der bei einer Subverhandlung über ein spezielles Teilintervall die Nachgiebigkeit des Agenten bestimmt.

3.3.1 Nutzen

Die Bewertung von Vorschlägen in COSMA beruht auf einem einfachen Zeitmodell, bei dem mit jedem Zeitpunkt eine Präferenz assoziiert ist. Dies ist eine reelle Zahl aus dem Intervall [0, 1], die sozusagen den Auslastungsgrad eines Zeitpunktes angibt: 0 bedeutet, daß hier noch Termine akzeptiert werden können, 1 bedeutet, daß dieser Zeitpunkt schon fest belegt ist, und jeder Wert dazwischen gibt den Grad der Akzeptanz an mit dem neue Termine angenommen werden können.

Die Priorität eines Treffens wird durch den Initiator vorgegeben. Die Zeitpräferenzen dagegen werden von mehreren Faktoren beeinflußt und müssen in der Bewertungsebene ständig aktualisiert werden. Dabei fließt Information aus 4 Quellen zusammen:

- **Weltwissen**: Allgemein anerkannte Präferenzen also z.B. geringe Neigung, Termine in der Nacht zu vereinbaren.
- **Benutzervorlieben**: Spezielle Präferenzen, die die Eigenheiten des Benutzers berücksichtigen.
- **Kalenderdaten**: Die Prioritäten von bereits fest vereinbarten Terminen.
- **Aktionsgedächtnis**: Vorschläge von laufenden Verhandlungen können Zeitintervalle für andere parallel laufende Verhandlungen sperren.

Wie in Abbildung 4 gezeigt, werden Weltwissen und Benutzervorlieben mit Hilfe einer Durchschnittbildung vereinigt und die "harten Constraints" vom Kalender und Aktionsgedächtnis durch Maximumbildung vereint. Schließlich wird durch Maximumbildung die endgültige Zeit-Präferenz Struktur gebildet.

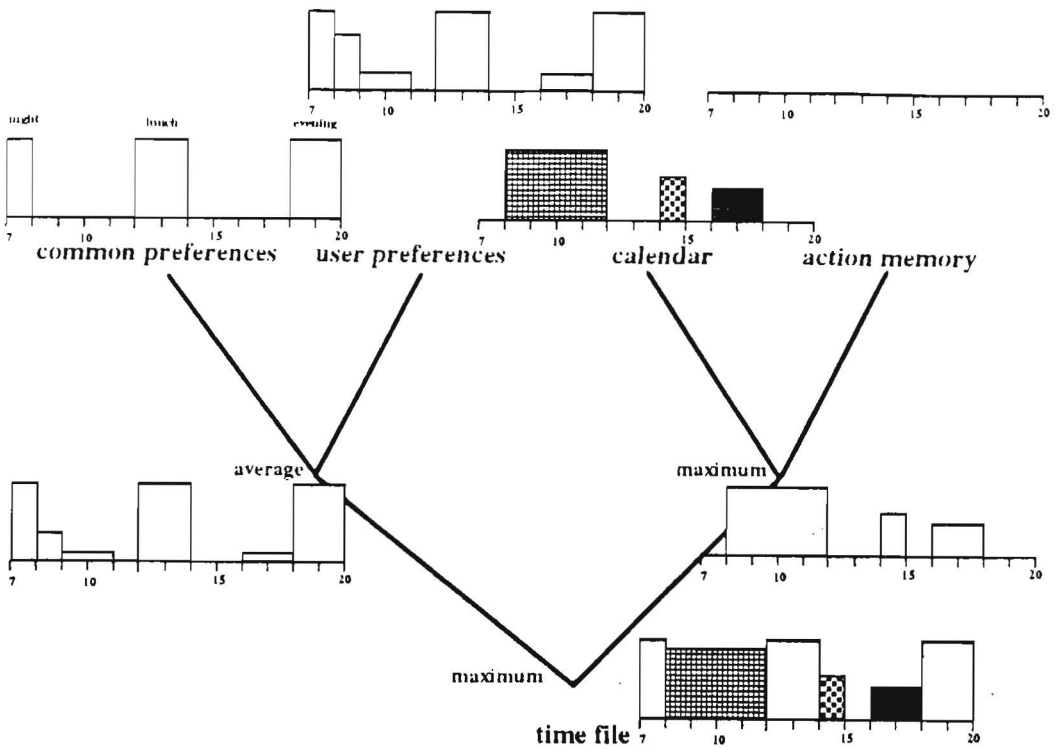


Abbildung 4: Berechnung der Zeitpräferenzen

Wir nennen nun die Differenz aus der Priorität eines Treffens m und der Präferenz eines Zeitintervalls von t bis $t + d$ den **Nutzen** des Agenten, den dieser hat, wenn das Treffen m mit der Dauer d zum Zeipunkt t beginnt:

$$Nutzen(m, t) = Prio(m) - Präf(t, t + d)$$

Der Nutzen ist das erste Kriterium, welches die Bewertungsebene für die Entscheidungen der Verhandlungebene bereitstellt.

3.3.2 Faktor für die Kompromißbereitschaft

Dieser Faktor soll die Kompromißbereitschaft des Agenten in der Unterverhandlung über ein spezielles Teilintervall steuern. Ziel soll es sein, die Zugeständnisse, die die Agenten zur Lösung von Konflikten eingehen, über eine Menge von mehreren Verhandlungen hinweg, innerhalb der Agentengesellschaft gleichmäßig zu verteilen. (Vergl. 3.5.)

Dazu stellt die Bewertungsebene ein sehr simples Partnermodell bereit, indem sie für jeden Partner X mit dem sie je einen Termin verhandelt hat sich dessen Nutzen N_X sowie den eigenen Nutzen N_E^X aus diesem Termin merkt. Bei einem erneuten Termin, an dem auch X teilnimmt, werden diese beiden Werte (nach einer erfolgreichen Verhandlung!) auf einfache Weise aktualisiert:

$$N_X \leftarrow \frac{N_X + n_x}{2} \quad ; \quad N_E^X \leftarrow \frac{N_E^X + n_e}{2}$$

Dabei ist n_x der Nutzen von X und n_e der eigene Nutzen bei diesem Termin. Auf diese Weise braucht man sich nicht die Daten von allen bisherigen Treffen mit X zu merken

und hat den gewollten Nebeneffekt, daß die älteren Daten aufgrund der iterativen Durchschnittbildung verblassen, während die aktuelleren Werte stärker gewichtet werden. Der eigentliche Faktor der Kompromissbereitschaft C wird nun als Differenz aus dem durchschnittlichen Nutzen und dem eigenen Nutzen errechnet. Wird etwa um einen Termin mit X, Y und Z verhandelt, so berechnet der Agent:

$$C = N_d - N_E \quad \text{wobei} \quad N_d = \frac{N_E + N_X + N_Y + N_Z}{4} \quad \text{und} \quad N_E = \frac{N_E^X + N_E^Y + N_E^Z}{3}$$

Ein positives C bedeutet, daß der Agent im Vergleich zu seinen Partnern bisher weniger Nutzen aus den Terminen gezogen hat. Seine Kompromißbereitschaft in zukünftigen Verhandlungen sollte geringer werden. Doch hier sind wir bereits mitten in der Diskussion der Verhandlungsebene, die festlegt auf welche Art und Weise die Kriterien der Bewertungsebene für die Entscheidungsfindung verwendet werden:

3.4 Die Verhandlungsebene

Die Verwendung der Bewertungskriterien bei den zu treffenden Entscheidungen resultiert in unterschiedlichen Verhandlungsstrategien. Der Nutzen ist definiert als Differenz aus Priorität des Treffens und Präferenz des Agenten für einen bestimmten Zeitraum. Ein positiver Nutzen bedeutet, daß die Priorität des Treffens die Präferenz überwiegt. Der Faktor der Kompromißbereitschaft quantifiziert die relativen Zugeständnisse, die der Agent in bisherigen Verhandlungen gemacht hat. Unter Verwendung dieser beiden Kriterien haben wir mit COSMA zunächst einmal drei relativ einfache Verhandlungsstrategien untersucht, die die Zustimmung oder Ablehnung zu einem vorgeschlagenen Termin determinieren:

- **Ablosute Entscheidung**: Der Agent stimmt dem Vorschlag zu, wenn sein Nutzen N positiv ist:
$$N > 0$$
- **Relative Entscheidung**: Der Agent stimmt dem Vorschlag zu, wenn sein Nutzen N größer ist als der durchschnittliche Nutzen N_{av} aller Verhandlungspartner:
$$N > N_{av}$$
- **Entscheidung, die bisherige Treffen berücksichtigt**: Der Agent stimmt dem Vorschlag zu, wenn sein Nutzen N vermindert um den Faktor der Kompromißbereitschaft C größer ist als der durchschnittliche Nutzen N_{av} aller Verhandlungspartner:
$$N - C > N_{av}$$

Die Simulationsexperimente zeigen, daß die Termine bei absoluten und relativen Entscheidungen nicht so dicht gepackt sind wie bei der dritten Strategie. Es kommt aber bei ihr sehr viel öfter zur Absage von bereits fixierten Terminen, weil sie durch neue Termine höherer Priorität verdrängt wurden. Über sie muß dann erneut verhandelt werden. Dies kann je nach der Struktur des bisherigen Kalenders zu einem "Schneeballeffekt" führen. Allerdings wird die Zeit der Agenten so besser ausgenutzt. Abhilfe könnte eine Modifikation bei der Berechnung der Präferenzen geben: Fixierte Termine sollten mit einer vergrößerten Priotität gewichtet werden. Andererseits war die Möglichkeit der Verdrängung von Terminen niedrigerer Priorität durch solche mit höherer Priorität ja einer der wesentlichen Motivationen um die Prioritäten überhaupt einzuführen. Ein modifizierter Mechanismus sollte

diese Möglichkeit also auch nicht zu stark einschränken. Eine andere Möglichkeit, dieses Problem anzugehen besteht in der Modellierung eines weiteren Bewertungskriteriums, das in die Entscheidung mit einfließt und den aus den Neuverhandlungen entstehenden Aufwand abschätzt.

Schließlich wollen wir noch darauf hinweisen, daß die grundsätzliche Verhandlungsmethode natürlich schon durch die Protokollebene determiniert ist und nicht auf dieser Ebene verändert werden kann. So kann z.B. mit unserem Protokoll nicht eine Strategie erzwungen werden, die die Rolle des Initiators des Treffens als Moderator der gesamten Verhandlung umgeht. Dies kann nur durch ein modifiziertes Protokoll eingeführt werden.

3.5 Die Kooperationsebene

Diese Ebene beschreibt die Effekte der Verhandlungsstrategien, die erst bei der Betrachtung der gesamten Agentengesellschaft auftreten. Eigentlich ist diese Ebene also eine Meta-ebene, weil sie emergent aus dem Zusammenwirken der Agenten entsteht. Ein Hauptproblem in der gesamtes VKI-Forschung in diesem Zusammenhang ist die Frage: Welches globale Verhalten auf der Ebene der Gesellschaften resultiert aus dem lokalen Agentenverhalten? Oder um das Problem noch etwas zu verdeutlichen: Wie kann man aus der Kenntnis des lokalen Verhaltens der einzelnen Agenten das emergente Verhalten der Gesellschaft vorhersagen?

Wir haben versucht für COSMA den Faktor der Kompromißbereitschaft so zu berechnen (vergl. 3.3.2.) und zu verwenden (vergl. 3.4.), daß der Nutzen, den die Agenten ihren Terminen zuordnen über eine Historie von mehreren Verhandlungen und über alle Mitglieder der Gesellschaft gleichmäßig verteilt wird. Der intendierte Kooperationsmechanismus sollte ein Verhalten simulieren, wie man es natürlicherweise auch in Terminverhandlungen zwischen Menschen findet: Hat ein Agent bei einer Verhandlung eine hohe Kompromißbereitschaft gezeigt die zu einer Konfliktlösung führte, so ist bei einer nächten Verhandlung ein anderer Agent an der Reihe sich so zu verhalten. Dieser Effekt war bei unseren Beispielverhandlungen bei der dritten Strategie gegenüber den ersten beiben auch gut zu beobachten.

Ein Problem, daß wir jedoch vor weiteren Simulationsexperimenten angehen müssen ist die quantitative Darstellung der Effekte auf der Kooperationsebene. Desweiteren haben wir bisher nur Agenten mit der gleichen Verhandlungsstrategie betrachtet. Wie interagieren Agenten wenn sie unterschiedliche Verhandlungsstrategien verwenden?

3.6 Schlußbemerkungen

Wir haben die verschiedenen Aspekte der Verhandlungsführung in einem Verhandlungsmodell strukturiert und sie an einem konkreten Beispiel - der Terminverhandlung zwischen COSMA-Agenten - dargestellt. Die Reihenfolge der Darstellung und die Anordnung der einzelnen Ebenen im Modell (vergl. Abb. 2) soll keineswegs eine Implementierungsstrategie für andere Anwendungen empfehlen. Die gesamte Verhandlung muß als Ganzes entworfen werden. Insbesondere sind die verschiedenen Schichten ja nicht unabhängig voneinander. Zum Beispiel wurde der Nachrichtentyp COND-ACCEPT erst nachträglich eingefügt, als wir die relative Verhandlungsstrategie entwickelten.

Einige Aspekte des Modells sind zwangsläufig stark auf die Problemstellung der Anwendung ausgerichtet und sehen bei anderen Systemen ganz anders aus. Trotzdem erscheint uns das Modell allgemein genug, um in anderen Systemen Anwendung zu finden. Eine weitere Verfeinerung und Formalisierung erscheint uns vor allem auf der Nachrichtenebene

und der Protokollebene sinnvoll. So könnte man z.B. die Nachrichtentypen in Klassen u Hierarchien gruppieren und das Konzept der Subprotokolle, wie es in COSMA bereits a gedacht wurde, weiter entwickeln. Dadurch könnte man die Flexibilität weiter erhöhe um etwa ein "Springen aus dem Kontext" wie es bei menschlichen Verhandlungspartner vorkommen mag abzufangen. Wegen der Konzeption von COSMA als offenes Syste (Schnittstelle für nat. Sprache) bietet sich eine solche Vorgehensweise gut an.

Literatur

[CW91] M.K. Chang and C.C. Woo. Sanp: A communication level protocol for negotiations. I D.D. Steiner and J. Müller, editors, *3rd European Workshop on Modelling Autonomou Agents and Multi-Agent Worlds*. DFKI, Kaiserslautern, 1991. D-91-10.

[MS91] J. Müller and J. Siekmann. Structured social agents. In W. Brauer and D. Hernandez editors, *Verteilte Künstliche Intelligenz und Kooperatives Arbeiten*, pages 42–52, München October 1991. Proc. 4. Internationaler GI-Kongress "Wissensbasierte Systeme", Springer Verlag.

[Sea69] J. R. Searle. *Speech Acts*. Cambridge University Press, 1969.

VERTEILTES PARSING NATÜRLICHER SPRACHE

Udo Hahn, Susanne Schacht & Norbert Bröker

Arbeitsgruppe Linguistische Informatik / Computerlinguistik
Albert-Ludwigs-Universität Freiburg
Friedrichstr. 50
7800 Freiburg i. Brsg.

{hahn,sue,nobi}@coling.uni-freiburg.de

Abstract

In diesem Beitrag wird ein Modell für das verteilte Parsing von natürlichen Sprachen beschrieben. Seine wesentlichen Eigenschaften sind die verteilte Organisation linguistischen Wissens in Form einer Lexikon-Grammatik, die objektorientierte Spezifikation der Struktur des Lexikons mittels Vererbungsmechanismen und der Kommunikation zwischen lexikalischen Einheiten durch ein linguistisch valides Botschaftsprotokoll sowie die Implementation des Modells im Rahmen einer physisch verteilten Smalltalk-Plattform, die asynchrone Kommunikation zwischen einzelnen Prozessen erlaubt. Ausführlich wird auf das Design der Lexikonhierarchie und der Botschaftskonstrukte des verteilten Parsers eingegangen.

1. Einführung

In der theoretischen Linguistik wie in der Computerlinguistik vollzieht sich zur Zeit eine deutliche Abkehr von regelorientierten hin zu lexikonorientierten Systemen zur grammatischen Beschreibung natürlicher Sprachen. Im Rahmen dieser *Lexikalisierungsbestrebungen* wird linguistisches Wissen aus homogen strukturierten Regelsystemen (etwa Phrasenstruktur- oder Baumtransformationsgrammatiken) in strukturell eher heterogene Spezifikationen lexikalischer Einheiten verlagert. Die kombinatorischen Beschränkungen zwischen lexikalischen Elementen werden unter diesen Modellannahmen neben lexikalisch kodierten Merkmalsspezifikationen durch globale Wohlgeformtheitsbedingungen (Constraints, Prinzipien, Filter o.ä.) ausgedrückt, die den Aufbau valider linguistischer Strukturbeschreibungen beschränken. Dies führt folgerichtig zu einer zunehmenden *Modularisierung* in dem Sinne, daß verschiedene Dimensionen sprachlicher Regularitäten (Kasusrollen, Phrasenstrukturen, Wortfolge) in jeweils eigenen, logisch voneinander jedoch abhängigen Moduln beschrieben werden (exemplarisch etwa in der Government-Binding-Theorie; vgl. Berwick et al. [1991]). Es ist offensichtlich, daß der gemeinsame Nenner dieser Entwicklungen die *Verteilung* linguistischen Wissens ist, wobei die "natürliche" Dekompositionseinheit solcher Lexikon-Grammatiken die Wörter (genauer: Morpheme) einer Sprache sind. Auffällig ist dabei, daß selbst in der mit der Algorithmisierung linguistischer Theorien befaßten Computerlinguistik bislang kaum Beiträge entwickelt worden sind (vgl. Abschnitt 4), die den aus der Verteilung grammatischen Wissens erwachsenden *Kommunikationsanforderungen* durch geeignete Konstrukte auf der Modellierungs- oder Implementationsebene gerecht werden.

Solche Anforderungen fallen um so stärker ins Gewicht, je anspruchsvoller die Sprachverstehensaufgabe formuliert wird, d.h. Sprachverstehen nicht allein als linguistische Strukturanalyse (Parsing), sondern als integrierter Verstehensprozeß verstanden wird, in den auch nicht-sprachliche Wissensquellen eingehen. Dies wird bereits deutlich in Anwendungen wie dem Verstehen gesprochener Sprache oder dem Textverstehen, die durch eine Vielzahl miteinander interagierender Teilprozesse (etwa Phonemsegmentation, Postulierung und Evaluierung von Worthypothesen, Ambiguitätsbehandlung, Referenzauflösung, Fokusie-

rung usw.) charakterisiert sind. Unter diesen Bedingungen erweisen sich die im (computer)linguistischen Kontext verbreiteten streng hierarchischen Kontrollmuster innerhalb von und zwischen verschiedenen linguistischen Modellebenen (etwa: Syntax vor Semantik, Semantik vor Pragmatik) als kontraproduktiv, da sie wenig Flexibilität, z.B. im Umgang mit Ambiguitäten, erlauben. Diesen Beschränkungen auf der Modellebene entspricht auf der Implementationsseite die Realisierung von (primär syntaktisch orientierten) Parsern durch monolithische Systeme, die grammatische Beschreibungen anhand einer globalen, meist starren seriellen Kontrollstruktur (*depth-first*-Suche, Backtracking) interpretieren. Zusätzlich verschärft wird die Problematik zu rigider Steuerungsmechanismen im Rahmen von Systemen, die mit *authentischem* Sprachmaterial (etwa unverändert übernommenen Zeitschriftentexten oder realen Dialogsequenzen) arbeiten. Hierbei müssen lexikalische bzw. grammatische Spezifikationslücken, fehlerhafte Eingaben, unvollständige Daten u.ä. angemessen behandelt werden, was die Anforderungen an die Flexibilität der Interaktion einzelner Systemkomponenten weiter erhöht.

Nicht nur unter Gesichtspunkten des Entwurfs komplexer Sprachverstehenssysteme, sondern auch vor dem Hintergrund der Ergebnisse psycholinguistischer Studien ist die Postulierung autonomer, strikt seriell organisierter Sprachverarbeitungsprozesse [Forster 1979, Frazier 1987] durch gegenteilige experimentelle Evidenzen erschüttert worden und somit pauschal nicht länger haltbar. Vielmehr scheint der Einfluß unterschiedlicher Ebenen der Sprachverarbeitung aufeinander nurmehr dahingehend unterscheidbar, ob man eine *schwache* [Altmann & Steedman 1988] oder eine *starke Interaktivität* [Marslen-Wilson & Tyler 1980, Thibadeau et al. 1982] zwischen den einzelnen Komponenten des menschlichen Sprachprozessors zugrunde legt. Ob diese Interaktion im Sinne einer gezielt gesteuerten Nachfrage nach zu liefernden Informationen (als Konsultierungsprozesse) oder einer simultanen, wechselseitigen Beeinflussung stattfindet, ändert wenig an der grundlegenden Annahme, daß Sprachverstehen in beiden Modellvarianten als ein *paralleler* Prozeß verstanden wird. Serielle autonome Modelle - wie sie in der (computer)linguistischen Theoriediskussion vorherrschen - sind auch kognitiv unplausibel.

Um den aus der Lexikalisierung und Modularisierung linguistischen Wissens sowie den aus Architekturprinzipien kognitiv plausibler Systeme erwachsenden Anforderungen gerecht zu werden, entsteht derzeit in der Arbeitsgruppe Linguistische Informatik / Computerlinguistik der Universität Freiburg ein verteiltes Sprachverstehenssystem, das die der strikten Lexikalisierung grammatischen Wissens innewohnende feine Verteilungsgranularität mit Modellierungs- und Designprinzipien objektorientierter Systeme verbindet. Damit wird der Anspruch verbunden, den oben erwähnten besonderen Kommunikationsanforderungen sowohl modell- als auch implementationsseitig gerecht zu werden und einen angemessenen Rahmen für die Ausführung impliziter Parallelität bereitzustellen. Die Verteilung von Wissen orientiert sich einerseits an den einzelnen Ebenen des Sprachverstehens (Grammatik, Lernheuristiken, Domänen- sowie Diskurswissen), andererseits zielt sie auch auf die Verteilung von Wissen innerhalb dieser einzelnen Komponenten; entsprechend kann in einem solchen Kontext zwischen *Inter-* und *Intrakomponenten-Parallelität* sprachverstehender Systeme [Hahn & Adriaens 1991] unterschieden werden. Im Parser - auf den wir uns im folgenden konzentrieren - wird grammatisches Wissen über Morphosyntax, Wortfolge, Kongruenz, Kasusforderungen etc. lexikalisch verteilt spezifiziert. Zur Realisierung einer flexiblen Kontrollstruktur und zur Integration der verschiedenen Wissenssysteme werden *Wortaktoren* eingeführt, die das Verhalten konkreter lexikalischer Einheiten repräsentieren. Diese Form der Spezifikation lehnt sich eng an Hewitts Aktorenmodell [Hewitt & Baker 1978] an. Sprachverstehen wird dabei als Kommunikationsprozeß zwischen Wortaktoren modelliert, die einzelne Textwörter repräsentieren und durch Kooperation miteinander das inhaltliche Verstehen eines Textes simulieren. Wegen vielfältiger Störeinflüsse (Unterspezifikation der Grammatik, inkrementelle Lernaufgaben, ungrammatische Eingabe) kann die Grammatikalität einer Äußerung nicht pauschal festgestellt werden. Vielmehr werden zur Sicherung der Robustheit der Analyse notwendige und hinreichende Kriterien für die Zulässigkeit bestimmter linguistischer Analysen überprüft. Für eine Nominalphrase kann dies etwa bedeuten, daß ihre konzeptuellen bzw. Kasusrollenforderungen strikt erfüllt sein müssen, bestimmte Wortstellungs- oder Kongruenzverletzungen jedoch toleriert werden. Zusätzlich sind die Parser-Spezifikationen, die die Kombination einzelner Anfrageergebnisse beschreiben, um Verhaltsvorschriften zu ergänzen, die den Umgang mit unvollständigen oder widersprüchlichen Daten betreffen. Durch entsprechende Vorkehrungen wird sichergestellt, daß ein Parsing-Prozeß nicht durch fehlende Spezifikationen oder inkonsistente Werte abgebrochen wird, sondern nur

zu eingeschränkten Verstehensleistungen führt (Prinzip der *graceful degradation*). Eine weitere Konsequenz, die sich aus diesen Rahmenbedingungen ergibt, ist die inhärente Beschränkung des Parsers auf ein letztlich nur *partielles* Verstehen der analysierten Texte. Die konkrete Tiefe der Verstehensleistung ist in jedem Falle abhängig von den jeweils verfügbaren grammatischen und konzeptuellen Spezifikationen sowie den der inkrementellen Erweiterung des Konzeptsystems dienenden Lernheuristiken.

2. Die Modellebene

2.1 Wortaktoren

Zwei Entwicklungsrichtungen der objektorientierten Modellierung werden bei der Spezifikation von Wortaktoren vereint. Aus dem klassischen objektorientierten Programmiersprachenparadigma, wie es etwa Smalltalk [Goldberg & Robson 1989] repräsentiert, werden die Strukturierungsmöglichkeiten einer Objektmenge durch Klassen- und Klassenhierarchiebildung (Vererbung) übernommen. Dadurch können sowohl die (Daten-)Struktur als auch das Verhalten eines Objektes aus übergeordneten Klassen ererbt werden. Ein solcher Ansatz unterstützt unmittelbar den Entwurf, die Konsistenz und die Wartbarkeit größerer Objektsysteme. Das Aktorenmodell stellt hingegen eine formal ausgearbeitete Spezifikationsplattform dar, auf deren Grundlage logisch voneinander unabhängige und damit potentiell nebenläufige Prozesse in Gestalt von Aktoren formuliert werden können. Die Kommunikation zwischen und Steuerung von Aktoren beruht ausschließlich auf dem Austausch von Nachrichten (*message passing*); beim Eintreffen einer Nachricht reagiert ein Aktor mit einem Verhalten, das in einem *Skript* (Methode) spezifiziert ist.

Unter dem Blickwinkel des Parsing natürlicher Sprachen wird dieses allgemeine Aktorenkonzept in folgender Form spezialisiert: Ein Wortaktor repräsentiert ein Wort und ist mit dem grammatischen Wissen ausgestattet, das über dieses Wort bekannt ist. Wortaktoren sind also keine prozeßtechnischen Hülsen, sondern stellen bereits direkt einen Teil des für das Textverstehen benötigten Wissens (das eigentliche Sprachwissen) zur Verfügung. Dieses Wissen ist durch eine Reihe von Attributen, den lokalen Zustand des Wortaktors und durch seine Reaktionsweisen auf eintreffende Nachrichten in Skripten beschrieben. Es determiniert sein Verhalten in bestimmten sprachlichen Kontexten. So kann etwa *"der"* die Rolle des definiten Artikels einer Nominalphrase, eines Pronomens oder eines Relativpronomens besetzen und wird für diese drei unterschiedlichen Konstellationen jeweils verschiedene Verhaltensmuster beim Parsing aktualisieren. Ein Wortaktor ist ein selbständiger Prozeß, der über Nachrichten mit anderen, ihm bekannten Wortaktoren oder mit anderen Wissensquellen (wie Diskurs- und Weltwissen) durch einen festgelegten Satz von Interaktionsprimitiven kommuniziert. Er verfügt über einen nur ihm zugänglichen inneren Zustand (*data encapsulation*) und kann diesem entsprechend auf eingehende Nachrichten reagieren. Die Bekanntschaften (*acquaintances*) eines Aktors beschreiben diejenige endliche Teilmenge im Aktorenset, mit der ein Wortaktor zur Bestimmung der inhaltlichen Repräsentation des Textes während des Analyseprozesses zusammenarbeitet (das Aktorenmodell sieht kein *broadcasting* vor).

2.2 Die Aktorenhierarchie

Die Wortaktoren werden in einer Hierarchie beschrieben, so daß Generalisierungen über Klassen von lexikalischen Einheiten formuliert werden können. Diese Hierarchie ähnelt auf den ersten Blick gängigen Wortartspezifikationen. Wir differenzieren diese bekannte Schablone unter den Anforderungen objektorientierter Beschreibungen jedoch auf zwei Ebenen weiter aus: *Grammatische Merkmale* werden in Form von (vordefinierten) Attributen beschrieben, das *grammatische Verhalten* durch Methodenskripte, die die Reaktion der Aktoren auf bestimmte Nachrichten beschreiben. Dabei ist es möglich, Default-Vorgaben in Superklassen in spezielleren Klassen zu überschrieben. Multiple Vererbung ist in unserem Modell vorgesehen, aber noch nicht spezifiziert.

Für die Klassenbildung können die folgenden Spezialisierungsformen von Lexikoneinträgen genutzt werden[1]:

1. **Grammatische Parameter.** Neue Parameter werden eingeführt (z.B. Kasus und Genus für Nominale) oder ein Parameter wird auf einen bestimmten Wert(ebereich) eingeschränkt (z.B. Bestimmtheit auf [+definit] für den bestimmten Artikel). Diese Parameter werden in Attributtermen (mit Koreferenzen und mengenwertigen Attributen) beschrieben.

2. **Relationenbestand.** Eine neue Dependenzrelation kommt hinzu, wie es bei der Unterscheidung von intransitiven und transitiven Verben (ohne/mit zusätzlichem Akkusativobjekt) der Fall ist. In diesem Fall erweitert sich oft das Verhaltensrepertoire der Wortklasse, wie es z.B. bei transitiven Verben der Fall ist, die eine Valenz für eine Nominalphrase im Akkusativ besitzen und gleichzeitig passivierbar sind.

3. **Dependenzrelation.** Eine Dependenzrelation wird in ihren Bedingungen spezialisiert (was letztendlich zu einer Hierarchie von Relationen führt). So ist z.B. das SUBJekt eines Verbs nur in den grammatischen Parametern Kasus, Numerus und Person eingeschränkt, während bei Handlungsverben (*laufen, geben*) eine speziellere AGENS-Relation auch die Belebtheit des Modifikators fordert.

4. **Verhalten.** Eine neue Nachricht wird eingeführt, auf die nur Wortaktoren einer bestimmten Klasse reagieren können (z.B. die Antezedenssuche bei Nomen, aber nicht bei Pronomen).

Zur Illustration dieser allgemeinen Prinzipien betrachten wir einen Ausschnitt aus der Aktorenhierarchie. Die allgemeinste Klasse der Wortaktorenhierarchie ist das **Wort**. Dort werden die für alle Wörter gültigen Attribute (Instanzvariablen) und Verhaltensmuster (Methodenskripte) bereitgestellt, die für die Kommunikation mit anderen Wortaktoren und die Bestimmung konkreter Dependenzrelationen notwendig sind. Zu den Attributen zählen die *Bekanntschaften* (linker und rechter Nachbar im Text, Kopf und Modifikatoren in der Dependenzstruktur), die sprachliche Realisierung des betrachteten lexikalischen Elements (flektiertes *Textwort*), die *Position* des Worts im Text sowie das *Postfach*, in dem Nachrichten an den rechten Nachbarn gelagert werden, solange dieser noch nicht bekannt ist. Das Verhaltensmuster eines Aktors wird durch die mit den einzelnen Botschaftstypen assoziierten Methodenskripte determiniert (vgl. Abschnitt 2.3.3).

Eine Unterklasse von **Wort** ist beispielsweise die Klasse der **Nominale**, die durch die grammatischen Parameter Kasus, Genus, Numerus gekennzeichnet ist. **Nominale** werden weiter differenziert in **Artikel**, **Adjektive**, **Nomen** und **Pronomen**, die sich in ihren Valenzen[2] unterscheiden: **Artikel** und **Pronomen** haben keine Valenzen, **Adjektive** können durch Adverbien modifiziert werden und **Nomen** durch Artikel, Adjektive, Präpositionen, Nomen im Genitiv und Relativsätze. Diese verschiedenen Dependenzrelationen werden beschrieben durch den Namen der Relation und die jeweiligen grammatischen Bedingungen (in Form eines Attributterms). Die Differenzierung zwischen Nomen und Pronomen ist u.a. notwendig, da nur Nomen (und die Pronomen der 3.Person) als Anaphern fungieren können und eine Methode zum Auffin-

---

[1] Im folgenden wird wiederholt auf sog. *Dependenzrelationen* Bezug genommen, einem wesentlichen Konstrukt der hier zugrunde liegenden Sprachtheorie. Dependenzrelationen beschreiben einseitig gerichtete sprachliche Abhängigkeitsbeziehungen innerhalb und zwischen Phrasen (Teilsätzen). Diese Abhängigkeiten können am Beispiel der Nominalphrase *"ein ziemlich großer Hauptspeicher"* illustriert werden. Das zentrale Element (Kopf) dieser Nominalphrase ist das Nomen (*Hauptspeicher*), das sog. Modifikatoren, wie Artikel (*ein*) und Attributphrase (*ziemlich großer*), dominiert. Als Kopf ist dasjenige minimale Element einer Phrase ausgezeichnet, das durch andere Elemente der Äußerung syntaktisch oder semantisch modifiert wird, innerhalb dieser Phrase aber selbst keine anderen Elemente modifiziert (was für *"Hauptspeicher"* in der Nominalphrase - modifiziert durch *"ein"* und *"ziemlich großer"* - und *"großer"* in der Attributphrase - modifiziert durch *"ziemlich"* - jeweils gilt). Das Beispiel verdeutlicht ferner Reichweitenbeschränkungen von Dependenzrelationen, die klassischen Phrasengrenzen entsprechen.

[2] Die *Valenz* eines lexikalischen Elements kennzeichnet die (obligatorischen und fakultativen) Rollen, die ein Kopf für seine Modifikatoren anbietet. Valenzen können je nach Sicht syntaktisch markiert sein (Spezifikation zulässiger Wortartklassen oder syntaktischer Kasus als Modifikatoren) oder auch semantisch (dies wird in der Literatur häufig als Kasusrolle bezeichnet).

den des Referenzpunktes benötigen (vgl. Abschnitt 2.6). Sie wird in der Klasse **Nomen** und nicht in **Pronomen** definiert. Konkrete lexikalische Einheiten werden als Blätter der Wortaktorenhierarchie beschrieben, in denen spezifische grammatische Parameter (gegeben durch die Flexion) sowie mögliche idiosynkratische Eigenschaften eines Wortes (etwa die Permutierbarkeit des Objekts bei *"wegen"*) spezifiziert werden.

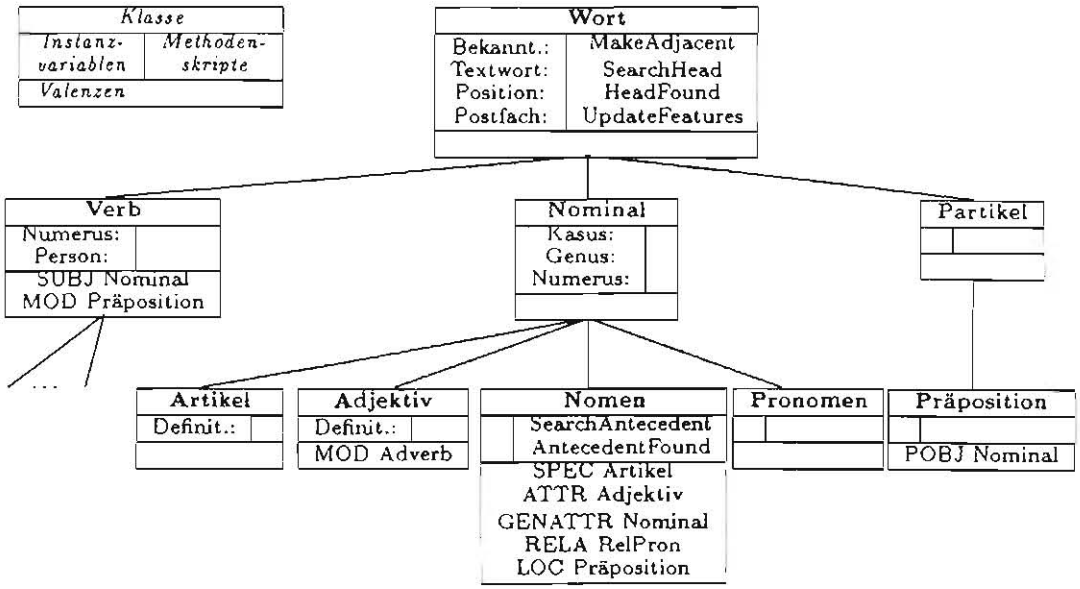


Abb. 1. Ein Ausschnitt der Wortaktorenhierarchie

2.3 Kommunikation zwischen Aktoren

Die Kommunikation zwischen Wortaktoren vollzieht sich ausschließlich durch Botschaftsaustausch, dem ein linguistisch valides Protokoll zugrunde liegt (vgl. a. Hahn [1990]). Die Nachrichten dienen dazu, zwischen einzelnen Wortaktoren Relationen zu etablieren, die sowohl die grammatische Struktur in und zwischen Sätzen eines Textes als auch die inhaltliche Interpretation auf der Grundlage der berechneten Dependenzstrukturen in der entsprechenden Textwissensbasis charakterisieren. Ein Wortaktor setzt dazu eine Suchmeldung nach einem passenden Kopf ab, die an einen (textuellen) Nachbarn (je nach Suchrichtung) adressiert ist und von diesem bearbeitet und/oder weitergeleitet wird. Ein Wortaktor, der die in der Suchmeldung enthaltenen Bedingungen erfüllt, reagiert mit einer Bestätigung an den Initiator der Suchmeldung, was zur Einrichtung der jeweiligen Relation führt. Die durch die Analyse eines Satzes entstehenden Dependenzstrukturen aktivieren wiederum entsprechende Änderungsoperationen im Textwissen, das die inkrementelle inhaltliche Interpretation des Textes wiederspiegelt.

2.3.1 Versendemodi

Das Aktorenmodell bietet für die Analyse einen *asynchronen* Kommunikationsmodus zwischen einzelnen Aktoren an. Dieser Modus erlaubt weiter die Unterscheidung zwischen direkten und vermittelten Nachrichten. *Direkte Nachrichten* ermöglichen eine Punkt-zu-Punkt-Kommunikation, wenn ein eindeutiger Adressat bekannt ist (z.B. bei der Bestätigung einer Anfrage). Sie lösen im adressierten Wortaktor entsprechende Aktionen aus, ohne weitergeleitet zu werden. *Vermittelte Nachrichten* werden für die Suche nach noch unbekannten Wortaktoren benutzt, die gewisse Bedingungen erfüllen (z.B. die Suche nach dem Kopf). Sie werden an den direkten Nachbarn adressiert, evtl. von ihm bearbeitet und in jedem Fall weitergeleitet (bis zu ihrer maximalen Reichweite), um alle möglichen Relationen zu berechnen. Ein bereits dominierter Wortaktor leitet jede vermittelte Nachricht ohne Bearbeitung an seinen Kopf weiter, da dieser als Repräsentant einer Phrase die Kontrolle über die Kommunikation mit anderen Wortaktoren hat. Der Kopf wird die Nachricht verarbeiten, evtl. an (einige) Modifikatoren und unter Beachtung von Reichweitenbeschränkungen an den nächsten Nachbarn weiterleiten. Zur Realisierung dieses Nachrichtenverkehrs sind Nachrichtenattribute eingeführt worden, die den *Versendemodus* bestimmen:

1. **flat** zeigt an, daß die Nachricht vom direkten Nachbarn kommt und an den Kopf delegiert werden soll. Hat der empfangende Aktor keinen Kopf, wird die Nachricht von ihm selbst verarbeitet, sonst wird das Attribut zu **delegating** verändert und die Nachricht an den Kopf geschickt.
2. **delegating** zeigt an, daß diese Nachricht von einem Modifikator kommt und an den Kopf gerichtet ist. Hat der empfangende Aktor keinen Kopf, wird die Nachricht von ihm selbst verarbeitet, sonst wird sie an den Kopf dieser Konfiguration geschickt.

Mit diesen beiden Versendemodi kann jede vermittelte Nachricht an den (momentanen[3]) absoluten Kopf eines Wortaktors verschickt werden (vgl. (1) und (2) in Abb.2). Dieser verarbeitet die Nachricht entsprechend seiner Spezifikation und kann sie weiterleiten. Falls die Nachricht an einen Modifikator weitergeleitet wird, erhält sie vom Sender das Attribut **down** und wird an den Modifikator gerichtet. Falls die Nachricht an den nächsten Nachbarn der gesamten Phrase weitergeleitet werden soll, erhält sie vom Sender das Attribut **left-deliver** bzw. **right-deliver** und wird an den am weitesten links bzw. rechts stehenden Modifikator verschickt ((3) in Abb.2).

3. **down** zeigt an, daß die Nachricht vom Kopf kommt und nur vom empfangenden Aktor verarbeitet werden darf. Evtl. kann dieser entscheiden, die Nachricht an seine Modifikatoren weiterzuvermitteln, dann ebenfalls mit dem Attribut **down**.
4. **left-deliver** (**right-deliver**) zeigt an, daß diese Nachricht vom Kopf kommt und an den linken (rechten) Nachbarn der Phrase adressiert ist. Hat der empfangende Aktor noch weiter links (rechts) liegende Modifikatoren, wird die Nachricht an den am weitesten links (rechts) liegenden weitergeleitet. Sonst wird das Attribut zu **flat** geändert und die Nachricht an den direkten linken (rechten) Nachbarn des empfangenden Aktors gesendet ((4) in Abb.2).

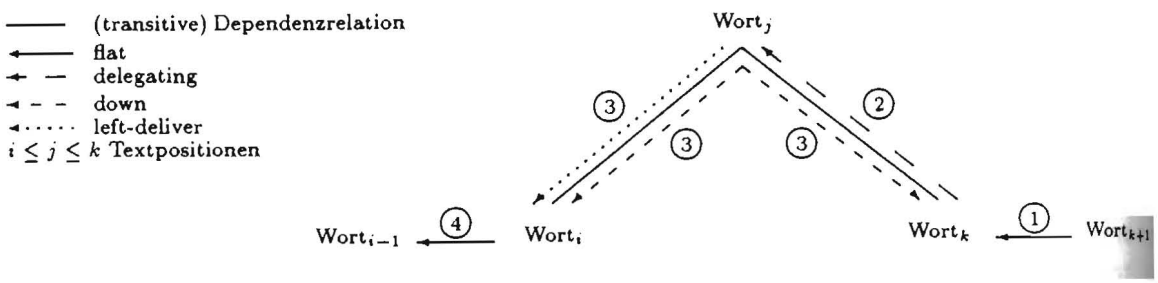


Abb. 2. Die Weiterleitung vermittelter Nachrichten.

Ist $Wort_j$ Kopf einer Phrase von $Wort_i$ bis $Wort_k$, und verschickt $Wort_{k+1}$ eine vermittelte Nachricht nach links, wird sie in 4 Schritten weitergeleitet, wie Abb.2 zeigt. Nachrichten mit derselben Ziffer werden parallel versandt und bearbeitet.

2.3.2 Struktur der Nachrichten

Nachrichten sind selbst Objekte, wenngleich bislang noch ohne hierarchische Struktur (wir beabsichtigen hier ebenfalls eine Hierarchisierung vorzunehmen). Sie enthalten neben ihrem Selektor (Nachrichtennamen) einige weitere Angaben:

1. *Initiator* bezeichnet den Wortaktor, der die Nachricht als erster versandt hat.
2. *Sender* bezeichnet den Wortaktor, der die Nachricht zuletzt weitergeleitet hat. Beim ersten Versenden einer Nachricht sind Initiator und Sender identisch.
3. *Empfänger* bezeichnet den Wortaktor, an den die Nachricht adressiert ist.
4. *Attribut* kennzeichnet den Versendemodus, der zur Verteilung und Weiterleitung der Nachricht benutzt wird (s.o.).

---

[3] Der Ausschluß von Zeitfehlereffekten im Rahmen dieser Lösung wird in Abschnitt 2.6 diskutiert.

5. *Reichweite* bezeichnet eine Grenze, bis zu der diese Nachricht laufen darf (Phrase, Satz, aktueller oder letzter Absatz, Textanfang). Der empfangende Aktor an einer solchen Grenze leitet die Nachricht nicht mehr weiter. Absehbar ist, daß hier noch komplexere, vor allem linguistisch motivierte Grenzbedingungen ("bis zum nächsten finiten Verb" o.ä.) zu formulieren sind.
6. *Parameter* sind die eigentlichen Daten der Nachricht, die durch den Selektor festgelegt werden

2.3.3 Nachrichtentypen

Die folgenden Nachrichtentypen sind allen Aktoren des Parsers bekannt, d.h. die Aktoren reagieren auf eine Nachricht gleichen Selektors mit dem jeweils skizzierten Verhalten (davon ist nur **SearchHead** eine vermittelte Nachricht):

1. **MakeAdjacent** macht den sendenden Aktor als rechten Nachbarn des empfangenden Aktors bekannt; bewirkt außerdem die Versendung aller dort im Postfach gespeicherten Nachrichten an den sendenden Aktor.
2. **SearchHead** sucht einen Kopf für den sendenden Aktor. Die Nachricht wird in die lexikalisch beschriebene Suchrichtung (**left**, **right** oder **both**) verschickt und enthält die grammatische Beschreibung des sendenden Aktors. Falls die grammatische Beschreibung eine der Dependenzrelationen des empfangenden Aktors erfüllt, wird dem Initiator die Nachricht **HeadFound** geschickt.
3. **HeadFound** als Antwort auf **SearchHead** macht den sendenden Aktor als Kopf des empfangenden Aktors bekannt. Empfängt ein Aktor mehrere solcher Nachrichten, wird die Behandlung struktureller Ambiguitäten angestoßen (s. Abschnitt 2.4). Ähnlich wie bei **UpdateFeatures** werden die grammatischen Merkmale des empfangenden Aktors instantiiert (ein neuer Satz von lokalen Daten erzeugt, damit die ursprüngliche lexikalische Spezifikation des Wortaktors nicht verlorengeht) oder, falls bestehende Instantiierungen umgeschrieben werden sollen, wird eine Kopie des ursprünglichen Merkmalssatzes angelegt (sog. Schatten, s.a. Abschnitt 2.4) und dieser instantiiert.
4. **UpdateFeatures** wird vom Kopf an Modifikatoren (bzw. in bestimmten Fällen auch in umgekehrter Richtung) verschickt, sofern deren grammatische Beschreibung verändert werden muß.

2.4 Ambiguitätsbehandlung

Zwei Arten von sprachlichen Mehrdeutigkeiten müssen vom Parser behandelt werden: *lexikalische Ambiguitäten*, die auftreten, wenn durch ein Textwort mehrere Wortarten oder Konzepte repräsentiert werden (z.B.: *"Schloß"* in der Verbform und in der wiederum ambigen Nominalform), sowie *strukturelle Ambiguitäten*, die vorliegen, wenn ein Wort mehrere alternative Dependenzrelationen eingehen kann (illustriert durch den dafür exemplarischen Satz *"Der Mann sah die Frau mit dem Fernrohr"*). Lexikalische Mehrdeutigkeiten sind direkt im Lexikon kodiert und werden bereits bei der Initialisierung berücksichtigt. Den Lesarten entsprechend werden simultan mehrere Wortaktoren initialisiert, die um die im Textzusammenhang "richtige" Interpretation konkurrieren und i.a. im Verlauf der Analyse rasch zur Selektion einer Lesart führen.

Strukturelle Mehrdeutigkeiten werden erst während der Analyse entdeckt. Sie liegen vor, wenn mehrere Bestätigungen auf eine **SearchHead**-Nachricht beim Initiator eingehen. Um in diesem Fall jeweils eine konsistente Belegung der Instanzvariablen zu garantieren, werden sog. *Schatten* angelegt, d.s. Kopien des originalen Wortaktors, die die verschiedenen Lesarten durch alternative Variablenbelegungen repräsentieren. Schattierte Wortaktoren weisen dasselbe Kommunikationsverhalten auf wie der originale Aktor, so daß die Mehrdeutigkeit sich nicht in einer Änderung des Protokolls niederschlägt.

2.5 Ein Parsing-Beispiel

Ein typischer Analyseablauf für ein Wort setzt sich aus folgenden Schritten zusammen: Nachdem ein neues Wort aus dem Text eingelesen worden ist, wird der entsprechende Lexikoneintrag gesucht und ein Wortaktor mit Angaben über den linken Nachbarn, die Form des flektierten Textworts und dessen Position

im Text sowie die morpho-syntaktischen Parameter (Numerus, Kasus, Genus u.ä.) instantiiert. Der Aktor meldet sich zunächst als neuer rechter Nachbar bei seinem linken Nachbar an (**MakeAdjacent**). Daraufhin wird ihm von dort die gespeicherte Post zugeschickt. Die zweite Nachricht, die ein neuer Wortaktor verschickt, ist die Suchmeldung für seinen Kopf (**SearchHead**), die je nach grammatisch bestimmter Suchrichtung nach rechts oder links adressiert wird. Die letzte Aktion in der Instantiierungssequenz ist die Meldung an den Scanner, daß ein neues Wort gelesen werden kann. Erst jetzt ist der Wortaktor bereit, die von seinen Nachbarn eingehenden Nachrichten zu bearbeiten. Sofern seine Suchmeldung nach einem Kopf akzeptiert wird, erhält er von dem betreffenden Aktor eine Bestätigung (**HeadFound**), worauf er seine Bekanntschaften entsprechend anpaßt. Damit ist die Bestimmung der (hierarchischen) dependentiellen Struktur des Satzes als Kommunikation zwischen autonomen Wortaktoren beschreibbar.

Dies soll durch die Analyse des Satzes *"Auf der Messe präsentiert XYZ den Laserdrucker der Konkurrenz"* verdeutlicht werden. Zusätzlich zu den in Abb.1 gezeigten Lexikonspezifikationen bietet das Verb *"präsentieren"* für Nomina die Relationen DIROBJ (Akkusativ) und INDIROBJ (Dativ); nur SUBJ und DIROBJ sind obligatorisch. Außer dem Wortaktor, der das Verb repräsentiert, senden alle Wortaktoren Suchmeldungen nach ihrem Kopf ab, das Verb selbst ist Kopf dieses Satzes. Nachdem *"Auf der Messe präsentiert XYZ den Laserdrucker"* geparst wurde, ist *"präsentieren"* der Kopf des Teilsatzes, seine Relationen MOD (*auf der Messe*), SUBJ (*XYZ*) und DIROBJ (*den Laserdrucker*) sind bereits belegt. Außerdem soll *"der"* bereits in SPEC-Relation zu *"Konkurrenz"* stehen (damit ist der Kasus der Phrase auf Dativ oder Genitiv eingeschränkt).

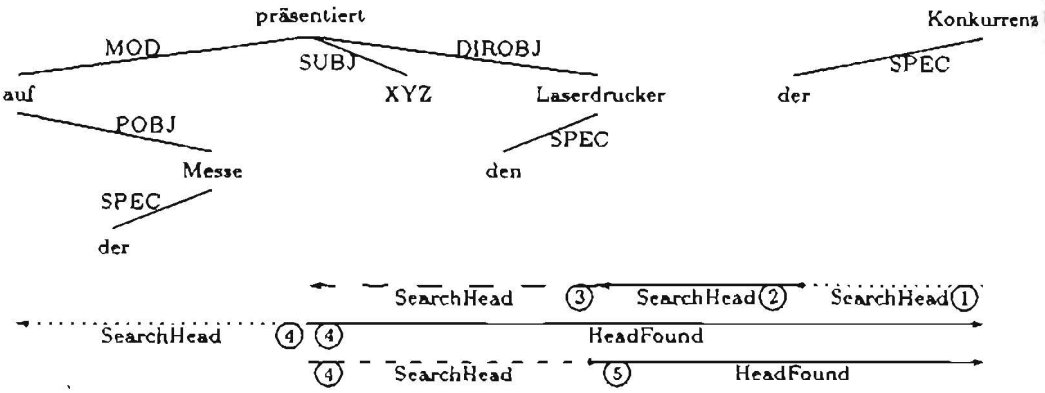


Abb. 3. Der Nachrichtenverkehr zur Suche des Kopfes von "Konkurrenz". Zur Legende siehe Abb.2.

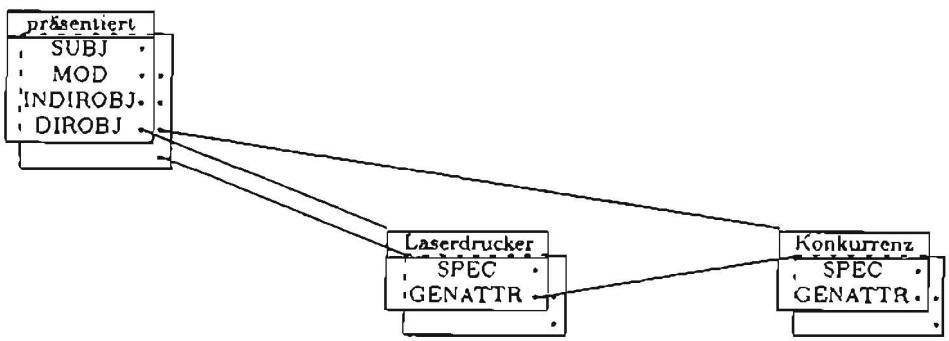


Abb. 4. Da *"Konkurrenz"* nicht mehrfach von *"präsentiert"* dominiert werden kann, wird der Wortaktor für *"Laserdrucker"* schattiert, und die Genitiv-Lesart von *"Konkurrenz"* besetzt die Relation GENATTR des neuen Schattens.

"Konkurrenz" sucht nun nach seinem Kopf, indem es die Nachricht **SearchHead** versendet. Rechts stößt die Nachricht an die Satzgrenze und bleibt ohne Effekt. Links gelangt die Suchmeldung über *"der"* zu *"Laserdrucker"*, der jedoch dominiert wird und daher die Nachricht mit dem Attribut **delegating** an seinen Kopf *"präsentiert"* weiterleitet. Dort wird sie verarbeitet, indem die Bewerbung akzeptiert wird und an *"Konkurrenz"* ein **HeadFound** geschickt wird; darauf reagiert *"Konkurrenz"* mit der Erzeugung eines Dativ-Schattens, in dem *"präsentiert"* als Kopf notiert ist. Außerdem wird die **SearchHead**-Nachricht aber

noch an Modifikatoren weitergegeben, jetzt mit dem Attribut **down**. Von diesen hat *"Laserdrucker"* ebenfalls Verwendung für *"Konkurrenz"*, diesmal als Genitiv in der Relation GENATTR, und schickt ebenfalls ein **HeadFound** an *"Konkurrenz"*. Dort wird nun ein Schatten des ursprünglichen Aktors mit einem Genitiv-Schatten und mit *"Laserdrucker"* als Kopf aufgebaut, der zunächst gleichberechtigt neben der Dativ-Lesart steht, aufgrund von Ausschlußbedingungen (eine Strukturanalyse darf jedes Wort nur einmal enthalten) aber nicht als globale Ambiguität erscheint.

2.6 Synchronisationsaspekte

Jeder Aktor führt zu einem Zeitpunkt maximal ein (unteilbares) Skript aus (oder wartet auf eine neue Nachricht), daher sind Konflikte bei Zugriffen auf *einen* Aktor unmöglich (vgl. Abschnitt 3). Im Zusammenspiel *mehrerer* Aktoren während der Analyse können sich zunehmend komplexere Konfigurationen ergeben, die auch die Wege der Nachrichten beeinflussen können. Damit daraus folgende Zeiteffekte das Analyseergebnis nicht beeinträchtigen, gilt die Verabredung, daß eine Nachricht unabhängig vom Status des Empfängers (innerhalb der Dependenzstruktur) und unabhängig davon, ob sie schon einmal empfangen worden ist (Aktoren haben kein "Nachrichtengedächtnis"), bearbeitet wird. Dabei können zwei Situationen unterschieden werden: der Aktor wird entweder schon von einem Kopf regiert oder er steht bislang in keiner solchen Dominanzrelation. Hieraus können keine auf Zeitfehler rückführbare inkonsistente Zustände der Analyse resultieren. Ein Wortaktor ohne Kopf prüft die Zulässigkeit der Anfrage des Nachbarn und etabliert im Falle der Verträglichkeit eine entsprechende Dependenzrelation. Ein Wortaktor mit Kopf erhält eine von diesem Kopf delegierte Nachricht und prüft diese in eigener Hoheit ebenfalls auf Akzeptabilität. Folgt aus der Kopf-Modifikator-Beziehung eine Einschränkung möglicher Dependenzbezüge, die Analysen ohne diese Beschränkungen ausschließen, werden Schatten angelegt, die nachträglich (etwa am Satzende) im Rahmen einer den gesamten Satz umfassenden Konsistenzprüfung evaluiert werden. Dieser Mechanismus dient dann dazu, nur lokal konsistente, satzglobal aber inkonsistente Lesarten zu eliminieren.

Als besondere Synchronisationsaufgabe erweist sich die linguistisch plausible Koordinierung der Aktivitäten einzelner Wortaktoren. Ein Beispiel für grammatikalisch zu handhabende linguistische *Serialisierungsbedingungen* ist der Vorrang der Anaphernauflösung vor der Modifikation des bezeichneten Konzepts (vgl. Hahn [1989, 1990]). Durch (Nominal-)Anaphern werden bereits erwähnte Konzepte (das Antezedens) im Text durch Referenten sprachlich erneut aufgenommen. Üblicherweise ist das im Text vorangehende Antezedens ein Unterbegriff bzw. eine Instanz des Referenten (Beispiel: *"Windows NT .... Diese Benutzeroberfläche..."*). Das hierbei auftretende Problem ist die korrekte Identifizierung des Konzepts, das durch den anaphorisch wirksamen Referenten bezeichnet wird (z.B. für die nachträgliche Modifikation durch im Text aufgeführte Attribute). Die Anaphernauflösung (*Benutzeroberfläche = Windows NT*) muß also vor möglichen Änderungsoperationen des vom Referenten denotierten Konzepts (hier *Benutzeroberfläche*) im Textwissen geschehen. Dies wird durch die Priorisierung der Anaphernbehandlung erreicht: Sie wird angestoßen, sobald ein definiter Artikel gefunden wird, und läßt erst nach ihrem Abschluß mögliche Konzeptmodifikationen zu.

3. Implementation

Der hier vorgestellte objektorientierte Dependenz-Parser wird im Rahmen eines *verteilten* Smalltalk80-Systems [Schacht 1993] implementiert. Zu den von uns vorgenommenen Erweiterungen des ursprünglichen Smalltalk80 [Goldberg & Robson 1989] zählen neben der physikalischen Verteilung in einem SUN-SPARCstation-Cluster die Realisierung eines *asynchronen* Kommunikationsmodus zwischen Objekten. Beide Erweiterungen sind in Smalltalk selbst implementiert.

Ein Smalltalk-Image als Single-User-System ist ein einziger Betriebssystemprozeß, verfügt aber intern über Multitasking durch eine eigene Verwaltung von Prozessen (im folgenden: *Smalltalkprozesse*). Um im prinzipiell auf synchronem Botschaftenaustausch basierenden Smalltalk-System asynchrone Nachrichten zu verschicken, wird für jede Nachricht ein Smalltalkprozess erzeugt, der die Verschickung und Bearbei-

lung der Nachricht durchführt. Pro Aktor wird ein Semaphor verwendet, um die Ununterbrechbarkeit d Skripts (der Methode) zu garantieren.

Die Verteilungserweiterung beruht auf einem Public-Domain-Paket [Xu 1993]. Aktive Smalltalk-Im ges können mit Hilfe je eines als Server dienenden Smalltalkprozesses miteinander kommunizieren. D Server verwalten eine Namens- und Adresstabelle, in der die wichtigen Systemkonstanten aller beteiligt Images mittels globaler Identifikationsnummern notiert sind. Objekte, die sich auf einem anderen H befinden, werden lokal durch ein Stellvertreter-Objekt, ein *Proxy*, repräsentiert. Die Klasse Proxy ist nic als Unterklasse von Object definiert und erbt daher auch keinerlei Methoden von dort. Proxies verstehe selbst nur einige Initialisierungs- und Testmethoden und reagieren ansonsten auf jede Nachricht, di eigentlich für ihr Original bestimmt ist, indem die Nachricht über den Server an den wirklichen Empfäng weitergeleitet, die Antwort darauf entgegengenommen und an den ursprünglichen Sender zurückgegeb wird.

Ein neues Objekt kann gezielt in einem bestimmten Image erzeugt werden, aber auch eine automa sche Lastverteilung zwischen den Images ist möglich. Dazu muß bei der Instantiierung eine Entscheidu über die Lokalisierung des neuen Objektes getroffen werden. Sinnvoller, als alle Objekte gleichmäßig im Netz zu verteilen, ist es jedoch, Gruppen von Objekten, die häufig miteinander kommunizieren müsse lokal zu halten, um die Kommunikationskosten zu minimieren. Dies trifft z.B. für alle Wortaktoren inner halb eines Satzes oder Absatzes zu, so daß der Verteilungsalgorithmus nicht bei jeder Aktoreninstanti ierung, sondern nur auf einen Schlüsselreiz (etwa das Auftreten eines Satzendezeichens) hin ausgefü werden muß.

4. Alternative Ansätze zur verteilten Sprachanalyse

Die aktuelle computerlinguistische Methodendiskussion ist wegen der engen Bindung an unifikations grammatische Konzepte nahezu ausschließlich vom Paradigma der logischen Programmierung dominier Wenngleich eine Kopplung von Techniken der Logikprogrammierung mit denen der objektorientierte Programmierung am Beispiel des Aktorenmodells durchaus schon vorgeschlagen worden ist [Shapiro & Takeuchi 1983], vollzieht sich die Rezeption objektorientierter Modellierungskonzepte in der Computer linguistik eher zurückhaltend. Entsprechende Adaptionsversuche, die vor allem auf die Nutzung von Ver erbungsmechanismen abstellen (vgl. die Übersichten von Gazdar [1987] und Daelemans et al. [1992] zielen auf die Definition von Lexikonhierarchien [Flickinger et al. 1985, van der Linden 1992, Andry et al 1992, Russell et al. 1992] bzw. beziehen neben der Lexikon- auch die Regelebene linguistischer Spezifika tionen ein [Miyoshi & Furukawa 1985, Vijay-Shanker & Schabes 1992, Fraser & Hudson 1992, Zaj 1992] und führen damit konsequent auch zur Adaption ihrer grundlegenden Parsing-Mechanismen, etwa durch Erweiterung der Standard-Unifikation zur Default-Unifikation [Bouma 1992].

Der Nutzen dieser redundanzminimierenden Form von objektorientierter Grammatik- bzw. Lexikon spezifikation beschränkt sich auf den deklarativen Kern einer lexikalischen Grammatik. Ausgeblende bleiben Mechanismen zur Kontrollspezifikation für lexikalische Parser (etwa in Form von Botschaftsme chanismen) in den oben erwähnten Arbeiten. So liegen zur Adaption regelbasierter Grammatikformate z Steuerungsprinzipien des *message passing* nur vereinzelte Beiträge vor, etwa die objektorientierte Rek dierung klassischer kontextfreier Grammatiken durch Yonezawa & Ohsawa [1988] bzw. die linguistisch interessanteren Ansätze von Phillips [1984] im Kontext einer Erweiterten Phrasenstrukturgrammatik (APSG) als Teil eines Textverstehenssystems bzw. von Marino [1988], der die mit klassischen Regelsyste men und ihren starren Kontrollstrukturen schwer erfaßbaren *long-distance dependencies* mit durch *mes sage passing* direkt kommunizierenden Merkmalbündeln beschreibt. Den modellmäßig weitgehendsten Integrationsversuch einer entwickelten computerlinguistischen Theorie mit objektorientierten Program mierprinzipien beschreiben Uehara et al [1985] im Rahmen einer Aktorenimplementation einer Lexika lisch-Funktionalen Grammatik (LFG).

Neben diesen auf das Parsing natürlicher Sprachen ausgerichteten Arbeiten sind auch andere Anwen dungen dokumentiert. So beschreiben Finkler & Neumann [1989] ein objektorientiertes Sprachgenerie

rungssystem, das auf einer in PATR-II eingebetteten Dependenzgrammatik beruht, De Mori et al. [1982] skizzieren ein aktorbasiertes System zur Erkennung gesprochener Sprache und Costantini et al. [1987] stellen einen verteilten Experten-Parser für ein Textverstehenssystem vor, dessen Protokoll auf der Metapher verteilter Problemlöser von Lesser & Corkill [1981] und dem Kontraktnetzprotokoll [Smith 1980] aufbaut. Ähnlich wie die Blackboard-Organisation eines von Sabah [1990] beschriebenen Sprachverstehenssystems wird bei Costantini et al. ein sehr grobkörniger Verteilungs- und Parallelisierungsansatz vertreten. So sind die Dekompositionseinheiten identisch mit dem, was in unserem Kontext der Interkomponenten-Parallelität entspricht, also Parallelität etwa zwischen morphologischer, syntaktischer oder semantischer Komponente. Damit werden - anders als im von uns beschriebenen Ansatz - Parallelisierungspotentiale unterhalb der Komponentenebene nicht ausgeschöpft.

Nicht zuletzt liegen die Wurzeln der Wortaktoren in den Arbeiten zum sog. *conceptual parsing*, d.h. stark semantisch geprägten lexikalischen Parsern wie ELI [Riesbeck & Schank 1978] oder IPP [Lebowitz 1983] sowie der Verfeinerung dieser Konzepte im Rahmen von Wortexpertensystemen [Small & Rieger 1982]. Diesem Ansatz war - wenn auch in Gestalt der LISP-Koroutinen, und damit Pseudo-Parallelität - eine konzeptionelle Nähe zu parallelen Implementationen stets inhärent (ein explizites Parallelisierungskonzept für Wortexpertensysteme im Kontext des logischen Programmierparadigmas verfolgen Devos et al. [1988]; aktuelle Parallelisierungsversuche für semantische Parser beschreiben Riesbeck & Martin [1986] im Rahmen des DMAP-Ansatzes, den man am ehesten als fallbasiertes Parsing charakterisieren kann). Neben der Kritik, daß in den Arbeiten zum konzeptuellen Parsing stets Modell- und Implementationsebene durchmischt werden, gelten die Haupteinwände gegen konzeptuelle Parser einer unzureichenden Berücksichtigung von linguistischem Wissens bzw. Grundelementen linguistischer Theoriebildung [Hahn 1989], die in unserer Entwicklung angemessen integriert werden sollen.

5. Systemumgebung des Parsers und Ausblick auf offene Fragen

Die hier vorgestellten Arbeiten beschreiben skizzenhaft ein Projekt, dessen Ziel die Entwicklung eines Spezifikationskontextes für Lexikon-Grammatiken und einer Implementationsumgebung für Lexikon-Parser ist, die beide auf Prinzipien der objektorientierten Programmierung beruhen. Die Orientierung daran drückt sich sowohl durch strukturelle (in Gestalt multipler Vererbungsmechanismen, s. Abschnitt 2.2) als auch prozedurale Konzepte (wie die Steuerung durch Nachrichtenaustäusche auf der Basis eines linguistisch validen Protokolls, s. Abschnitt 2.3) aus. Methodische Bezüge aus dem Kontext lexikalisierter Unifikationsgrammatiken werden auf der Ebene von Merkmalssystemen integriert, ordnen sich aber dependenztheoretischen und kasusgrammatischen Konstrukten unter (eine ähnliches, wenngleich weniger semantisch artikuliertes Sprachmodell vertreten Fraser & Hudson [1992]). Mit dieser Umgewichtung linguistisch-theoretischer Prämissen paßt sich das Modell ganz besonders Anforderungen komplexer Textverstehenssysteme und Einflüssen aus dem prozeßorientierten Paradigma der Kognitiven Linguistik (Interaktion sprachlicher mit nicht-sprachlichen Wissensquellen, dynamisch modifizierbare Präferenzordnungen syntaktischer, semantischer und pragmatischer Constraints, inhärente Parallelität) an.

Neben der hier ausführlich beschriebenen Sprachanalysekomponente kommt hierbei der *Weltwissensbasis* eine entscheidende Rolle zu. In unserem Fall interagiert der Parser mit einer LOOM-Wissensbasis [Brill 1992], wobei für jede Textanalyse eine eigene Wissensbasis, die sog. *Textwissensbasis*, instantiiert wird, die schließlich für Retrievalzwecke oder weitere informationelle Transformationen (Wissenssynthese, Textzusammenfassung o.ä.) verwandt werden kann. Der Parser verfügt über einen Satz von Anfrage- und Änderungs-Primitiven zur direkten Interaktion mit der Welt- bzw. Textwissensbasis.

Textverstehen impliziert in den meisten Fällen das Lernen neuer Zusammenhänge - einfache Fälle der Extraktion von Fakten, aber auch komplexere Prozesse des Konzept- und Regellernens. Beim Konzeptlernen führt die Interaktion zwischen der *Lernkomponente* und dem Parser zur Bestimmung von Bezeichnern für neue Konzepte und zur Identifizierung von konzeptuellen Eigenschaften dieser neuen Konzepte wie sie in den zugrunde liegenden Texten beschrieben werden. Die hierfür nötigen Protokolle zwischen Lerner und Parser bzw. Lerner und Weltwissensbasis befinden sich erst in einem frühen Entwicklungsstadium. Es ist absehbar, daß die bislang entwickelten ausschließlich linguistisch motivierten Parsing-Protokolle in

Nicht unerwähnt bleiben soll auch die Tatsache, daß mittelfristig eine Verfeinerung der linguistischen Beschreibungsebene für Wortaktoren ansteht, und zwar eine, bei der die Analyse nicht auf der Ebene der Wörter, sondern auf der Ebene der Morpheme ansetzt. Diese linguistisch adäquatere Lösung verlangt einen eigenen Satz von Nachrichtenprimitiven, denn die jetzige Ebene der Wörter kompiliert bereits statisch linguistische Prozesse, die innerhalb der Wortebene bei der Flexion und Wortbildung (Komposition, Derivation usw.) wirksam werden.

Langfristig streben wir auf der Basis einer formalen Spezifikation des deklarativen Vererbungs- und prozeduralen Kommunikationsmodells (Nachrichten) eine allgemeine Spezifikationssprache für Lexikon-Grammatiken an (in Anlehnung an Arbeiten zur Lexikonspezifikation für Unifikationsgrammatiken im DATR-Kontext [Evans & Gazdar 1989]). Auf dieser Basis soll eine Entwicklungsumgebung (Tracer, Debugger, Editor, Compiler) entstehen, die Spezifikationsaufgaben und Entwicklungs- sowie Testarbeiten automatisch unterstützen soll.

Danksagung. Die hier beschriebenen Arbeiten werden von der Deutschen Forschungsgemeinschaft im Rahmen des Schwerpunktprogramms "Kognitive Linguistik" unter dem Aktenzeichen Ha 2097 1/1 gefördert.

6. Literatur

Altmann, G.; Steedman, M. [1988]. Interaction with context during human sentence processing. *Cognition*, 30, 1988, pp.191-238.

Andry, F.; Fraser, N. M.; McGlashan, S.; Thornton, S.; Youd, N. J. [1992]. Making DATR work for speech: lexicon compilation in SUNDIAL. *Computational Linguistics*, 18, 1992 (3), pp.245-267.

Berwick, R.C.; Abney, S.P.; Tenny [1991]. *Principle-based Parsing.* Dordrecht, Boston, London: Kluwer.

Brill, D.[1992]. *LOOM Reference Manual. Version 1.4.*

Bouma, G. [1992]. Feature structures and nonmonotonicity. *Computational Linguistics*, 18, 1992 (2), pp.183-203.

Constantini, C. D.; Fum, G.; Montanari, A.; Tasso, C. [1987]. Text understanding with multiple knowledge sources: an experiment in distributed parsing. *Proc. 3rd Conf. of the European Chapter of the ACL.* Copenhagen/DK, 1987, pp.75-79.

Daelemans, W.; De Smedt, K.; Gazdar, G. [1992]. Inheritance in natural language processing. *Computational Linguistics*, 18, 1992 (2), pp.205-218.

De Mori, R.; Giordana, A.; Laface, P.. [1982]. Parallel algorithms for interpreting speech patterns. In K. Preston, L. Uhr (Eds.), *Multicomputers and Image Processing: Algorithms and Programs.* Orlando/FL etc.: Academic P., 1982, pp.193-205.

Devos, M.; Adriaens, G.; Willems, Y.D. [1988]. The parallel expert parser (PEP): a thoroughly revised descendent of the word expert parser (WEP). *COLING Budapest. Proc. of the 12th Intl. Conf. on Computational Linguistics (COLING '88).* Budapest, 22-27 August, 1988. Budapest: John von Neumann Society for Computing Sciences, 1988, Vol.1, pp.142-147.

Evans, R.; Gazdar, G. [1989]. Inference in DATR. *Proc. of the 4th Conf. of the European Chapter of the Association for Computational Linguistics.* Manchester, England, 10-12 April 1989, pp.66-71.

Finkler, W.; Neumann, G. [1989]. POPEL-HOW: A distributed parallel model for incremental natural language production with feedback. *IJCAI' 89: Proc. of the 11th Intl. Joint Conf. on Artificial Intelligence. Vol.2.* Detroit, Michigan, USA, 20-25 Aug., 1989. San Mateo/CA: Morgan Kaufmann, 1989, pp.1518-1523.

Flickinger, D.; Pollard, C.; Wasow, Th. [1985]. Structure-sharing in lexical representations. *Proc. of the 23rd Annual Meeting of the Ass. for Computational Linguistics.* Chicago, Ill., USA, 8-12 July 1985, pp.262-267.

Forster, K.I. [1979]. Levels of processing and the structure of the language processor. In W.E. Cooper, E.C.T. Walker (Eds.), *Sentence processing: Psycholinguistic studies presented to Merrill Garrett.* Hillsdale/NJ: Erlbaum, 1979, pp.27-85.

Fraser, Norman M.; Hudson, Richard A. [1992]. Inheritance in word grammar. *Computational Linguistics*, 18, 1992 (2), pp.133-158.

Frazier, L. [1987]. Theories of sentence processing. In L. Garfield (Ed.), *Modularity in Knowledge Representation and Natural-Language Understanding.* Cambridge/MA: MIT Pr., 1987, pp.291-307.

Gazdar, G. [1987]. Linguistic applications of default inheritance mechanisms. In P. Whitelock et al. (Eds.), *Linguistic Theory and Computer Applications.* London: Academic Pr., 1987, pp.37-67.

Goldberg, A.; Robson, D. [1989]. *Smalltalk-80.* The Language. Reading/MA. etc.: Addison-Wesley, 1989.

Hahn, U. [1989]. Making understanders out of parsers: Semantically driven parsing as a key concept for realistic text understanding applications. *International Journal of Intelligent Systems*, 4, 1989 (3), pp.345-393.

Hahn, U. [1990]. *Lexikalisch verteiltes Text-Parsing. Eine objektorientierte Spezifikation eines Wortexpertensystems auf der Grundlage des Aktorenmodells*. Berlin etc.: Springer, 1990. (Informatik-Fachberichte, 243 - K.I.).

Hahn, U.; Adriaens, G. [1991]. *Parallel natural language processing: background and overview*. Univ. Freiburg, Ling. Informatik/Computerlinguistik, July 1991 (= CLIF-Report 2/91). [erscheint in: G. Adriaens & U. Hahn (Eds.), *Parallel Natural Language Processing*. Norwood/NJ: Ablex, 1993]

Hewitt, C.; Baker, H. [1978]. Actors and continuous functionals. In E.J. Neuhold (Ed.), *Formal Description of Programming Concepts. Proc. of the IFIP Working Conf. on Formal Description of Programming Concepts*. St.Andrews, N.B., Canada, Aug. 1-5, 1977. Amsterdam etc.: North-Holland, 1978, pp.367-390.

Lebowitz, M. [1983]. Memory-based parsing. *Artificial Intelligence*, 21, 1983, (4), pp.363-404.

Lesser, V. R.; Corkill, D. D. [1981]. Functionally accurate, cooperative distributed systems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11. 1981, 81-96.

Linden, E.-J. van der [1992]. Incremental processing and the hierarchical lexicon. *Computational Linguistics*, 18. 1992 (2), pp.219-238.

Marino, M. [1988]. A process-activation based parsing algorithm for the development of natural language grammars. *COLING Budapest. Proc. of the 12th Intl. Conf. on Computational Linguistics [COLING '88]*. Budapest, 22-27 August, 1988. Budapest: John von Neumann Society for Computing Sciences, 1988, Vol.1, pp.390-395.

Marslen-Wilson, W.D.; Tyler, L.K. [1980]. The temporal structure of spoken language understanding. *Cognition*, 8, 1980, pp.1-71.

Miyoshi, H.; Furukawa, K. [1985]. Object-oriented parser in the logic programming language ESP. In V. Dahl, P. Saint-Dizier (Eds.), Natural Language Understanding and Logic Programming. Proc. of the 1st Intl. Workshop. Rennes, France, 18-20 Sept., 1984. Amsterdam etc.: North-Holland, 1985, pp.107-119.

Phillips, B. [1984]. An object-oriented parser. In: B.G. Bara, G. Guida (Eds.), *Computational Models of Natural Language Processing*. Amsterdam etc.: North-Holland, 1984, pp.297-321.

Riesbeck, Chr. K.; Schank, R. C. [1978]. Comprehension by computer: expectation-based analysis of sentences in context. In: W.J.M. Levelt & G.B. Flores d'arcais (Eds.), *Studies in the perception of language*. Chichester etc.: J. Wiley, 1978, pp.247-293.

Riesbeck, Chr. K.; Martin, Ch. E. [1986]. Direct memory access parsing. In J.L.Kolodner / C.K. Riesbeck (Eds), *Experience, Memory and Reasoning*. Hillsdale, NJ: L.Erlbaum, pp.209-226.

Russell, G.; Ballim, A.; Carroll, J.; Warwick-Armstrong, S. [1992]. A practical approach to multiple default inheritance for unification-based lexicons. *Computational Linguistics*, 18, 1992 (3), pp.311-337.

Sabah, G. [1990]. CARAMEL: A computational model of natural language understanding using a parallel implementation. *ECAI '90 - Proc. of the 9th European Conf. on Artificial Intelligence*. Stockholm, Sweden, August 6-10, 1990. London: Pitman, 1990, pp.563-565.

Schacht, S. [1993]. *A Distributed Smalltalk System. V1.0*. Univ. Freiburg, Ling. Informatik/Computerlinguistik, March 1993 (CLIF- Memo 1/93).

Shapiro, E.; Takeuchi, A. [1983]. Object oriented programming in concurrent Prolog. *New Generation Computing*, 1, 1983 (1), pp.25-48.

Shoham, Y. [1993]. Agent-oriented programming. *Artificial Intelligence*, 60, 1993 (1), pp.51-92.

Small, St.; Rieger, C. [1982]. Parsing and Comprehending with Word Experts (a Theory and its Realization). In W. Lehnert & M.H. Ringle (Eds.), *Strategies for natural language processing*. Hillsdale, NJ: Erlbaum, pp.89-147.

Smith, R. G. [1980]. The contract net protocol: high-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29, 1980 (12), pp.1104-1113.

Thibadeau; Just; Carpenter [1982]. A model of the time course and content of reading. *Cognitive Science*, 6. 1982, pp.157-203.

Uehara, K.; Ochitani, R.; Mikami, O.; Toyoda, J. [1985]. An Integrated Parser for Text Understanding: Viewing Parsing as Passing Messages among Actors. In V. Dahl, P. Saint-Dizier (Eds.), *Natural Language Understanding and Logic Programming. Proc. of the 1st Intl. Workshop*. Rennes, France, 18-20 Sept., 1984. Amsterdam etc.: North-Holland, 1985, pp.79-95.

Vijay-Shanker, K.; Schabes, Y. [1992]. Structure-Sharing in Lexicalized Tree-Adjoining Grammars. *COLING '92: Proc. of the 15th Intl. Conf. on Computational Linguistics*. Nantes, 23-28 Aug. 1992, Vol. 1., pp.205-211.

Xu, W. [1993]. *Distributed, Shared and Persistent Objects. A Model for Distributed Object-Oriented Programming*. Ph.D. Dissertation, Dept. of Computer Science, Queen Mary & Westfield College, London University, 1993.

Yonezawa, A.; Ohsawa, I. [1988]. Object-oriented parallel parsing for context-free grammars. *COLING Budapest. Proc. of the 12th Intl. Conf. on Computational Linguistics [COLING '88]*. Budapest, 22-27 August, 1988. Budapest: John von Neumann Society for Computing Sciences, 1988, Vol.2, pp.773-778.

Zajac, R. [1992]. Inheritance and constraint-based grammar formalisms. *Computational Linguistics*, 18. 1992 (2), pp.159-182.

Intelligente Wertpapiere

Bernd Mack und Christof Weinhardt, Universität Gießen, Licher Straße 60, 6300 Gießen

Abstract

Multi-Agenten-Systeme (MAS) bieten sich für eine informationstechnologische Unterstützung von Angebotsprozessen einer Bank an. Sie erlauben, *physisch* verteilt vorliegendes Fachwissen einzelner Produktbereiche zu integrieren und dabei gleichzeitig die positiven Effekte einer gewünschten *logischen* Verteilung zu nutzen.

Vereinfachend läßt sich jedes Anlageprodukt einer Bank - sogar jeder für ein Anlageprodukt zuständige Geschäftsbereich - als Wertpapier auffassen. Solche als autonome Agenten modellierte "intelligente Wertpapiere" bilden zusammen mit den für die Durchsetzung globaler Strategien verantwortlichen Bank- und Kunden-Agenten das in diesem Beitrag formal dargestellte MAS.

Der vorgestellte Formalismus zeigt, wie unter Verwendung spieltheoretischer und ökonomischer Konzepte das Verhalten der Agenten aus dem Gesamtverhalten des Systems heraus erklärt werden kann. In einem *globalen Spiel* bilden sich bei grundsätzlich vorherrschender *Konkurrenz* und *individuell rationalem* Verhalten Koalitionen unter den Wertpapieren. Koalitionsinterne Konflikte werden in *lokalen Spielen* durch Verhandlungen gelöst, in denen *kollektive Rationalität* der Wertpapiere gewährleistet wird. Auf beiden Spielebenen kann dabei das Verhalten der Wertpapiere im Sinne der Bank- und Kundeninteressen beeinflußt werden.

1 Einleitung

Eine ganzheitliche informationstechnologische Unterstützung des Angebotsprozesses einer Bank muß das räumlich, funktional und organisatorisch verteilt vorliegende Fachwissen einzelner Produktbereiche entscheidungsunterstützend an der Kundenschnittstelle zur Verfügung stellen und der Bank die Möglichkeit bieten, eigene strategische Interessen zu verfolgen. Multi-Agenten-Systeme (MAS) eignen sich für diese Problemstellung, da sie einerseits eine Integration *physisch* verteilt vorliegenden Wissens erlauben, andererseits aber auch die positiven Effekte einer gewünschten *logischen* Verteilung nutzen.

Aus unserer Sicht rücken zwei Aspekte bei der Entwicklung eines MAS in der Finanzberatung in den Vordergrund: Die Fähigkeit zur Entwicklung innovativer Lösungen durch Kooperation verschiedener Produktbereiche und die Möglichkeit, die Verfolgung globaler Strategien sicherzustellen, die das Verhalten aller an der Problemlösung beteiligten Bereiche beeinflussen, ohne direkt auf die dezentrale Entscheidungsfindung einzuwirken. Globale Strategien orientieren sich sowohl an Kunden- als auch an Bankinteressen. Die explizite Einbeziehung von Bankinteressen stellt dabei eine Erweiterung bisheriger Ansätze dar (vgl. *Buhl/Will [1993], Meyer et al. [1993], Kirn et al. [1991]*).

Gegenstand unseres Beitrages ist die formale Beschreibung eines MAS, das den oben genannten Aspekten Rechnung trägt. Bevor wir jedoch den Formalismus im einzelnen darstellen (Abschnitt 3), skizzieren wir die Organisationsstruktur des MAS (Abschnitt 2). In Abschnitt 4 weisen wir auf Grenzen sowie Möglichkeiten des vorgestellten Ansatzes hin.

2 Organisationsstruktur

Wertpapiere werden, gemäß der von *Markovitz (1952)* begründeten Portfolio-Theorie, in der Praxis meist aufgrund ihrer erwarteten Rendite und ihres Risikos (Varianz, Standardabweichung) beurteilt. Auch die Kombination verschiedener Wertpapiere zu einem Portfolio ist charakterisiert durch diese beiden Größen. Ziel der Bildung eines Portfolios ist i.d.R. die mit der Diversifikation einhergehende Reduktion des Risikos, wodurch der Nutzen eines risikoaversen Anlegers (Kunde) über das mit einzelnen Wertpapieren erreichbare Niveau angehoben werden kann.

Vereinfachend läßt sich jedes Anlageprodukt sowohl aus Sicht der Bank als auch aus Sicht des Kunden als Wertpapier auffassen. Bezieht man die mit dem Produktangebot verbundenen Kosten, Gebühren, Prämien sowie weitere spezifische Merkmale wie Fristigkeiten, Losgrößen oder Kundenboni mit ein, so kann sogar jeder für ein bestimmtes Anlageprodukt zuständige Geschäftsbereich einer Bank als Wertpapier mit eigenen Intentionen, wie z.B. Gewinnmaximierung, modelliert werden.

Eine *marktähnliche* Organisation der Wertpapiere bzw. Geschäftsbereiche unterstützt einerseits die Entwicklung innovativer Lösungen, da keine exogen vorgegebenen Strukturen die Kooperation bzw. Koalitionsbildung einschränken, andererseits die Entwicklung effizienter Lösungen, da nur solche Lösungen sich bei der vorherrschenden Konkurrenz durchsetzen können. Durch die Möglichkeit der Kooperation sind die Wertpapiere in der Lage, Probleme zu lösen, die nach Art und Umfang über ihre individuellen Fähigkeiten hinausgehen.

Hierachische Strukturen erlauben die Durchsetzung globaler Strategien aus Bank- bzw. Kundensicht. Die Verfolgung von Bank- und Kundeninteressen führt dabei häufig zu Interessenkonflikten: Neben der aus Kundensicht vorteilhaften Lösung ("Kundenportfolio") muß die Bank auch für die "optimale" Konstellation ihrer abgesetzten Produkte ("Bankportfolio") sorgen.

In diesem Sinne weist das in Bild 1 dargestellte MAS eine *hybride* Organisationsform auf (vgl. *Mack/Weinhardt [1993]*). Im Zentrum stehen die Wertpapiere, sprich Agenten, die jeweils ein Wertpapier bzw. ein Anlageprodukt vertreten. Es handelt sich um "intelligente Wertpapiere", die bei grundsätzlich vorherrschender Konkurrenz in einer marktähnlichen Organisationsstruktur in der Lage sind, sich zu Koalitionen zusammenzuschließen und gemeinsam innovative Lösungsangebote zu entwickeln. Die Wertpapiere agieren dabei im Spannungsfeld zwischen Kunden- und Bankinteressen. Dieses Spannungsfeld ist gekennzeichnet durch sich entgegenstehende hierarchische Strukturen, an deren jeweiligen Spitzen Kunden- und Bank-Agent stehen.

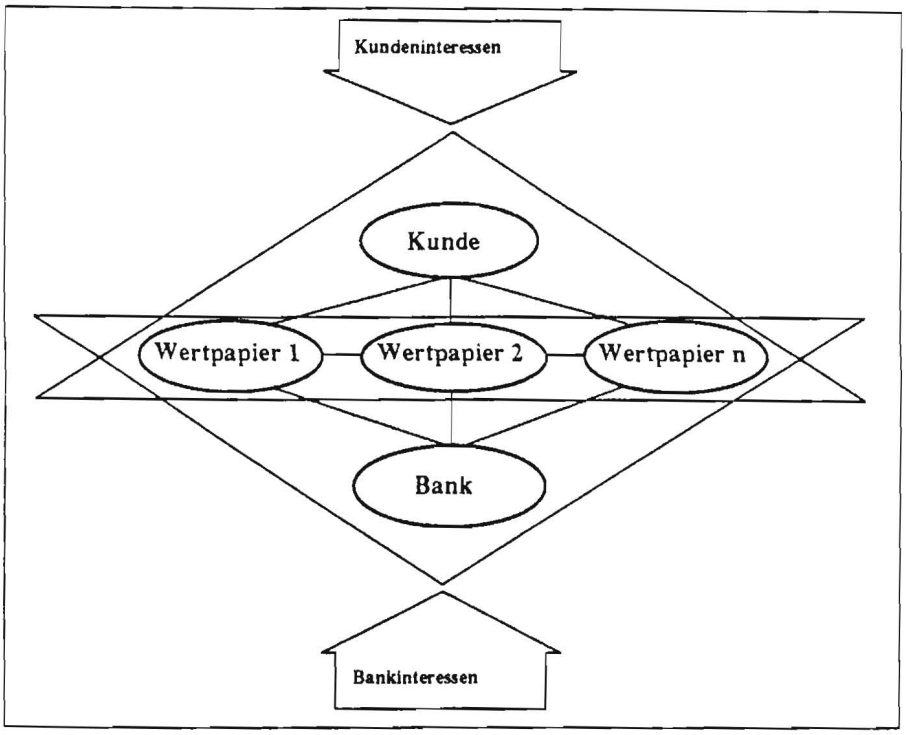


Bild 1: Organisationsstruktur

3 Formale Darstellung

In diesem Abschnitt stellen wir formal das in Bild 1 skizzierte MAS dar. Wir folgen einem "top-down"-Ansatz, wie er auch in *Burkhard [1992]* vorgeschlagen wird und erklären das Verhalten der Agenten aus dem Gesamtverhalten des Systems heraus. Dabei greifen wir auf spieltheoretische Konzepte zurück.[1]

Im betrachteten Kontext sind sogenannte *Koalitionsspiele* von Interesse - also Spiele bei denen zwei oder mehr Akteure sich zu Koalitonen zusammenschließen und als solche in das Spielgeschehen eingreifen. In einem *globalen Spiel* zeigen wir, wie sich Koalitionen aus Wertpapieren bilden und mit anderen Koalitionen bzw. Wertpapieren konkurrieren. Wir beschränken uns auf Koalitionen aus je zwei Wertpapieren.

Koalitionen bilden sich, wenn für die Mitglieder ein zusätzlicher Nutzen entsteht. Dieser zusätzliche Nutzen muß unter den Mitgliedern aufgeteilt werden. *Hart/Kurz [1983]*, S. 1047 charakterisieren das mit dieser Aufteilung verbundene Problem wie folgt:

> *"Further bargaining occurs among the members of each coalition on how to divide what they obtained together. Thus, the existence of coalitions implies that the*

---

[1] Die Spieltheorie hat bereits verschiedentlich Eingang in die VKI gefunden (vgl. z.B. *Rosenschein/Genesereth [1988]* oder *Kraus/Wilkenfeld [1990]*). Die folgende Charakterisierung der Spieltheorie bei *Kreps [1990]*, S. 355, verdeutlicht die Ähnlichkeit der in beiden Disziplinen behandelten Problemstellungen: "[...] the emphasis has been on explaining how cooperation (and many other forms of aggregate behaviour) can emerge from self-interested individual behaviour within a given set of 'rules'".

*interactions among the players will be conducted on two levels: first, **among** the coalitions, and second, **within** each coalition."*

Wir ergänzen daher das globale Spiel um *lokale Spiele*, die den bilateralen Verhandlungsprozeß zwischen Koalitionsmitgliedern beschreiben.

Gegenstand unserer formalen Darstellung ist ein dezentraler *Mechanismus* (vgl. z.B. *Holler [1992]*, S.141ff.). Es geht dabei um die Festlegung von Spiel- und Verhaltensregeln, die unter Berücksichtigung der Eigeninteressen (Intentionen) der Agenten die endogene Durchsetzung kooperativen Problemlösungsverhaltens ermöglichen und damit die Entwicklung *wünschenswerter* Ergebnisse gewährleisten.

Ein *wünschenswertes* Ergebnis läßt sich anhand der folgenden Kriterien beurteilen:[2]

- *Problemlösung*: Das gegenüber dem MAS formulierte Problem sollte gelöst werden.
- *Agenten*: Die Agenten sollten sich *individuell rational* verhalten.
- *Koalitionen*: Die zu einer Koalition gehörenden Agenten sollten sich *kollektiv rational* verhalten und eine *pareto-effiziente*[3] Nutzenallokation anstreben.

3.1 Das Multi-Agenten-System

Ein **M**ulti-**A**genten-**S**ystem **I**n der **F**inanzberatung (**MASIF**) - wie es in Bild 1 dargestellt wird - läßt sich definieren als:

$$MASIF := \left(A = \{W, B, K\}, S = \{S_W, S_B, S_K\}, V = \{V_W, V_B, V_K\}\right),$$

wobei A die Menge der *Agenten* (Akteure), S die Menge der möglichen *Spielzüge* (Aktionen) der Agenten und V die Menge der möglichen *Verhaltens*-Ausprägungen der Agenten ist.

3.2 Die Agenten

Die Menge der Agenten besteht aus dem Kunden K, der Bank B und den Wertpapieren W. Alle Agenten besitzen eine individuelle *Bewertungsfunktion* (Gewinn- oder Nutzenfunktion). Die Intentionen der Agenten orientieren sich an diesen Bewertungsfunktionen. Sie handeln individuell rational, indem sie versuchen, diese Bewertungsfunktion zu maximieren.

Der **Kunde *K*** vertritt die Interessen des Bankkunden, d.h. seine Bewertungsfunktion reflektiert die Entscheidungssituation des Bankkunden, die durch die erwartete Rendite r und das Risiko σ einer Anlagealternative gekennzeichnet ist. Die Bewertungsfunktion U ordnet

---

[2] Ähnliche Kriterien wenden auch *Rosenschein/Zlotkin [1990]*, S. 4f., oder *Rosenschein/Genesereth [1988]*, S. 233f., für Verhandlungsspiele an.

[3] *Pareto-Effizienz*: Keiner der Agenten kann sich verbessern, ohne dadurch einem anderen zu schaden.

jeder (r,σ)-Kombinationen unter Berücksichtigung des "Risiko-Rendite-Präferenz-Parameters" Θ des Bankkunden eine reelle Zahl zu. Das Kundenproblem ist außerdem durch das Anlagevolumen V gekennzeichnet.

Damit läßt sich der Kunde K definieren als Vektor

$$K := (V,U) \quad \text{mit} \quad U: R^3 \mapsto R \quad \text{und} \quad U:(\Theta,r,\sigma) \mapsto U(\Theta,r,\sigma) = \Theta \cdot r - \sigma^2.$$

Die **Bank** B als übergeordnete Institution soll die Möglichkeit haben, die Wertpapiere in ihrem Verhalten zu beeinflussen. Dies geschieht über Anreizmechanismen: Die Bank zahlt an ein Wertpapier W_i, $i=1,2,...,n$, Prämien für den realisierten Gewinn p_G^i, das erreichte Volumen p_V^i und den Geschäftsabschluß selbst p_A^i. Die Bank ist in der Lage, diese Prämien dynamisch zu verändern, um so durch eine indirekte "Steuerung" ihre Ziele zu verfolgen.[4]

In *Mack/Weinhardt [1993]* wird der bankbetriebliche Aspekt ausführlich behandelt und gezeigt, daß sich die Bank dabei im wesentlichen auf ihre *Bilanzkennziffern* $bk^1,...,bk^m$ stützt, die neben der Gewinnsituation auch Aufschluß über die Einhaltung von Bilanzstrukturnormen und anderen strategischen Zielvorgaben (z.B. seitens des Vorstandes) geben. Alle Abweichungen von den Vorgaben lassen sich in Erfolgsgrößen umrechnen, die wiederum die Bewertungsfunktion, nämlich den Gesamterfolg der Bank G^B, beeinflussen. Eine detaillierte Darstellung der funktionalen Zusammenhänge zwischen Bilanzkennziffern, Gesamterfolg und Prämiensystem würde den Rahmen dieser Arbeit sprengen.

Die Bank läßt sich definieren als Vektor

$$\begin{aligned} B := & \left(G^B, bk^1,...,bk^m, \left(p_G^1, p_V^1, p_A^1\right),...,\left(p_G^n, p_V^n, p_A^n\right)\right) \\ \text{mit} \quad & p_G^i: R^m \mapsto R \text{ und } p_G^i:\left(bk^1,...,bk^m\right) \mapsto p_G^i\left(bk^1,...,bk^m\right), \\ & p_V^i: R^m \mapsto R \text{ und } p_V^i:\left(bk^1,...,bk^m\right) \mapsto p_V^i\left(bk^1,...,bk^m\right), \\ & p_A^i: R^m \mapsto R \text{ und } p_A^i:\left(bk^1,...,bk^m\right) \mapsto p_A^i\left(bk^1,...,bk^m\right) \\ \text{für} \quad & i = 1, \ldots, n. \end{aligned}$$

Ein **Wertpapier** W_i ist charakterisiert durch Marktrendite r_M^i, Risiko σ^i und Gewinnfunktion g^i. Die Gewinnfunktion ist die Bewertungsfunktion des Wertpapieres bezüglich eines Spielzuges. Sie orientiert sich am Beitrag zum Gesamterfolg der Bank, den das Wertpapier leistet. Dieser Beitrag ergibt sich aus der Differenz zwischen der Marktrendite r_M^i und der tatsächlich dem Kunden angebotenen Rendite r^i multipliziert mit dem Anteil q^i am Volumen

---

[4] Die Gestaltung der Prämien erfolgt bei asymmetrischer Information zwischen der Bank und den Wertpapieren. D.h. die Bank kann das Verhalten der Wertpapiere weder vorherbestimmen noch beobachten bzw. kontrollieren. Sie muß daher durch ein anreizkompatibles Prämiensystem sicherstellen, daß sich die Wertpapiere zielkonform verhalten. Damit delegiert die Bank als Prinzipal die Aufgabe der Angebotserstellung an ihre Wertpapiere. Probleme solcher Delegationsbeziehungen bei asymmetrischer Information sind Gegenstand der *Principal-Agent-Theorie* in der Mikroökonomie (vgl. u.a. *Laux [1990]*).

V des Kundengeschäftes. Die tatsächliche Höhe des Wertpapiergewinnes hängt von den Prämiensätzen ab, die die Bank bei Beteiligung an einem Geschäftsabschluß einräumt.

Damit läßt sich die Menge der Wertpapiere definieren als:

$$W = \left\{ W_i \middle| i = 1,\ldots,n \wedge W_i = \left(r_M^i, \sigma^i, g^i\right) \right\} \text{ mit } g^i : R_+^3 \times R^3 \times R \mapsto R \text{ und}$$

$$g^i : \left(r_M^i, r^i, q^i, p_G^i, p_V^i, p_A^i, V\right) \mapsto$$

$$g^i\left(r_M^i, r^i, q^i, p_G^i, p_V^i, p_A^i, V\right) = \begin{cases} \left[\left(r_M^i - r^i\right) \cdot p_G^i + p_V^i\right] \cdot q^i \cdot V + p_A^i & \text{für } 0 < q^i \leq 1 \\ 0 & \text{für } q^i = 0 \end{cases}$$

3.3 Die Spielzüge

Ausgehend von den obigen Definitionen können nun die möglichen Aktionen der Agenten durch Spielzüge festgelegt werden:

Die Spielzüge k des Kunden K werden beschrieben durch das Nutzenniveau $\overline{U}$, das Volumen V und den Parameter Θ:

$$S_K := \left\{ k \middle| k = \left(\overline{U}, V, \Theta\right) \right\}$$

Die Spielzüge b der Bank B umfassen jeweils das vollständige Prämiensystem, das jedem Wertpapier i ein Tripel $p^i := \left(p_G^i, p_V^i, p_A^i\right)$ zuordnet:

$$S_B := \left\{ b \middle| b = \left(p^1, p^2, \ldots, p^n\right) = \left(\left(p_G^1, p_V^1, p_A^1\right), \left(p_G^2, p_V^2, p_A^2\right), \ldots, \left(p_G^n, p_V^n, p_A^n\right)\right) \right\}$$

Die Spielzüge w^{ii} eines Wertpapieres W_i (bzw. w^{ij} einer Koalition aus den Wertpapieren W_i und W_j) werden gebildet aus der angebotenen Rendite r^{ii} (r^{ij}), dem Risiko $\sigma^{ii}\left(\sigma^{ij}\right)$ und dem Anteil $q^{ii}\left(q^{ij}\right)$ am Volumen:

$$S_W := \left\{ w^{ij} \middle| i, j = 1,\ldots,n \wedge w^{ij} = \left(r^{ij}, \sigma^{ij}, q^{ij}\right) \wedge 0 \leq q^{ij} \leq 1 \right\}$$

Der Spielzug w^{ij} mit $i \neq j$, d.h. das Angebot einer Koalition, berechnet sich aufgrund der portfoliotheoretischen Zusammenhänge aus den Spielzügen w^{ii} und w^{jj} der beteiligten Wertpapiere W_i und W_j wie folgt:

$$w^{ij} = \left(r^{ij}, \sigma^{ij}, q^{ij}\right) = \left(q^{ii} \cdot r^{ii} + q^{jj} \cdot r^{jj}, \left(q^{ii} \cdot \sigma^{ii}\right)^2 + \left(q^{jj} \cdot \sigma^{jj}\right)^2, q^{ii} + q^{jj}\right)$$

Mit jedem Spielzug w^{ij} mit $q^{ij} = 1$ liegt ein Angebot vor, unabhängig davon ob $i = j$ oder $i \neq j$, also ob ein Wertpapier alleine oder eine Koalition dieses Angebot ins Spiel bringt. Mit jedem Spielzug w^{ii} mit $q^{ii} < 1$ liegt ein Teilangebot vor, zu dem noch ein Koalitionspartner gesucht wird (beachte: Für die Annahme von Zweierkoalitionen gibt es keine Portfolio-Angebote mit $q^{ij} < 1$).

3.4 Das Verhalten

In diesem Abschnitt gilt es, das Verhalten der Agenten in Form von Strategien zu beschreiben. Eine Strategie ist ein Auswahlmechanismus, der für jede Situation einen bestimmten Spielzug oder eine bestimmte Menge von Spielzügen für den Agenten festlegt (vgl. *Burkhard [1992]*, S. 19). Die Strategien der Agenten orientieren sich an ihren jeweiligen Bewertungsfunktionen.

Wir diskutieren im ersten Schritt das Verhalten aller Agenten im globalen Spiel und anschließend das Verhalten der zu einer Koalition gehörenden Wertpapiere im lokalen Spiel.

3.4.1 Das globale Auktionsspiel

Das globale Spiel entspricht einer Auktion, innerhalb derer das meistbietende Wertpapier oder die meistbietende Wertpapierkoalition den Zuschlag bekommt.

Die Rolle des Auktionators übernimmt der Kunde K, der in den Spielrunden $T = T_0, T_1, \ldots, T_z$ versucht, die Angebote w_T^{ij} der konkurrierenden Wertpapiere bzw. Wertpapierkoalitionen in die Höhe zu treiben: Er eröffnet die Auktion in T_0 mit dem Spielzug k_{T_0} und gibt eine Untergrenze $\overline{U}_{MIN} = \overline{U}_{T_0}$ für das zu erreichende Nutzenniveau an. Sofern in einer Spielrunde mindestens ein Angebot vorliegt, erhöht er in der folgenden Runde das geforderte Niveau. Das Verhalten des Kunden K läßt sich als Folge von Spielzügen k in den Spielrunden T definieren:

$$V_K := \left(k_T \middle| \begin{array}{l} k_T \in S_K \wedge \overline{U}_{T_0} = \overline{U}_{MIN} = U(\Theta, r_0, \sigma_0) \wedge \\ \overline{U}_T = \overline{U}_{T'} \cdot (1+d) \quad \text{mit } d \in R_+ \text{ und } T \text{ Folgerunde von } T' \end{array} \right)_{T=T_0,\ldots,T_z}$$

Alle Angebote einer Spielrunde T dominieren aus Kundensicht alle Angebote der vorhergehenden Spielrunden. Die letzte Spielrunde T_Z ist dadurch gekennzeichnet, daß in nachfolgenden Runden die Wertpapiere nicht mehr in der Lage sind, Angebote abzugeben, die über das in T_Z geforderte Nutzenniveau hinausgehen. Liegt in T_Z mehr als ein Angebot vor, muß willkürlich eines ausgewählt werden, da der Kunde K indifferent gegenüber verschiedenen Angeboten einer Spielrunde T ist.

Die Bank B greift nicht direkt in diese Auktion ein. Sie kann jedoch in jeder Spielrunde T durch Spielen von b_T eine Variation der individuellen Prämien p_T^i hervorrufen. So kann sie die Spielsituation der einzelnen Wertpapiere und damit auch die Koalitionsbildung beeinflussen. Welchen Spielzug sie wählt, wird durch die Bilanzkennziffern bestimmt. Wie oben erwähnt, gehen wir hier nicht näher auf den funktionalen Zusammenhang $b_T\left(bk_T^1, \ldots, bk_T^m\right)$ ein.

Vereinfacht läßt sich das Verhalten der Bank B als Folge von Spielzügen b in den Spielrunden T definieren:

$$V_B := \left(b_T \middle| b_T \in S_B \wedge b_T\left(bk_T^1,\ldots,bk_T^m\right)\right)_{T=T_0,\ldots,T_z}$$

Im globalen Spiel treten Koalitionen als Einheit auf. Sie konkurrieren mit anderen Wertpapieren bzw. Koalitionen, die ebenfalls in der Spielrunde T Angebote w_T^{ij} mit $q_T^{ij} = 1$ entwickeln. Individuelle Rationalität bestimmt dabei die Koalitionsbildung: Ein Wertpapier koaliert nur dann, wenn der Gewinn, den es als Koalitionsmitglied realisieren kann, größer ist als der Gewinn, den es alleine realisieren kann.

In jeder Spielrunde T prüft ein Wertpapier W_i , ob es das Kundenproblem alleine lösen kann. Falls ja, wird so der Referenzgewinn $\overline{g}_T^i = g_T^i\left(\ldots,r_T^{ii},\sigma_T^{ii},q_T^{ii}=1,\ldots\right)$ bestimmt. Wenn eine eigenständige Lösung nicht möglich ist, gilt für den Referenzgewinn: $\overline{g}_T^i = 0$. Anschließend versucht das Wertpapier W_i einen Koalitionspartner zu finden, indem es ein Teilangebot w_T^{ii} mit $q_T^{ii} < 1$ mit Hilfe eines Zufallsgenerators bildet und ausschreibt. Dabei muß $g_T^i\left(\ldots,w_T^{ii},\ldots\right) \geq \overline{g}_T^i$ gelten, d.h. durch Koalitionsbildung muß mindestens der Referenzgewinn $\overline{g}_T^i$ sichergestellt werden.

Ein Teilangebot $w_T^{ii} = \left(r_T^{ii},\sigma_T^{ii},q_T^{ii} < 1\right)$ wird von den potentiellen Koalitionspartnern geprüft. Die positive Reaktion eines Koalitionspartners W_j durch ein komplementäres Teilangebot $w_T^{jj} = \left(r_T^{jj},\sigma_T^{jj},q_T^{jj} = 1 - q_T^{ii}\right)$ bildet die Basis einer Koalition in der Spielrunde T, wenn für das resultierende vorläufige Koalitionsangebot w_T^{ij} gilt: $\overline{U}_T = U\left(\Theta,r_T^{ij},\sigma_T^{ij}\right)$. Analog werden auch bei dem Wertpapier W_i Teilangebote eingehen, auf die es reagieren kann.

Für jede mögliche Wertpapierkoalition läßt sich ein *Lösungsraum* L_T^{ij} bestimmen, der sowohl die möglichen Koalitionsangebote als auch die eigenständigen Lösungen für je zwei Wertpapiere W_i und W_j mit $i,j \in \{1,\ldots,n\}$ in der Spielrunde T enthält:

$$L_T^{ij} := \left\{ w_T^{ij} \middle| U\left(\Theta,r_T^{ij},\sigma_T^{ij}\right) = \overline{U}_T \wedge q_T^{ii} + q_T^{jj} = 1 \wedge g_T^i\left(\ldots,w_T^{ii},\ldots\right) \geq \overline{g}_T^i \wedge g_T^j\left(\ldots,w_T^{jj},\ldots\right) \geq \overline{g}_T^j \right\}$$

Jedes Koalitionsangebot $w_T^{ij} \in L_T^{ij}$ impliziert einen bestimmten Gewinnvektor $\gamma_T^{ij}\left(w_T^{ij}\right)$, der die Gewinnsituation der beiden Koalitionspartner W_i und W_j beschreibt. Für jeden Lösungsraum L_T^{ij} läßt sich der korrespondierende *Gewinnraum* Γ_T^{ij} bestimmen:

$$\Gamma_T^{ij} := \Gamma_T^{ij}\left(L_T^{ij}\right) = \left\{ \gamma_T^{ij} \middle| \gamma_T^{ij} = \gamma_T^{ij}\left(w_T^{ij}\right) = \left[g_T^i\left(\ldots,w_T^{ii},\ldots\right), g_T^j\left(\ldots,w_T^{jj},\ldots\right)\right] \wedge w_T^{ij} \in L_T^{ij} \right\}$$

Der Gewinnraum Γ_Z^{ij} ist *abgeschlossen* und *konvex*.[5] Konvexität und Abgeschlossenheit sichern die Existenz pareto-effizienter Gewinnvektoren. Wir bezeichnen die Menge aller pareto-effizienten Gewinnvektoren als *Kontraktkurve* $H(\Gamma_T^{ij}) \subseteq \Gamma_T^{ij}$.

Ausgehend von jedem vorläufigen Koalitionsangebot w_T^{ij} mit $i,j \in \{1,\ldots,n\}$ und $i \neq j$ beginnen die beteiligten Wertpapiere W_i und W_j lokale Verhandlungsspiele (vgl. nächster Abschnitt), deren Ergebnis das aus den Teilangeboten $\hat{w}_T^{ii}$ und $\hat{w}_T^{jj}$ gebildete endgültige (pareto-effiziente) Koalitionsangebot $\hat{w}_T^{ij}$ mit $\gamma_T^{ij}(\hat{w}_T^{ij}) \in H(\Gamma_T^{ij})$ ist. Dabei ist das endgültige Koalitionsangebot abhängig vom Verhalten der Wertpapiere im lokalen Spiel v_{W_i} und v_{W_j}, d.h. $\hat{w}_T^{ij} = \hat{w}_T^{ij}(w_T^{ij}, v_{W_i}, v_{W_j})$.

Das Gesamtverhalten eines Wertpapieres W_i läßt sich somit als Folge von Mengen von Spielzügen $\hat{w}_T^{ii}$ (mit unterschiedlichen Agenten W_j) definieren:

$$V_{W_i} := \left(\left\{ \hat{w}_T^{ii} \middle| \exists \hat{w}_T^{jj} : \hat{w}_T^{ij} = \hat{w}_T^{ij}(w_T^{ij}, v_{W_i}, v_{W_j}) \wedge \hat{w}_T^{ij}, w_T^{ij} \in L_T^{ij} \wedge \gamma_T^{ij}(\hat{w}_T^{ij}) \in H(\Gamma_T^{ij}) \right\} \right)_{T=T_0,\ldots,T_Z}$$

Die Elemente dieser Mengen sind so implizit über die Lösungs- und Gewinnräume der jeweiligen Koalitionen und über das Verhalten der beteiligten Wertpapiere im lokalen Verhandlungsspiel erklärt.

3.4.2 Das lokale Verhandlungsspiel

Im lokalen Verhandlungsspiel geht es um eine pareto-effiziente Aufteilung des Koalitionsgewinnes. Dabei bildet das vorläufige Koalitionsangebot w_T^{ij} eine Verhandlungsbasis, die den Koalitionspartnern W_i und W_j bereits einen Gewinn $g_T^i(\ldots, w_T^{ii}, \ldots) \geq \overline{g}_T^i$ und $g_T^{jj}(\ldots, w_T^{jj}, \ldots) \geq \overline{g}_T^j$ sichert. Ziel der Koalitionspartner im lokalen Spiel ist es, ihre Gewinnsituation weiter zu verbessern.

Bild 2 illustriert die lokale Verhandlungssituation anhand einer Darstellung des Gewinnraumes Γ_T^{ij}. Die Achsen repräsentieren die jeweilige Gewinnsituation g_T^i und g_T^j. Die Abgeschlossenheit von Γ_T^{ij} wird bestimmt durch die Referenzgewinne $\overline{g}_T^i$ und $\overline{g}_T^j$ sowie die Kontraktkurve $H(\Gamma_T^{ij})$ (konvexer Rand). Der Verhandlungsprozeß wird beschrieben durch den Pfad von einem Punkt $S = \gamma_T^{ij\,S}(w_T^{ij}) \in \Gamma_T^{ij}$, der den Gewinnvektor eines vorläufigen

---

[5] Auf einen vollständigen Nachweis dieser Eigenschaften wird hier verzichtet. Da aber die Konvexität von Γ_T^{ij} die Voraussetzung für die Existenz pareto-effizienter Gewinnvektoren ist, soll der Beweis kurz skizziert werden: Für zwei beliebige $\gamma_T^{ij'}, \gamma_T^{ij''} \in \Gamma_T^{ij}$ wird zunächst gezeigt, daß auch für jede Linearkombination $\gamma_T^{ij} = l \cdot \gamma_T^{ij'} + (1-l) \cdot \gamma_T^{ij''} \in \Gamma_T^{ij}$ mit $l \in [0,1]$ gilt. Anschließend wird gezeigt, daß auch die den Linearkombinationen zugeordneten Koalitionsangebote $w_T^{ij} \in L_T^{ij}$ sind.

Koalitionsangebots repräsentiert, zu einem Punkt $X = \gamma_T^{ij\,X}\left(\hat{w}_T^{ij}\right) \in H\left(\Gamma_T^{ij}\right)$ *auf der Kontraktkurve, mit dem das endgültige Koalitionsangebot* $\hat{w}_T^{ij}$ *im globalen Spiel feststeht.*

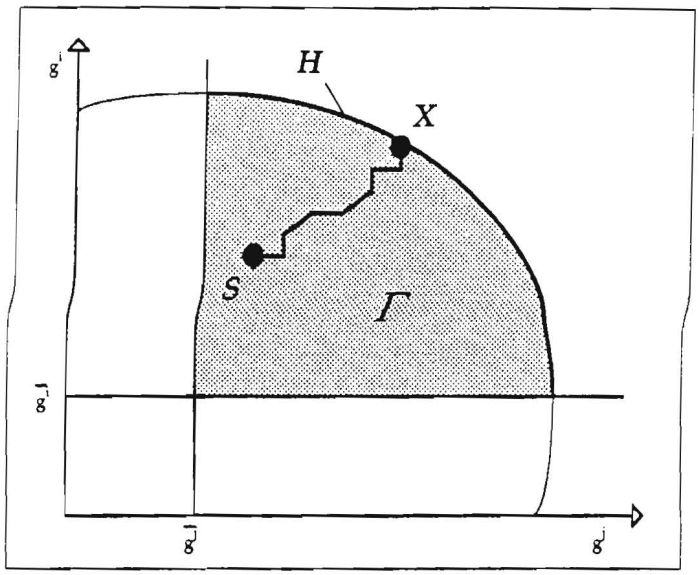


Bild 2: Verhandlungssituation

Wir bezeichnen die Spielrunden des lokalen Verhandlungsspiels mit $t = 0,1,...,h$. Das im globalen Spiel für eine Koalition gefundene vorläufige Angebot w_T^{ij} wird zur Verhandlungsbasis $\hat{w}_{t=0}^{ij}$ des lokalen Spiels mit den jeweiligen Teilangeboten $\hat{w}_{t=0}^{ii}$ und $\hat{w}_{t=0}^{jj}$. Es handelt sich um ein *sequentielles* Verhandlungsspiel mit wechselnder Initiative. In den geraden Spielrunden $t = 0,2,4,...$ erzeugt das Wertpapier W_i einen Verhandlungsvorschlag; in den ungeraden Spielrunden $t = 1,3,5,...$ tut dies das Wertpapier W_j.

Die Verhandlungsvorschläge sind jeweils *Mengen individuell rationaler Spielzüge*, die ein Wertpapier W_i in der Verhandlungsrunde t aufgrund seines Verhaltens v_{W_i} zu spielen bereit ist. Für den Verhandlungspartner W_j können diese Vorschläge Spielzüge enthalten, die auch zu seiner Menge individuell rationaler Strategien gehören. Diese Schnittmenge ist die *Menge kollektiv rationaler Spielzüge*, wie sie auch in Bild 3 für eine durch $Y = \gamma_T^{ij\,Y}\left(\hat{w}_t^{ij}\right) \in \Gamma_T^{ij}$ beschriebene Verhandlungssituation dargestellt ist.

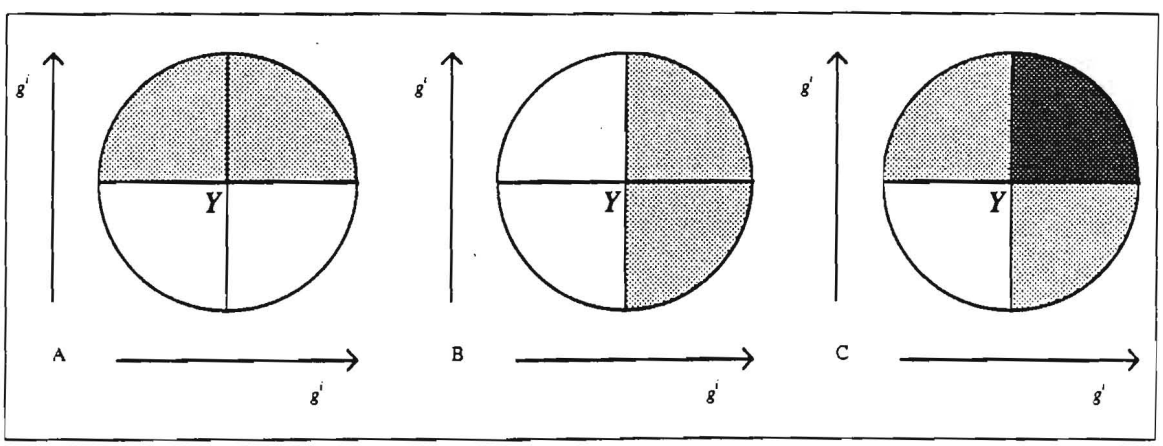


Bild 3: Mengen individuell rationaler (A, B) und kollektiv rationaler Spielzüge (C)

In den Spielrunden $t \in \{0,2,4,...\}$ wählt W_i den von ihm präferierten kollektiv rationalen Spielzug $\hat{w}_t^{ij*}$ und erzeugt von diesem ausgehend einen Verhandlungsvorschlag durch Variation der Größen $\hat{r}_t^{ij*}$ und $\hat{q}_t^{ij*}$:

$$v_{W_i} := \left\{ \left\{ \vec{w}_t^{ij} \;\middle|\; \begin{array}{l} \hat{w}_t^{ij} = \left[\left(1 + s_\rho^i \cdot \rho_t^i\right) \cdot \bar{r}_t^{ij*}, \bar{\sigma}_t^{ij*}, \left(1 + s_\theta^i \cdot \theta_t^i\right) \cdot \bar{q}_t^{ij*} \right] \\ \text{mit } \hat{w}_t^{ij*} = \max\limits_{r_t^{ij} \cdot q_t^{ij} = 1 - q_{t-1}^{ij}} \left(\left\{ w_{t-1}^{ij} \right\} \in v_{W_j} \right) \wedge \\ g_T^i\left(..., \hat{w}_t^{ij}, ...\right) \geq g_T^i\left(..., \hat{w}_t^{ij*}, ...\right) \wedge \\ s_\rho^i, s_\theta^i \in \{-1,0,1\} \wedge \rho_t^i, \theta_t^i \in [0,1] \end{array} \right\} \right\}_{t=0,2,4,...}$$

Analog verhält sich der Verhandlungspartner W_j in den Spielrunden $t = 1,3,5,....$

Das Verhandlungsspiel endet in der Spielrunde $t=h$, wenn keiner der Partner bereit ist, von dem in $t=h$-1 gefundenen Koalitionsangebot abzuweichen, d.h. wenn $\hat{w}_h^{ij*} = \hat{w}_{h-1}^{ij*}$. Das Angebot $\hat{w}_h^{ij*}$ ist das endgültige Koalitionsangebot $\hat{w}_T^{ij}$ im globalen Spiel. Jedes aufgrund dieses *monotonen* Verhandlungsprozesses entwickelte Koalitionsangebot $\hat{w}_T^{ij}$ impliziert eine pareto-effiziente Gewinnaufteilung in der jeweiligen Spielrunde T, denn es gilt: $\gamma_T^{ij}\left(\hat{w}_T^{ij}\right) \in H\left(\Gamma_T^{ij}\right)$.[6]

3.5 Das Spielergebnis

Das beschriebene Verhalten der Agenten im globalen Auktionsspiel und in den lokalen Verhandlungsspielen führt trotz individuell rationalem Verhalten und grundsätzlich vorherrschender Konkurrenz zwischen den einzelnen Wertpapieren zur endogenen Formation von Koalitionen. Die Koalitionsstruktur, d.h. welche Koalitionen sich letztlich durchsetzen, und auch das aus Kundensicht erreichbare Nutzenniveau werden dabei indirekt durch die Strategien der Bank bestimmt, die durch ihre "Prämienpolitik" auf das individuelle Verhalten der Wertpapiere einwirkt.

Das formal dargestellte **MASIF** liefert *wünschenswerte* Ergebnisse, denn jedes dem Bankkunden unterbreitete Angebot $\hat{w}_{T_z}^{ij}$ erfüllt alle drei der anfangs erläuterten Kriterien:

- *Problemlösung*: Das durch k_{T_0} repräsentierte Kundenproblem wird durch $\hat{w}_{T_z}^{ij}$ gelöst: $U\left(\Theta, r_{T_z}^{ij}, \sigma_{T_z}^{ij}\right) \geq \overline{U}_{T_z} \geq \overline{U}_{T_0} = \overline{U}_{MIN}$.

---

[6] Aufgrund der diskreten Verhandlungsschritte kann nicht gewährleistet werden, daß $\gamma_T^{ij}\left(\hat{w}_T^{ij}\right)$ gerade auf der Kontraktkurve $H\left(\Gamma_T^{ij}\right)$ zum Liegen kommt. Durch entsprechende Wahl der Schrittweiten s_ρ^i, s_θ^i und s_ρ^j, s_θ^j kann jedoch sichergestellt werden, daß $\gamma_T^{ij}\left(\hat{w}_T^{ij}\right)$ in einer hinreichend kleinen ε-Umgebung von $H\left(\Gamma_T^{ij}\right)$ liegt.

- *Agenten:* Die Agenten handeln individuell rational. Insbesondere gilt für die an $\hat{w}^{ij}_{T_z}$ beteiligten Wertpapiere: $g^i_{T_z}\left(\ldots, w^{ii}_{T_z}, \ldots\right) \geq \overline{g}^i_{T_z} \wedge g^j_{T_z}\left(\ldots, w^{jj}_{T_z}, \ldots\right) \geq \overline{g}^j_{T_z}$.
- *Koalitionen:* Die Koalitionsmitglieder handeln auch kollektiv rational, denn bei $\hat{w}^{ij}_{T_z}$ handelt es sich um ein Angebot, dessen Gewinnvektor auf der Kontraktkurve liegt: $\gamma^{ij}_{T_z}\left(\hat{w}^{ij}_{T_z}\right) \in H\left(\Gamma^{ij}_{T_z}\right)$.

4 Schlußbemerkung

Die formale Darstellung von **MASIF** zeigt, wie Erkenntnisse aus der Spiel- und Principal-Agent-Theorie der Mikroökonomie konkret Anwendung finden können. Die Methoden der VKI helfen dabei, Problemlösungen zu generieren, die in konventionellen wissensbasierten Systemen nicht erreichbar sind. Der Formalismus bildet die theoretische Grundlage einer prototypischen Implementation.

Die Restriktion auf Zweier-Koalitionen bedeutet zwar eine Einschränkung des Lösungsraums, bietet jedoch die Chance, **MASIF** in einem geschlossenen Modell überschaubar darzustellen. Bereits dieser einfache Ansatz weist auf eine Reihe offener Fragestellungen hin, die Gegenstand zukünftiger theoretischer wie auch experimenteller Untersuchungen sind, wie z.B.: Auf welche Weise können Koalitionen aus mehr als zwei Wertpapieren gebildet werden? Wie muß der Verhandlungsprozeß innerhalb solcher Koalitionen aussehen? Wie kann taktisches Verhalten (Drohstrategien, Blockieren, Täuschung, o.ä.) im lokalen und globalen Spiel aussehen? Wie sehen "optimale" Prämiensysteme aus? Wie kann die Bank diese als Reaktion auf getätigte, bilanzwirksame Geschäfte dynamisch anpassen?

Literatur

Buhl, H U ; Will, A.: "Unterstützung von Allfinanz-Angebotsprozessen mit verteilten wissensbasierten Systemen (ALLFIWIB)". Erscheint in: Information Management, 2/1993.

Burkhard, H -D : "Ein Formalismus für Multi-Agenten-Systeme". In: Künstliche Intelligenz (KI), Heft 1, 1992, S. 17-21.

Hart, S.; Kurz, M.: "Endogenous Formation of Coalitions". In: Econometrica, Band 51, Heft 4, 1983, S. 1047-64.

Holler, M. J.: Ökonomische Theorie der Verhandlungen, 3. Auflage, München, 1992.

Kraus, S.; Wilkenfeld, J.: "The Function of Time in Cooperative Negotiations: Preliminary Report". In: Proceedings of the 10th. International Workshop on Distributed Artificial Intelligence, Bandera, 1990.

Kreps, D. M.: A Course in Microeconomic Theory, Princeton, 1990.

Kirn, S; Scherer, A.; Schlageter, G.:"The FRESCO Agent Model: Cooperative Behavior in Federative Environments" In: Intelligent & Cooperative Systems (ICIS): Bringing AI & Information Technology together, IJCAI-91 Workshop, Darling Harbour, Sydney, Australia, 1991.

Laux, H.: Risiko, Anreiz und Kontrolle - Principal-Agent-Theorie: Einführung und Verbindung mit dem Delegationswert-Konzept, Berlin, 1990.

Mack, B ; Weinhardt, C.: "MASIF - ein Multi-Agenten-System in der Finanzberatung: Ein prototypischer Ansatz zur Integration von Marketing und Controlling-Strategien". Erscheint in: Information Management, 1993.

Markovitz, H.: "Portfolio Selection". In: The Journal of Finance, Bd. 7(1952): S. 77-91.

Meyer, U.; Mülheims, A.; Müller-Wünsch, M.; Schopf, C.; Woltering, A.: "Multi-Agent Architecture for Intelligent Financial Consulting (MAGNIFICO): Eine kooperierende, wissensbasierte Anwendung zur Unterstützung der Allfinanzberatung. Erscheint in: Kurbel, K. et al. (Hrsg.): Tagungsband Wirtschaftsinformatik 1993 Münster, 1993.

Rosenschein, J. S.; Genesereth, M. R.: "Deals Among Rational Agents". In: Bond, A. H.; Gasser, L. (Hrsg.): Readings in Distributed Artificial Intelligence. San Mateo, 1988, S. 227-234.

Rosenschein, J. S.; Zlotkin, G.: "Blocks, Lies, and the Postal Freight: The Nature of Deception in Negotiation". In: Proceedings of the 10th. International Workshop on Distributed Artificial Intelligence. Bandera, 1990.

Schierenbeck, H.: Ertragsorientiertes Bankmanagement, 3. Auflage. Wiesbaden, 1991.

Gründungsworkshop der GI-Fachgruppe 1.1.6 VKI, Saarbrücken, 29.-30. April 1993

ORGANISATIONAL INTELLIGENCE IN MULTIAGENTENSYSTEMEN: STATE OF THE ART UND FORSCHUNGSANSÄTZE

Stefan Kirn

Westfälische Wilhelms-Universität Münster
Institut für Wirtschaftsinformatik
D-4400 Münster

Andi Klöfer

Deutsche Bundesbank
D-6000 Frankfurt / Main

1 MOTIVATION UND UNTERSUCHUNGSANSATZ

Eine wesentliche Motivation zur Entwicklung von Multiagentensystemen besteht in der Überlegung (oder Hoffnung?), daß eine Gruppe von Agenten durch kooperative Zusammenarbeit Probleme lösen kann, die für einzelne Agenten des Verbundes im Hinblick auf das benötigte Wissen, verfügbare Ressourcen, Fehlertoleranzen oder zu beachtende Zeitrestriktionen unlösbar sind. Auf der Gruppenebene gibt es also offensichtlich eine eigene Problemlösungsfähigkei, die erst durch das Zusammenspiel mehrer Agenten verfügbar wird. Damit stellt sich die Frage nach den Eigenschaften eines MAS, die diese organisationale Problemlösungsfähigkeit erzeugen (oder verhindern).
Die jüngere Organisationsforschung hat dazu zwei verschiedene Ansätze zur sogenannten "Organizational Intelligence" (OI) hervorgebracht: Zum einen die japanische Differenzierung in organisationale Prozeß- und Produktintelligenz (die eng mit der Unterscheidung in Ablauf- und Aufbauorganisation verbunden ist) und zum andern den US-amerikanischen Ansatz, die für OI notwendigen Eigenschaften (betriebswirtschaftlicher) Organisationen zu betrachten, der zahlreiche Querbezüge zur traditionellen Organisationstheorie besitzt. Beide Ansätze zielen jedoch im Kern auf die Bereitstellung von Richtlinien für "OI-gerechtes" organisationales Design.

Organisationstheoretische Konzepte können beim gegenwärtigen Stand der Forschung jedoch nicht ohne weiteres auf MAS übertragen werden. So existieren innerhalb der Verteilten KI bisher keinerlei Verfahren für ein (gezieltes) organisationales Design von MAS. Es ist nicht einmal bekannt, ob es, ähnlich wie in der Organisationslehre, überhaupt klar unterscheidbare Dimensionen organisationaler Gestaltung gibt.

Ziel und Zweck des Aufsatzes lassen sich vor diesem Hintergrund wie folgt präzisieren: Zum einen motivieren die Ausführungen, daß sich die MAS-Forschung deutlich stärker als bisher mit den intellektuellen Eigenschaften von MAS auf einer organisationsbezogenen Ebene befassen sollte. Zum andern wird am konkreten Beispiel OI ein Weg aufgezeigt, wie organisationstheoretische Erkenntnisse Schritt für Schritt in Entwurfsrichtlinien für MAS umgesetzt werden können. Auf diese Weise wird es möglich, MAS hinsichtlich ihrer globalen Eigenschaften so zu entwickeln, daß sie ganz gezielt in menschliche Organisationen eingebettet werden und dort organisationale Prozesse informiert und effektiv unterstützen können. Nach Ansicht der Verfasser stellt das eine notwendige Voraussetzung auf dem Weg zu integrierten Mensch-Computer-Organsiationen dar.

Der Aufsatz führt zunächst in das organisationstheoretische Konzept der Organizational Intelligence ein. Auf dieser Grundlage wird ein Rahmenkonzept entwickelt, welches die wichtigsten der für die Herausbildung von OI in Multiagentensystemen relevanten Eigenschaften umfaßt. Dazu zählen insbesondere das Organisationale Problemlösen, Organisationales Lernen und Gedächtnis, die Selbstorganisation, Anpassungsfähigkeit sowie Flexibilität von MAS bzgl. dynamischer Entwicklungen in der MAS-Umwelt, und schließlich die Fähigkeit von MAS, mit ihrer Umwelt (zielgerichtet und problembezogen) zu kommunizieren. Diese für ein MAS-orientiertes OI-Konzept als wesentlich angesehenen Eigenschaften werden dann einer Detailbetrachtung unterzogen. Die Detailbetrachtung zielt, dem Gesamtansatz sowie dem aktuellen Stand der Arbeiten entsprechend, auf die Ableitung von Entwurfshinweisen, welche die Herausbildung intellektueller Eigenschaften auf der organisationaler Ebene in MAS unterstützen können. Das Schlußkapitel faßt die Ergebnisse zusammen und gibt einen Ausblick auf nun anstehende Arbeiten.

2 ORGANISATIONSTHEORETISCHE GRUNDLAGEN

2.1 Zum Begriff der Organisational Intelligence

Dem japanischen Ansatz folgend kann *Organisational Intelligence* (OI) als das intellektuelle Potential einer Organisation aufgefaßt werden und beschreibt damit die kollektive Problemlösungsfähigkeit einer Unternehmung /22/. In diesem Sinn besteht OI aus der Gesamtheit an geordneten Informationen, Erfahrung, Wissen und Verstehen. Sie integriert die in einer Organisation vorhandene menschliche und maschinelle Intelligenz. Dabei ist es nötig, über diese Intelligenz nicht nur auf individueller und Gruppenebene, sondern auch organisationsweit verfügen zu können. Hierzu bedarf es der Koordination seitens der Führung. OI enthält zwei sich wechselseitig bedingende Komponenten: (1) OI als dynamischer (Arbeits-) Prozeß und (2) OI als (statisches) Produkt. Der Prozeßansatz dient der Analyse von Organisationen und hat die Interaktionen sowie die Integration menschlicher und maschineller Intelligenz zum Inhalt. Er gliedert sich in Wahrnehmungs-, Speicherungs-, Lern-, Kommunikations- und Entscheidungsprozesse. OI als Produkt bezeichnet das System strukturierter, synthetisierter und zielgerichteter Information als Resultat der Gesamtheit der OI-Prozesse innerhalb der Organisation. Hierbei werden drei aufeinander aufbauende Stufen unterschieden: (1) Daten sind geordnet, aber ohne inneren Zusammenhang, (2) Information enthält eine auf das Unternehmensziel ausgerichtete Ordnung und (3) Wissen ist aktiv genutzte Information. Information entsteht jeweils ad hoc auf eine bestimmte Zielsetzung ausgerichtet und enthält informelle, offizielle und geheime Nachrichten sowie daraus abgeleitete Botschaften. OI als Produkt entsteht insbesondere dann, wenn die Informationssysteme einer Organisation deren Problemlösungsfähigkeit unterstützen. Dazu sind Entwurfsrichtlinien zu entwickeln, die ein "OI-gerechtes" Design von Informationssystemen unterstützen.

Der US-Ansatz zur OI wurde v.a. durch das von G. Huber auf der HICSS-1987 gehaltene Tutorial über "Intelligent Organizations" geprägt. Im Mittelpunkt dieses Beitrags stand die "lernende" Organisation. Danach gründet sich intelligentes organisationales Verhalten auf: (1) Kenntnis der Organisationsziele, (2) Wissen über alternative Handlungsmöglichkeiten, (3) die Fähigkeit, die jeweils am besten geeignete Handlungsalternative zu erkennen, auszuwählen und zu verfolgen, (4) Lernfähigkeit und (5) organisationales Gedächtnis. Dabei ist zu beachten, daß die Intelligenz in Gesellschaften immer wichtiger wird, denn weniger intelligente Gesellschaften werden, so Huber /13/, langfristig nicht überleben.

2.2 Japanische Forschungsansätze

Sunita /36/ bildet Kennzahlen, um die Leistungsfähigkeit von Unternehmen zu messen. In quantitativer Hinsicht mißt er die physische Stärke eines Unternehmens mit einer Funktion aus Arbeit und Kapital. Er definiert, um OI zu messen, einen ordinal skalierten Indikator α und untersucht japanische Unternehmen in den Zeiten beider Ölkrisen. Als Zeichen von OI sieht er dabei die Fähigkeit eines Unternehmens, mit Krisen umgehen zu können, und die Effektivität, diese zu überwinden.

Organisations- und Informationsstruktur konstatieren Yamamoto, Nakano und Matsuda /40/ als zwei wichtige Säulen für Unternehmen. Die beiden wichtigsten Organisationsstrukturen, die hierarchische und die dezentralistisch-autonome Struktur, sind in bezug auf Effektivität und Systemverfügbarkeit reziprok zueinander. Durch Transformation und Selbstdynamik muß die optimale Struktur, z. B. ein Netzwerk, gefunden werden. Zugriff auf Information ist eine Erscheinungsform von OI.

Teramoto, Iwaski und Richter /37/ stellen fest, daß Unternehmen durch schwindende Konkurrenzfähigkeit mit ehemaligen Wettbewerbern zusammenarbeiten müssen. Sie zeigen auf, wie japanische und europäische Chemie- und High Tech-Unternehmen zusammenarbeiten und dadurch voneinander lernen. Das Lernen wird hierbei als dynamischer Prozeß und Quelle allen organisatorischen Wechsels verstanden. In der Chemiebranche suchen europäische Partner mehrere japanische Partner, während japanische Unternehmen in der High Tech-Branche meist auf einen oder wenige europäische Partner setzen. Europa gibt dabei Technologie preis und erhält Markteintritt. Europäer sind Japans Stellvertreter im Kampf gegen Amerika. Europäer wollen im High Tech-Bereich Zugriff auf japanische Technologie, während für Chemieunternehmen das japanische Marktwissen interessant ist. Auf beiden Seiten geht es um Information.

Niwa /25/ plädiert für Knowledge Sharing Systems, um menschliche und maschinelle Intelligenz zu vereinigen. Folgende grundlegende Fragen tauchen hierbei auf: Wie kann menschliches und Computerwissen verbunden werden, wie können Organisationen Einzelwissen aufnehmen und wie können intelligente Tätigkeiten von Menschen und Organisationen erleichtert werden? Klassische Expertensysteme helfen nach Niwa nur Neulingen und sind so Wissenseinbahnstraßen. Die Wissensbasis verändert sich zudem nach der Implementierung nicht mehr. Menschliche Experten nutzen ihr Wissen intuitiv. Deshalb stellt Niva ein kooperatives Mensch-Computer-System vor, in dem der Mensch mit seinen intuitiven und verbindenden Fähigkeiten mit den logischen Funktionen des Computers arbeitet. Der Computer macht Vorschläge und sucht in den Wissensbasen nach Schlüsselwörtern. Die Wissensbasis vereint unterschiedliche, unabhängige Quellen, so daß der Wissensaustausch zweiseitig wird. Somit entsteht Organisationswissen.

Watanabe /39/ definiert organisationales Lernen als Informationsaustausch von Organisationsmitgliedern. Hierzu entwirft er ein integriertes System aus Computernetzwerk und menschlichem Netzwerk als organisationsweites Informationsnetzwerk, um Lernprozesse zu beschleunigen. Dieses soll Wirtschaftlichkeit und Wirksamkeit steigern und OI als Basis für Organisationsstrategien meßbar machen.

Lose gekoppelte Systeme sieht Tsuchiya /38/ als Organisationsstil der Zukunft. Er stellt die These auf, daß Informationstechnologien durch organisationales Lernen und Formulierung angemessener Strategien Entscheidungsprozesse in lose gekoppelten Systemen verbessern. Strategische Entscheidungen zu treffen, fällt lose gekoppelten Organisationen schwer, da sie neue Strategien anwenden müssen. Organisationales Lernen, das von Natur aus unvollständig ist, ist hier besonders schwierig. Die Bildung von Organisationswissen ist durch Informationsinterpretationssysteme begrenzt. Quelle des Organisationswissens ist das schwer zu fassende individuelle Wissen der Mitarbeiter. Durch Gespräche wird dieses Wissen andern zugänglich und zu Gruppenwissen. Erst ein Interpretationssystem macht es zum Organisationswissen. Informationstechnologie soll dem einzelnen Freiräume für kreative Tätigkeiten schaffen und den Prozeß von Einzel-, Gruppen- und Organisation als Ganzem sowie den Wissensaustausch innerhalb der Organisation fördern.

2.3 US-amerikanische Forschungsansätze

Problemlösung durch Informationsverarbeitungssysteme zeigten Newell und Simon /24/ schon 1969 auf. Sie stellten dabei zwei Methoden vor. Auf der Basis von Generate and Test stellten sie ein heuristisches Suchverfahren vor. Das Ziel kann explitzit als Symbolstruktur dargestellt werden oder implizit in der Methode enthalten sein. Ein Informationsverarbeitungssystem arbeitet sequentiell und aktiviert jeweils nur eine Methode. Während einer Anwendung kann eine Methode neue Probleme erzeugen. Gemessen werden die Methoden daran, ob Ziele erreicht werden und wie schnell und wie gut dies geschieht. Methoden müssen dabei nicht unabhängig voneinander sein. Der Fehlschlag einer Methode kann Informationen für die Auswahl der nächsten liefern. Wichtig ist die Wiedererkennung von Mustern. Bei der Generate and Test Methode kommt es darauf an, eine billige Möglichkeit zur Generierung möglicher Lösungen und Tests zu finden sowie einen überschaubaren Lösungsraum zu haben. Generierungsprozeß und Testprozeß sind dabei völlig unabhängig. Bei heuristischen Methoden sucht man nicht den Pfad für das Problem, sondern das letzte Element des Pfades. Je nach Problem wird hier direkt ein Weg generiert oder aber ein Suchbaum erstellt. Durch Rückkopplung zwischen Generierung und Test wird so der Lösungsraum eingeschränkt.

US-amerikanische Unternehmen befinden sich nach Marsden und Pingry /21/ in einer Zeit schnellen Strukturwandels. Gründe sind die schnell wachsenden Möglichkeiten und fallenden Kosten von Kommunikation und Computertechnologie sowie der verschärfte globale Wettbewerb. Die Autoren stellen drei Ausprägungen einer intelligenten Firma vor: Die technologisch intelligente Firma nutzt stets die neueste Kommunikations- und Computertechnologie, ersetzt Arbeit durch Kapital und maximiert so den Technologienutzen, die intelligente Firma nach Huber und McDaniel /14/ (s. nächster Absatz), die sich an 10 Richtlinien zur Entscheidungsfindung orientiert und Comupter-Kommunikations-Technologie nutzt, sowie die ökonomisch intelligente Firma (vgl. /13/), die genug Gewinn zum Überleben macht. Gemeinsam ist jeder intelligenten Firma, daß sie eine gute Organisationsstruktur hat und gute Entscheidungen fällt. Der computergestützte Entscheidungsprozeß birgt auch Gefahren, da das Fällen von Entscheidungen komplexer wird. Weiterer Forschungsbedarf besteht in punkto Wettbewerbsfähigkeit, Flexibilität und Adaption sowie den Gruppenentscheidungsprozessen. Ziel ist es, eine ausgewogene Technologiemischung und eine effiziente Organisationsstruktur zu finden.

Uberleben ist ein Unternehmensziel in einer Zeit steigenden Wissens, steigender Komplexität und steigender Turbulenzen /14/. Hierzu sollen Entscheidungen auf der Ebene getroffen werden, auf der die Kosten minimal sind und die Komplexität handhabbar ist. Für Routine- und fallweise Entscheidungen müssen unterschiedliche Entscheidungsschemata bereitgestellt werden. Die Informationen müssen den Entscheidungen angepaßt sein, wobei kurze Kommunikationsketten angestrebt werden. Die Leistung des Entscheidungssystems soll durch ein qualitativ orientiertes Belohnungssystem maximiert werden. Entscheidungen sollten als Projekt organisiert werden.

Kognitive Grenzen der Rationalität zeigen March und Simon /20/ auf. Organisationsmitglieder verhalten sich nie vollständig rational, zudem sind ihr Wissen und ihre Fähigkeit zur Problemlösung begrenzt. Beim Entscheidungsmechanismus müssen nach March und Simon alle Alternativen und Konsequenzen sowie eine vollständige Nutzenordnung bekannt sein, um vollständige Rationalität zu ermöglichen. Die komplexe Realität muß in ein einfaches Modell abgebildet werden können, damit Problemlösungsmechanismen greifen. Hierzu zerlegt man es in nahezu unabhängige Teile, die der menschliche Verstand jederzeit erfassen kann. Damit besteht der Preis handhabbarer Komplexität in einer nur befriedigenden Lösung. Eine Organisationsstruktur muß sich schnell Umweltveränderungen anpassen können und so gestaltet werden, daß Kommunikationsprobleme minimiert werden, da bei zeitkritischen Entscheidungen nur sofort zugängliche Informationen berücksichtigt werden können. Werden Verfahren geändert, so entspricht kurzfristige Anpassung eher der Problemlösung, während langfristige dem Lernen gleichkommt.

Computerbasierte Informationssysteme benötigen nach Paradice /27/ Attribute intelligenten Verhaltens. Ein Charakteristikum von Entscheidungsprozessen ist ein "time lag" zwischen dem Fällen einer strategischen Entscheidung und deren Ausführung. Ein zielgerichtetes Informationssystem, das Ziele speichert und die Ausführung überprüft, sollte das Problem des "time lag" lösen helfen können. Dieses Informationssystem ist durch Nachrichtenweiterleitung, -zusammenfassung, -verzögerung und -veränderung charakterisiert. Es soll von einer passiven zu einer aktiven Rolle im Entscheidungsprozeß kommen. Informationssysteme sollen auf Informationen reagieren und in der Ausbaustufe Information prognostizieren. Hierzu muß die Speicherfunktion eine aktive Rolle spielen, und der Einfluß von Organisationszugriffen auf "time lags" muß minimiert werden.

Nunamaker et. al. /26/ stellen Hilfsmittel für das Krisenmanagement vor. Intelligenz heißt lernen, übernehmen, verstehen und Probleme lösen. Organisationen wissen oft nicht, was sie wissen. Somit sind Entscheidungen, die auf der Verfügbarkeit von Wissen beruhen, schwierig effizient zu treffen. Entscheidungen zu treffen ist das zentrale Ziel organisatorischer Tätigkeiten. Diese müssen im erkenntnismäßigen, sozialen und organisatorischen Kontext getroffen werden. Krisen müssen vor- und nachbereitet werden, um im Zeitpunkt selbst effizient tätig werden zu können. Es müssen also Abläufe und Möglichkeiten zur Erkennung von Krisen vorgesehen werden. Gruppenentscheidungen müssen hierzu entwickelt, trainiert und flexibel gemacht werden. Eine intelligente Organisation zeichnet sich dadurch aus, daß sie aus (Krisen-) Erfahrung lernt.

2.4 Dimensionen der Organisational Intelligence

Dimensionen

Der *Problemlösungsprozeß* /23/ besteht darin, Informationen zu beschaffen, auszuwerten und zu konsensfähigen Aussagen zu kommen. Eine sinnvolle Reihenfolge aller Schritte ist für die Qualität der Lösung wichtig. Bei der Wahl einer Technik und deren Perfektionsgrad ist auf die Relation des zeitlichen, personellen und finanziellen Aufwandes zur Qualität der Lösung zu achten. Dabei soll die angewandte Technik von allen Beteiligten verstanden werden, damit Ergebnisse und Aussagen nachvollziehbar sind. Nahe verwandt dem Problemlösungsprozeß ist der Prozeßbegriff, unter dem die dynamische, ablauforientierte Analyse von Organisationen subsumiert wird. Eine statische Strukturbetrachtung der Informationssysteme steht hingegen beim Begriff der "organisational product intelligence" im Vordergrund.

Organisationales Lernen läßt sich nach Reber /30/ in bezug auf die Gesamtheit aller Mitglieder einer Organisation sowie auf die Beziehungen und Interaktionen von Personen in einer Organisation definieren. Organisationales Lernen findet auf drei Ebenen statt, der individuellen Ebene, dem Lernen

der einzelnen Organisationsmitglieder, der Gruppenebene, deren Erfolg von den Eigenschaften der Mitglieder und den Intergruppenverflechtungen abhängt, und der gesamtorganisatorischen Ebene, auf der die Gruppenleistungen zu einem Ganzen verknüpft werden. Zum Lernen an Modellen ist die Symbolisierung von Merkmalen und Eigenschaften des Modells Voraussetzung. Um Wissen nutzbar zu machen muß es kommunizierbar, konsensfähig und integrierbar sein. Mit der Maximierung ihres Bestrebens nach Kompetenz laufen Individuen und Organisationen zunehmend Gefahr, ihre Rationalität zu verlieren. Nowendige Voraussetzung organisationalen Lernens ist das Vorhandensein eines *organisationalen Gedächtnisses*.

Den Begriff *Selbstorganisation* (Self Design) prägten die Sozialwissenschaften. In den Organisationswissenschaften versteht man heute darunter einen ganzheitlichen Ansatz, der "alle Prozesse, die aus einem System heraus von 'Selbst' Ordnung entstehen lassen verbessern oder erhalten" umfaßt /29/. An die Stelle von sich lösenden festen Strukturen und Verhaltensweisen treten Flexibilität, Kreativität, Innovation etc. Auf materieller Ebene sind Organisationsinstrumente, Aufgabenträger, Führungssysteme, auf geistig-symbolischer Ebene Wertesysteme, Empfindungen und Gefühle betroffen. Ein Unternehmen beobachtet sich selbst und strukturiert sich selbst um.

Unter *Flexibilität* wird ein Entscheidungsverfahren verstanden, das in einem mehrstufigen Entscheidungsprozeß unter Ungewißheit bei zunehmender Information Verhalten kurzfristig anpaßt. *Anpassungsfähigkeit* einer Unternehmung ist eine langfristige Reaktion auf das Marktgeschehen und drückt sich durch Strukturänderungen aus.

3 ORGANISATIONAL INTELLIGENCE IN MULTIAGENTENSYSTEMEN

3.1 Organisationstheoretische Grundlagen der Multiagentensystem-Forschung

In der Literatur zu Multiagentensystemen finden sich zahlreiche Hinweise darauf, daß Gruppierungen kooperativer Agenten in der einen oder anderen Form "organisiert" sind oder sein müssen. Begriffe wie Organisation kooperativer Agenten und Organisationsstruktur /33/, /5/, /7/, /10/, /18/, /19/, Rollen und Rollenverteilung /5/, /34/, Entscheidungsprozeduren /12/, Individual- und Gruppenintentionen /32/, /16/, Verpflichtung /3/ und Verantwortlichkeit /4/, /15/, /9/ prägen damit weite Teile der Diskussion zu Multiagentensystemen.
Allerdings haben die verschiedenen Anleihen der Verteilten KI in den Sozial-, Organisations- und Wirtschaftswissenschaften bis heute nicht dazu geführt, daß sich die Verteilte KI-Forschung ernsthaft mit der Entwicklung einer Organisationstheorie für Multiagentensysteme befaßt /8/. So gibt es bis heute keinerlei Ansätze, Dimensionen organisationalen (MAS) Designs zu entwickeln noch stehen Methoden zur Verfügung, mit denen organisationale Eigenschaften und Verhaltensweisen systematisch beschrieben, analysiert und erklärt werden können. Beide Forderungen stellen jedoch notwendige Voraussetzungen für zielgerichtetes und effektives, auf die Erzeugung ganz bestimmter MAS-Eigenschaften ausgerichtetes MAS-Design dar.
Die nachfolgenden Ausführungen zeigen am Beispiel der OI einen Weg auf, wie organisationstheoretische Konzepte auf Multiagentensysteme übertragen werden können. Dazu wird für die in Kapitel 2.4 genannten organisationalen Eigenschaften jeweils (1) zunächst der MAS-Bezug diskutiert, (2) dann wesentliche aus der Organisationsforschung ableitbare Forderungen ermittelt und (3) die dazu bereits existierenden MAS-Arbeiten skizziert, um auf der so gegebenen Grundlage (4) erste Entwurfsüberlegungen anzustellen und (5) die sich daraus ergebenden Konsequenzen für die MAS-Entwicklung resp. -Forschung zu analysieren.

3.2 Organisationales Problemlösen

Nach allgemein akzeptierter Ansicht sollen MAS Probleme lösen können, die für die Fähigkeiten ihrer einzelnen Mitglieder zu umfangreich, kompliziert oder ressourcenintensiv sind. Damit verfügt ein MAS über Fähigkeiten, die nicht in der Summe der Problemlösungsfähigkeiten der einzelnen Agenten enthalten sind. Beim Entwurf von Multiagentensystemen kann es also nicht nur darum gehen, individuelle Fähigkeiten intelligent zu kombinieren (additive Verknüpfung). Stattdessen ist es erforderlich, auch auf der organisationale Ebene explizite MAS-Problemlösungsfähigkeiten zu entwickeln.

In der Organisationsforschung wird diese Einsicht bereits seit langem als die grundlegende Bedingung überhaupt für die Existenz, den Erfolg und die langfristige Überlebensfähigkeit von Organisationen angesehen. Wesentliche damit verbundene Themen sind zum Beispiel die (positive oder negative) organisationale Produktivität, die Anreiz-Beitrags-Theorie[1] und die Erkenntnis, daß die Effizienz organisationaler Prozesse von der zugrundeliegenden Organisationstruktur abhängig ist. Geht man weiter davon aus, daß Organisationsstrukturen als eine Art langfristige Problemlösungsstrategie interpretiert werden können, dann folgt sowohl für sozio-technische Systeme als auch für rein softwarebasierte MAS, daß es neben der individuellen offensichtlich auch eine organisationsbezogene Ebene des Problemlösens geben muß.
Effektives und effizientes Problemlösen setzt lösungsrelevantes Wissen, die Verfügbarkeit geeigneter Problemlösungsstrategien und die Fähigkeit zur Steuerung der Problemlösung ("metalevel control") voraus. Für jede Organisation ist damit zu fragen, wann individuelles Wissen zu organisationalem Wissen wird (und umgekehrt), welche Organisationsmitglieder über die Fähigkeit und Befugnis zur Einflußnahme auf organisatiolnale Prozesse verfügen und welche Problemlösungsstrategien (an welchen Stellen der Organisation) zur Verfügung stehen.

Das Verhalten von Multiagenten-Organisationen ist bis heute kein zentraler Gegenstand der MAS-Forschung, es ist nicht einmal wirklich klar, was in MAS unter "organisationalem Verhalten" oder, enger, unter "organisationalem Problemlösen" zu subsumieren wäre /5/, /11/. Die heutige MAS-Forschung stellt stattdessen ganz überwiegend auf das Problemlösungsverhalten des einzelnen Agenten und dessen individueller Teilnahme an organisationalen Prozessen (z.B. Planen, Verhandeln, Dialoge) ab. Andererseits gibt es durchaus Arbeiten zur Modellierung und Repräsentation organisationalen Wissens (vgl. /2/, S. 28f.) und die Erkenntnis, daß eine übergeordnete Steuerung organisationaler Prozesse für die Kohärenz kooperativer Problemlösungsprozesse notwendig oder zumindest hilfreich ist (Beispiel: wissensbasierte Scheduler in Blackboardsystemen).

Setzt man diese Überlegungen in einen Bezug zur Entwicklung von MAS-Systemen, dann ergibt sich daraus offensichtlich die Notwendigkeit, MAS-Architekturen um eine explizit zu modellierende (jedoch nicht notwendig zentralisierte) Organisationsschicht zu erweitern, innerhalb derer als Minimalanforderung zumindest das organisationale Wissen, die organisationalen Problemlösungsstrategien sowie Komponenten zur Steuerung organisationaler Prozesse enthalten sein müssen. Das wiederum macht es erforderlich, die Transformation individuellen in organisationales Wissen (und vice versa) zu modellieren sowie unterschiedliche Arten von Zugriffsrechten auf organisationales Wissen und organisationale Problemlösungsstrategien zu definieren[2]. Gleichzeitig stellt eine solche Organisationsschicht die notwendige Basis dar, um organisationales Verhalten analysieren (und gestalten) zu können, um MAS (auf der Basis von organisationalen Selbstmodellen) selbstauskunftsfähig (und damit dialogfähig) zu machen, was letztendlich als notwendige Voraussetzung für die Schaffung integrierter Mensch-Computer-Teams anzusehen ist.

3.3 Organisationales Gedächtnis

Die Frage nach einem organisationalen Gedächtnis hängt einerseits eng mit der Speicherung organisationalen Wissens (Gedächtnis als logischer Ort der Wissensablage) zusammen, geht andererseits aber auch deutlich über diese hinaus. So muß ein Gedächtnis nicht nur Mechanismen zur Wissensrepräsentation, -indexierung und (physischen) Ablage bieten, sondern auch Verfahren zur Verfügung stellen, mit denen - bezogen auf ein konkretes Problem in einer konkreten Situation - das jeweils relevante organisationale Wissen aufgefunden und zusammengeführt werden kann.

Offensichtlich stellt das organisationale Gedächtnis eine notwendige Voraussetzung für alle Arten "höherer" intellektueller Eigenschaften (Anpassungs- und Lernfähigkeit, Rationalität, Intelligenz) von Organisationen dar. Deshalb befaßt sich die Organisationsforschung bereits seit langem und intensiv mit der Frage, wie organisationales Wissen abzulegen und zu repräsentieren ist und wie sichergestellt werden kann, daß exakt auf dieses Wissen genau dann zugegriffen wird, wenn ein entsprechendes Problem gegeben ist. Jede Art von Notfallplänen - vor allem in Bereichen mit hoher Personal-

---

1 Organisationsmitglieder verlassen eine Organisation dann, wenn sie ihre Beiträge höher bewerten als die von dieser (zurück-) erhaltenen Anreize.

2 Diese Forderung bedingt eine weitgehende Differenzierung von Rollen- und Strukturkonzepten in MAS.

fluktuation - illustriert das Problem, und moderne Workflowmanagementsysteme können als Beispiel dafür angesehen werden, wie das organisationale Gedächtnis DV-gestützt verbessert werden kann.

Der Begriff des organisationalen Gedächtnisses spielt im gesamten Bereich der Verteilten KI eine völlig nachrangige Rolle. In den meisten MAS gibt es zwar eine Art temporäres Gedächtnis, welches die Historie einer gerade ablaufenden kooperativen Problemlösung enthält (Kontexte, Argumentationsketten, etc.) und manchmal auch die während des Problemlösens mit anderen Agenten gemachten Erfahrungen (Verteilung von Wissen und Fähigkeiten im Verbund, Zuverlässigkeit von Kooperationspartnern). Es wird jedoch völlig darauf verzichtet, das dabei erworbene Wissen langfristig zu speichern und für zukünftige Aufgaben (neue auftretende Probleme, Selbstorganisation, Lastverteilung, etc.) zu nutzen. Damit ist bis heute nicht möglich, Konzepte organisationaler Selbstreflektion zu entwickeln, die ihrerseits eine grundlegende Voraussetzung für jede effektiv arbeitende Steuerung organisationaler Prozesse oder Kontrolle resp. Verbesserung organisationalen Verhaltens darstellt.

Für den Entwurf intelligenter MAS bedeutet das, die Organisationsschicht um eine leistungsfähige Gedächtniskomponente zu erweitern. Eine genauere Betrachtung der von dieser Komponente zu erbringenden Leistungen legt es nahe, beim Entwurf fallbasierte Konzepte in Verbindung mit modellbasierter Wissensrepräsentation und -verarbeitung in Betracht zu ziehen. Ähnlich wie beim organisationalen Wissen kommt der Verteilung der Zugriffsrechte (einschließlich der Update-Verpflichtung!) auch bei der Modellierung organisationaler Gedächtnisse eine wesentliche Bedeutung zu, zumal diese sich wiederum weitgehend auf Rollendefinitionen, Rollenzuweisungen, Verteilung von Verantwortlichkeiten und organisationaler Macht auswirken. Die möglichen Interaktionen zwischen organisationalem Gedächtnis und den verschiedenen individuellen Agentengedächtnissen stellen schließlich einen weiteren wichtigen Entwurfsaspekt dar.

Die Einführung organisationaler Gedächtnisse in MAS wirft eine Reihe wichtiger neuer Fragen auf. Dazu zählen unter anderem die Konsistenz zwischen individuellem und organisationalem Wissen, die sich aus der Verteilung von Zugriffsrechten (und -pflichten) ergebenden Konsequenzen auf Rollendefinitionen und MAS-Organisationsstruktur (Organisatoren vs. Problemlöser), die Frage nach den Auswirkungen auf MAS-Koordinationsmechanismen (Planen, Verhandeln) oder die Bedeutung von Erfahrungen für die organisationale Zielfindung und Strategieentwicklung, die Steuerung organisationaler Problemlösungsprozesse oder das nach innen und außen gerichtete Konfliktmanagement.

3.4 Organisationales Lernen

Organisationalem Lernen wird in der klassischen Organisationsforschung seit langem eine wichtige Rolle für die Flexibilität und Anpassungsfähigkeit, mithin also für die (langfristige) Überlebensfähigkeit von Organisationen zugewiesen. So ist es ganz sicher kein Zufall, daß sowohl die japanischen als auch die US-amerikanischen Arbeiten zu OI völlig unabhängig voneinander die Lernfähigkeit als ganz wesentliche (Japan) bzw. als die entscheidende intellektuelle Fähigkeit von Organisationen überhaupt (USA) herausgestellt haben. Dabei besitzt Lernen in Organisationen zwei verschiedene Dimensionen: Zum einen dient Lernen der Verbesserung eigenen Verhaltens (Lernen zur Vereinfachung, Beschleunigung, und qualitativen Verbesserung organisationaler Prozeduren). Zum anderen beinhaltet Lernen eine antizipative Komponente, die eine gestaltende Einflußnahme auf zukünftige Ereignisse erst möglich macht. Diese stellt eine wesentliche Voraussetzung dafür dar, daß sich Organisationen auch in dynamischen Umwelten ihren Zielen und Möglichkeiten gemäß verhalten können.

Lernfähigkeit stellt eine notwendige Voraussetzug für individuelle (maschinelle) Intelligenz dar. Die Verteilte KI greift diese Erkenntnis für die einzelnen Agenten eines MAS durchaus auf und ist auch bereit, kollektive Agenten[3] zu modellieren. Trotzdem gibt es bis heute keine nennenswerten Überlegungen zu einer eigenständigen organisationalen Lernfähigkeit von MAS, die mehr leistet als das weitgehend auf Statistik beruhende "Tunen" der Auftragsverteilung im Verbund. Stattdessen gilt das Hauptinteresse auch hier wieder dem einzelnen Agenten. Dabei wird allerdings nicht nur auf die individuellen Problemlösungsfähigkeiten abgestellt, sondern es wird durchaus auch individuelles Lernen organisationsbezogener Aspekte betrachtet. Das ist eine etwas überraschende Erkenntnis vor dem Hintergrund, daß sich zahlreiche Arbeiten innerhalb der Verteilten KI mit der Unterstützung von

---

[3] Teams kooperativer Agenten, die mit anderen individuellen oder kollektiven Agenten zusammenarbeiten

Büroarbeit, verteilten und kooperativen Entscheidungsprozessen, verteilter Interpretation oder Mensch-Maschine-Kooperation befassen, also Anwendungen modellieren, in denen Lernen auf einer organisationalen Ebene in natürlicher Weise eine wesentliche Rolle spielt.

Bezogen auf den Entwurf von auf organisationaler Ebene lernfähigen MAS ist hier - ergänzend zu den bereits oben genannten Aspekten - vor allem danach zu fragen,

(1) welches Wissen auf der organisationalen Ebene eines MAS gelernt werden kann (und soll). Dazu zählen unter anderem der Erfolg / Mißerfolg organisationaler Problemlösungsprozesse, die Eignung / Angemessenheit organisationalen Wissens für bestimmte Fragestellungen und Situationen, Erfahrungen mit der (kontextbezogenen) "Nützlichkeit" von Organisationsmitgliedern, das Entwickeln / Verbessern organisationaler "Sensoren" zur Beobachtung der Umwelt, das Verhältnis zwischen Organisation und Umwelt, die Einschätzung organisationaler Stärken / Schwächen und Maßnahmen zur Steigerung der organisationalen Produktivität.

(2) ob (und wann) organisationales Lernen bottom-up, top-down oder in beide Richtungen erfolgen soll . Ein MAS lernt bottom-up, wenn Agenten individuelle Erfahrungen teilweise oder vollständig an die Multiagenten-Organisation weitergeben. MAS lernen top-down, wenn das MAS als solches lernt und das Erlernte im notwendigen Umfang an die Organisationsmitglieder notifiziert.

(3) auf welche Weise ein MAS durch Lernen Wissen aufnehmen kann. Lernen durch Interaktion bedeutet die Bewertung eigener Handlungen durch Dritte, in Dialogen gewonnenes Kontextwissen, durch Kooperation erhaltenes Wissen über Fähigkeiten und Zuverlässigkeit von Organisationsmitgliedern und -teilnehmern aufzunehmen. Lernen durch Introspektion zielt auf Verkürzung von Lösungsprozessen, Verknüpfung von zusammengehörigem Wissen und Entdeckung / Beseitigung von Inkonsistenzen.

Weitergehende Aspekte des organisationalen Lernens in MAS beziehen sich - neben den bereits diskutierten Konsistenzproblemen und Zugriffsrechten - vor allem auf die Frage, in welchem Umfang Lernen vollständig automatisiert oder aber nur halbautomatisch und unterstützt durch den Menschen ablaufen kann.

3.5 Selbstorganisation

Der klassische Ansatz zum Aufbau von MAS integriert bereits existierende Agenten in eine Multiagenten-Organisation. Damit stellt sich sofort die Frage nach der aufbau- und ablauforganisatorischen Gestaltung der Organisation, und nach der Verteilung von Aufgaben, Kompetenzen und Verantwortung zwischen den Agenten ebenso wie nach der Herausbildung organisationaler Ziele, Strategien und Verhaltensweisen. Diese Gestaltungsfragen können bei Einrichtung des Systems zunächst durch den Entwickler vorgegeben werden. Soll das System jedoch als solches stabil bleiben und auf Änderungen in seiner Umwelt flexibel reagieren können, dann muß das MAS imstande sein, seine Aufbau- und Ablauforganisation den aktuellen Belangen entsprechend zu modifizieren und seine organisationalen Prozesse an der Veränderung von Anforderungen auszurichten.

In der Organisationstheorie gilt die Selbstorganisation als eine klassische "höhere" organisationale Fähigkeit. Sie setzt im allgemeinen voraus: (1) organisationale Wahrnehmungsfähigkeiten und die Fähigkeit zur Selbstbeobachtung, (2) organisationales Wissen über den Zusammenhang zwischen Organisationsstruktur (Aufbau / Ablauf), möglichem Organisationsverhalten und organisationalen Problemlösungsfähigkeiten, (3) Kenntnis über und Verfügbarkeit von Reorganisationsstrategien und (4) organisationales Selbstwissen.

In der aktuellen MAS-Forschung werden organisationsbezogene Aspekte faktisch durch den Systementwickler festgelegt. Teilweise besteht, wie im DVMT, auch die Möglichkeit, daß sich MAS dynamisch um- und reorganisieren. Allerdings sind die in diesem Bereich bisher verfolgten Ansätze noch weitgehend "konventioneller" (Statistik-basierter) Natur und leisten keinen Beitrag zur Ausbildung informiert-intelligenter Selbstorganisationsfähigkeiten in MAS.

Im Hinblick auf einen OI-orientierten MAS-Entwurf ist festzustellen, daß ganz offensichtlich Instanzen existieren müssen, die auf der organisationalen Ebene für die Selbstbeobachtung und Umgebungswahrnehmung zuständig sind und die verschiedenen Selbst- und Umweltmodelle aktuell halten.

Zusätzlich sind Reorganisationsstrategien zu entwickeln und das für die Anwendung dieser Strategien notwendige Wissen bereitzustellen. Hier ergibt sich allerdings sofort die Schwierigkeit, daß das Wissen um diese Zusammenhänge in der MAS-Forschung erst ganz allmählich zu entstehen beginnt.

Konsequenzen für Entwicklung und Einsatz von MAS sind nach diesen Überlegungen vor allem im Bereich der Rollendifferenzierung zu erwarten. Es wird eindeutig klar, daß organisational-intelligente MAS auf eine Differenzierung in "Organisationsagenten" und "domänenorientierte Problemlöser" nicht werden verzichten können. Damit wird eine weitgehende Neuorientierung der organisationstheoretischen Fundierung von MAS einhergehen. Das bietet dann jedoch Möglichkeiten, Konzepte wie Flexibilität und Anpassungsfähigkeit methodisch sauber zu definieren, zu analysieren und zu entwickeln. Des weiteren muß davon ausgegangen werden, daß ein MAS erst durch die Fähigkeit zur Selbstorganisation in die Lage versetzt wird, reale Anwendungen wie die Mitarbeit seiner individuellen oder kollektiven Agenten in verschiedenen, zeitlich parallel oder überlappend ablaufenden Projekten zu unterstützen.

3.6 Interaktionen zwischen MAS und Umwelt

MAS können einerseits als (weitgehend) geschlossene, rein softwarebasierte Systeme modelliert werden, die nur in geringem Umfang mit ihrer Umgebung interagieren. Dieses Szenario entspricht weitgehend dem gegenwärtigen Stand der Dinge. Andererseits wird seit Jahren gefordert, daß MAS durchaus nicht nur künstliche Agenten, sondern auch den menschlichen Problemlöser als Kooperationspartner integrieren sollten. Das setzt jedoch voraus, daß das künstlich geschaffene System sich mit dem menschlichen Teammitglied auch geeignet verständigen kann. Hier sind jedoch noch nicht einmal die Fragen bekannt, die durch diese Forderung aufgeworfen werden und notwendigerweise beantwortet werden müßten.

In der Organisationsforschung ist es dagegen völlig unstrittig, daß Organisationen in der Lage sein müssen, über sich selbst und ihr Verhalten Auskunft zu geben. Das führt zu der Dokumentation von Abläufen und Ergebnissen sowie zur Kodifizierung organisationalen Wissens.

Im engeren Bereich der MAS-Forschung fehlen derartige Konzepte, die jedoch - auf dem Umweg über die Entwicklung von sogenannten User Agents - derzeit beginnen, sich auch in der Verteilten KI zu etablieren. Wenn User Agents auf Basis eines MAS-Ansatzes realisiert werden sollen, dann benötigen sie für den Dialog mit ihrem lokalen (menschlichen) User notwendigerweise Wissen nicht nur über den einzelnen Agenten, sondern über das MAS als solches. In diesem Bereich kann man sich des Eindrucks im übrigen nicht ganz erwehren, daß die MAS-Forschung das Problem der User-MAS-Kommunikation zumindest bisher etwas unterschätzt hat.

Was bedeutet das nun für das MAS-Design? - Offensichtlich setzt die Forderung nach organisationaler Interaktionsfähigkeit eine (vermutlich stark anwendungsorientierte) Sprache voraus, mit der die Kommunikation zwischen System und Umwelt durchgeführt werden kann. Diese muß mindestens dafür geeignet sein, organisationale Prozesse zu erläutern und dem Anwender steuernde Eingriffe zu ermöglichen. Damit stellt sich wiederum die Frage nach einer weiteren Rollendifferenzierung, denn in den meisten Fällen wird es nicht erforderlich sein, daß jedes Mitglied eines MAS über identische Fähigkeiten zur Kommunikation mit der Außenwelt verfügt.

Konsequenzen für die Entwicklung von MAS sind aus dieser Sicht vor allem im Bereich integrierter Mensch-Computer-Teams zu erwarten. Die bei der Sprachentwicklung zu erwartenden Probleme werden zunächst vor allem zu einer starken Einschränkung in der Offenheit von MAS sowie zu einer weitgehenden Anwendungsorientierung und Einbettung in vorab bekannte, wohldefinierte Applikationskontexte und damit zu einem gewissen Konflikt mit bisher postulierten Ansprüchen führen.

4 POTENTIAL UND PERSPEKTIVEN: DER (IS-) DESIGN-ANSATZ

Der Ausführungen dieses Aufsatzes machen deutlich, daß die Theorie und Entwicklung von Systemen kooperativer Agenten eine fundierte organisationstheoretische Grundlage voraussetzen, die heute jedoch noch nicht einmal ansatzweise gegeben ist. Am Beispiel der Organizational Intelligence konnte gezeigt werden, daß die Übertragung konkreter organisationstheoretischer Modelle auf MAS

grundsätzlich machbar ist und für die MAS-Forschung überdies recht fruchtbar sein kann. Allerdings hat die Diskussion auch gezeigt, daß bis heute keine Methoden zur Verfügung stehen, ein gezieltes organisationales MAS-Design durchzuführen. Damit bleiben die Möglichkeiten, Organisationen kooperativer Softwareagenten für bestimmte Zwecke zu entwerfen und zu gestalten ("Organizational Engineering") weit hinter den in der Organisationsforschung bekannten und praktisch verfügbaren Möglichkeiten zurück.
Vor diesem Hintergrund wurde in diesem Aufsatz versucht, am konkreten Beispiel der OI einen Weg aufzuzeigen, wie organisationstheoretische Konzepte für ein MAS-bezogenes Organizational Engineering verfügbar gemacht werden können. Dabei stand die Problemanalyse (im Sinn von Requirements Engineering) im Vordergrund, es wurde bewußt auf eine Diskussion möglicher Designalternativen für MAS-Architekturmodelle verzichtet. Gleichzeitig haben die Ausführungen sehr deutlich gezeigt, daß die Entwicklung eines Konzeptes für organisationales MAS-Design eine ausgesprochen lohnende Aufgabe darstellt, die deshalb in unseren zukünftigen Arbeiten auch eine wesentliche Rolle spielen wird.

5 LITERATURVERZEICHNIS

/1/ Blanning, R.W., King, D.R., Marden, J.R., Seror, A.C.: Intelligent Models of Human Organizations: The State of the Art, in: Journal of Organzational Computing, 2 1992, S 123-130.

/2/ Bond, A.; Gasser, L. (eds.): Readings in Distributed Artificial Intelligence, Morgan Kaufmann Publishers, San Mateo, CA., 1988.

/3/ Cohen, P.R.; Levesque, H.J.: Persistence, Intention, and Commitment. Technical Report CSLI-87-88, Center for the Study of Language and Information, Stanford University, Stanford, CA: March 1987.

/4/ Cohen, P.R.; Levesque, H.J.: Intention = Choice + Commitment. Proceedings of the 6th National Conference on AI (AAAI-87). S. 410-415.

/5/ Corkill, D.: A Framework for Organizational Self-Design in Distributed Problem Solving Networks. Ph.D. Thesis. Department of Computer and Information Science, Univ. of Massachusetts at Amherst, MA. COINS-TR-82-33. Dec. 1982.

/6/ Cyert, R.M., March J.G.: The Behavial Theorie of the Firm - A behavioral Science-Economics Amalgam, in: New Perspectives in Organizational Reasearch, New York 1964, S 289 - 299.

/7/ Durfee, E.H.; Lesser, V.L.; Corkill, D.: Coherent Cooperation Among Communicating Problem Solvers, IEEE Transaction on Computers, C-36, 1987, pp. 1275. Also in: A. Bond, L. Gasser (eds.): Readings in Distributed Artificial Intelligence, Morgan Kaufmann Publishers, San Mateo, CA., 1988, pp. 268.

/8/ Farhoodi, F.; Proffitt, J.; Woodman, P.; Tunnicliffe, A.: Design of Organisations in Distributed Decision Systems. AAAI-Workshop on Cooperation Among Heterogeneous Intelligent Systems, 1991.

/9/ Fickas, S.; Helm, R.: Acting Responsibly: Reasoning about Agents in a Mutli-agent System. Technical Report CIS-TR-91-02. Department of Computer and Information Science, University of Oregon, Eugene, OR 87403. 1991.

/10/ Fox, M.S.: An Organizational View of Distributed Systems, IEEE Transactions on Systems, Man and Cybernetics, SMC-11, 1981, pp. 70-80.

/11/ Gasser, L.: DAI Approaches to Coordination. In: Avouris, N.M.; Gasser, L. (eds.): Distributed Artificial Intelligence: Theory and Practice, Kluwer Academic Publishers, 1992. S. 31-52.

/12/ Ginsberg, M.: Decision Procedures. In M. Huhns (ed.): Distributed Artificial Intelligence, Morgan Kaufman Publishers, 1987. S. 3-28.

/13/ Huber, G.P.: The Nature and Design of Post-Industrial Organizations, in: Management Science, Vol. 30, 1984, S. 928 - 951.

/14/ Huber, G.P. und McDaniel, R.R.: The Decision-Making Paradigm of Organizational Design, in: Management Science Vol 32, 1986 S. 572 - 589.

/15/ Jennings, N.; Mamdani, E.H.: Using Joint Responsibility to Coordinate Collaborative Problem Solving in Dynamic Environments". Proceedings of the Tenth National Conference on AI (AAAI-92), San Jose, California. S. 269-275.

/16/ Jennings, N.: Joint Intentions as a Model of Multi-Agent Cooperation. Ph.D. Thesis. Queen Mary and Westfield College, Department of Electronic Engineering, University of London, UK. August 1992.

/17/ Luhmann, N.: Soziale Systeme - Grundriß einer allgemeinen Theorie, Frankfurt am Main 1984.

/18/ Malone, T.: Modeling Coordination in Organizations and Markets. Management Science, 33 (1987) 10, S. 1317-1332.

/19/ Malone, T.: Organizing Information Processing Systems: Parallels Between Human Organizations and Computer Systems. In W. Zachary, S. Robertson, J. Black (eds.): Cognition, Cooperation, and Computation. Ablex Publishing Corporation, Norwood, NJ, 1988.

/20/ March, James G., Simon, Herbert A.: Organizations New York 1958.

/21/ Marsden, J.R., Pigry, D.E.: The Intelligent Organization: Some Observations and Alternativ Views in: Proceedings of the Twenty-First Annual Hawaii Conference on System Sciences, 1988, S. 19 - 24.

/22/ Matsuda, T.: Organizational Intelligence: Its Significance as a Process and as a Product, in: Proceedings of CEMIT92/CECOIA 3 - International Conference on Economics, Management and Information Technologie, Tokio 1992, S. 219 - 222.

/23/ Nagel, P.: Techniken der Problemanalyse und -lösung, in Frese, Erich: Handwörterbuch der Organisation, Stuttgart 1992 S. 2014 - 2024.

/24/ Newell, A., Simon, H.A.: Human Problem Solving, Englewood Cliffs NY, 1972.

/25/ Niwa, K.: Knowledge Sharing Systems for Organizational Intelligence, in: Proceedings of CEMIT92/CECOIA 3 - International Conference on Economics, Management and Information Technologie, Tokio 1992, S. 227 - 230.

/26/ Nunamaker, J.F. Jr., Weber, E.S., Smith, C.A.P.: Crises Planning Systems: Tools for Intelligent Action, in: Proceedings of the Twenty-First Annual Hawaii Conference on System Sciences, 1988, S. 25 - 34.

/27/ Paradice, D.B.: The Role of Memory in Intelligent Information Systems, in: Proceedings of the Twenty-First Annual Hawaii Conference on System Sciences, 1988, S. 2 - 9.

/28/ Pondy L.R., Frost, P.J., Morgan, G., Dandrigde, T.C.: Organizational Symbolism, Greenwich London 1983.

/29/ Probst, G.J.B.: Selbstorganisation, in Frese, Erich: Handwörterbuch der Organisation, Stuttgart 1992 S. 2255 - 2269.

/30/ Reber, G.: organisationales Lernen, in Frese, Erich: Handwörterbuch der Organisation, Stuttgart 1992 S. 1240 - 1255.

/31/ Simon, H.A.: The Science of the Artificial, Cambstributed Problem Solving. IJCAI-79, pp. 836-841.

/35/ Stephens, L.; Merx, M.: Agent Organization as an Effector of DAI System Performance. Ninth Workshop on Distributed Artificial Intelligence, Rosario Resort, Eastsound, Washington, September 12-14, 1989, S. 263-292.

/36/ Sunita, T.: A Study on Measurement of Organizational Intelligence, in: Proceedings of CEMIT92/CECOIA 3 - International Conference on Economics, Management and Information Technologie, Tokio 1992, S. 207 - 210.

/37/ Teramoto, Y., Iwaski, N., Richter F.-J.: Inter-Organizational Learning through Strategic Alliances - Evolutionary Process of Corporate Networking, in: Proceedings of CEMIT92/ CECOIA 3 - International Conference on Economics, Management and Information Technologie, Tokio 1992, S. 239 - 242.

/38/ Tsuchiya, S.: Organizational Learning and Information Technology:Information Technology Improving Strategy Formation Process in a Loosely Coupled System, in: Proceedings of CEMIT92/CECOIA 3 - International Conference on Economics, Management and Information Technologie, Tokio 1992, S. 249 - 252.

/39/ Watanabe,Y.A.: The Role of Intra-Organization Network (ION) for Organizational Learning, in: Proceedings of CEMIT92/CECOIA 3 - International Conference on Economics, Management and Information Technologie, Tokio 1992, S. 277 - 280.

/40/ Yamamoto, B., Nakano, B, Matsuda, T.: System, Information, Organizational Intelligence and Self-dynamics, in: Proceedings of CEMIT92/CECOIA 3 - International Conference on Economics, Management and Information Technologie, Tokio 1992, S. 211 - 214.

Ein verteiltes Problemlösungssystem für die Allfinanz-Kundenberatung[1]

Hans-Jürgen König, Mark Roemer, Klaus Sandbiller, Christof Weinhardt, Andreas Will,
Fachbereich Wirtschaftswissenschaften, Universität Gießen, Licher Str. 60, W-6300 Giessen

Zusammenfassung

Gute Lösungen finanzwirtschaftlicher Probleme sind oft Kombinationen mehrerer Finanzprodukte aus unterschiedlichen Domänen. Will ein Finanzdienstleister solche Lösungen anbieten, muß er in der Regel verteilt vorliegendes Domänenwissen während des Problemlösungsprozesses logisch integrieren. In der vorliegenden Arbeit wird ein verteiltes Problemlösungssystem zur Unterstützung der Erstellung von Allfinanzangeboten vorgestellt. Zunächst werden in einem Szenario Charakteristika des zugrundeliegenden verteilten Problemlösungsprozesses ermittelt, um anschließend die darauf aufbauende Konzeption eines Blackboardsystems darzustellen.

1. Einführung

Vor dem Hintergrund eines wachsenden Wettbewerbsdrucks auf dem Markt für Finanzdienstleistungen werden sich die Anbieter zukünftig noch stärker an den Bedürfnissen ihrer Kunden orientieren müssen. Ziel einer solchen Kundenorientierung sollte es sein, innovative und ganzheitliche Problemlösungen für individuelle Kundenprobleme systematisch zu erarbeiten und anzubieten. Dieser Anspruch erfordert in der Regel die intelligente Verknüpfung geeigneter Finanzprodukte zum Beratungszeitpunkt. Hierbei muß das in Form menschlicher Experten und/oder in Systeminseln nur punktuell verfügbare finanzwirtschaftliche Domänen- und domänenübergreifende Wissen logisch integriert an der jeweiligen Kundenschnittstelle bereitgestellt werden. Dieses ist aufgrund der Komplexität ohne eine (aktive) Systemunterstützung nicht zu leisten [BuHa93]. Verteilte organisatorische Strukturen der meisten Finanzdienstleister sowie die domänenspezifische Segmentierung des Finanzdienstleistungsbereichs in Produktgruppen erfordern eine Systemumgebung mit ausgewogener Balancierung von Verteilung und Integration hinsichtlich Daten, Wissen und Problemlösungskompetenz, um zusätzlich eine notwendige Erweiterbarkeit und Wartbarkeit insbesondere im Hinblick auf die Dynamik des Anwendungsgebietes zu unterstützen. Die genannten Anforderungen legen die Realisierung eines wettbewerbsorientierten Systems zur Unterstützung der Allfinanz-Kundenberatung als verteiltes Problemlösungssystem nahe.

[1] Der Beitrag entstand im Rahmen des DFG-Schwerpunktprogramms "Verteilte Systeme in der Betriebswirtschaft". Für wertvolle Hinweise bedanken sich die Autoren bei Herrn Dr. Sahin Albayrak.

Der vorliegende Beitrag beschäftigt sich mit der Spezifikation und Konzeption eines solchen verteilten Problemlösungssystems. Zunächst wird im 2. Abschnitt ein Allfinanz-Szenario entworfen, um einen Rahmen für ein verteiltes Problemlösen zur Erstellung kundenorientierter Allfinanzangebote abzustecken und Charakteristika des verteilten Problemlösungsprozesses abzuleiten. In Abschnitt 3 wird dann die Konzeption eines Blackboardsystems vorgestellt, das den dargestellten verteilten Problemlösungsprozeß realisieren soll. Die Ausführungen verdeutlichen die Maßgeblichkeit von Konzepten und Methoden der Verteilten Künstlichen Intelligenz (VKI) zur Unterstützung betriebswirtschaftlicher Problemlösungen, so daß der Beitrag im 4. Abschnitt mit einer Würdigung der Bedeutung der VKI für den Anwendungsbereich schließt.

2. Szenario eines verteilten Problemlösungsprozesses zur Erstellung von Allfinanzangeboten

Um Charakteristika des verteilten Problemlösungsprozesses zur Erstellung von Allfinanzangeboten abzuleiten, soll zunächst ein *Allfinanz-Szenario* [SaWe92] entworfen werden. Ausgangspunkt ist die Beobachtung, daß sich ein finanzwirtschaftliches Problem allgemein als eine Folge von geforderten oder gewünschten Zahlungen im Zeitablauf darstellen läßt. So wünscht ein Kunde mit einem *Anlageproblem* eine heutige Auszahlung ("Geldanlage"), um in späteren Perioden Einzahlungen zu erhalten. Bei einem *Finanzierungsproblem* wünscht der Kunde eine heutige Einzahlung, für die er bereit ist, in zukünftigen Perioden Auszahlungen zu leisten. Häufig sind Kundenprobleme komplexer und bestehen aus einer Folge von Anlage- und Finanzierungsproblemen, so daß sie als *gemischte Probleme* bezeichnet werden können.

Deutlich wird, daß jedes finanzwirtschaftliche Problem in Abhängigkeit von zusätzlichen Restriktionen und Präferenzen ein bestimmtes, gewünschtes Zahlungsmuster induziert und deshalb als ein n-dimensionaler Vektor beschrieben werden kann, dessen n Elemente erwartete bzw. gewünschte Zahlungen zu n Zeitpunkten repräsentieren. Die Elemente können in ihrer absoluten Höhe bekannt sein oder auch Zielfunktionen darstellen, die es zu minimieren bzw. maximieren gilt. In diesem Sinne können wir einen solchen Vektor als *Problemvektor* bezeichnen und ihn als formale Repräsentation des finanzwirtschaftlichen Problems des Kunden betrachten.[2]

Beispiel:

Kunde K möchte ein Haus in einem Wert von DM 880.000 finanzieren. Zunächst benötigt er eine Einzahlung, um das Haus bezahlen zu können. Über die nächsten 25 Jahre hinweg möchte er möglichst geringe, aber jährlich konstante Zahlungen leisten. Er hat ein Finanzierungsproblem.

---

[2] Andere Präferenzen wie Risikoneigung u.ä. werden an dieser Stelle bewußt außer Acht gelassen, um die Darstellung des Szenarios möglichst einfach zu halten; zu Kooperation unter Unsicherheit vgl. [Zhan92]

Sein Problem läßt sich in Form eines Problemvektors formalisieren (der Vektor besteht aus 26 Komponenten, für heute und die nächsten 25 Jahre):

$P1 = (+\,880.000;\ -\min\{|a|\};\ \ldots;\ -\min\{|a|\})$.

25 mal

Mit dieser formalen Beschreibung des Ausgangsproblems können wir nun das *Systemziel* spezifizieren als die Generierung eines *Lösungsvektors*, der die Anforderungen des Problemvektors erfüllt, d.h. für jeden Zeitpunkt die jeweils spezifizierte Ein- oder Auszahlung liefert. Da ein Finanzprodukt durch eine bestimmte Zahlungsreihe gekennzeichnet ist, bedeutet die Suche nach einem Lösungsvektor die Suche nach solchen Finanzprodukten, die das Zahlungsmuster des Problems erfüllen. Innerhalb des Systems stehen für diese Aufgabe autonome *Fachagenten* zur Verfügung: intelligente Einheiten, die grundsätzlich in der Lage sind, finanzwirtschaftliche (Teil-)Probleme ausschließlich mit ihrem lokalen Domänenwissen ohne die Hilfe anderer Fachagenten des Systems zu lösen.

Ein Allfinanzdienstleister A bietet K einen Kredit über DM 880.000 an und fordert 25 Jahre lang Zahlungen von K in Höhe von DM 62000 nach Steuern. Dieses Angebot kann in einem Lösungsvektor L1 abgebildet werden, der das Angebot aus Kundensicht repräsentiert:

$L1 = (+\,880.000;\ -\,62.000;\ldots;\ -\,62.000)$.

25 mal

L1 als zulässige und lokal optimierte Lösung deckt sich mit dem spezifischen Muster des Problemvektors P1 von K: Der Kredit-Spezialist des Allfinanzdienstleisters A ist in der Lage, das Finanzierungsproblem autonom zu lösen.

Die finanzwirtschaftlichen Domänen spiegeln dabei verteilte finanzwirtschaftliche Produktbereiche wider, die historisch gewachsen sein können oder auch aufgrund rechtlicher Vorschriften von zwingend getrennten Geschäftseinheiten bearbeitet werden. Aufgrund der unterschiedlichen Eigenschaften von Finanzprodukten verwenden die Fachagenten unterschiedliche Problemlösungs*methoden* (z.B. Selektion von Wertpapieren, Klassifikation/Konfiguration von Leasingverträgen). Trotz dieser Unterschiede ist es aber aus Gesamtsicht entscheidend, daß den Fachagenten eine einheitliche Repräsentationsform ihrer Lösungsangebote eigen ist: Alle generieren Zahlungs-

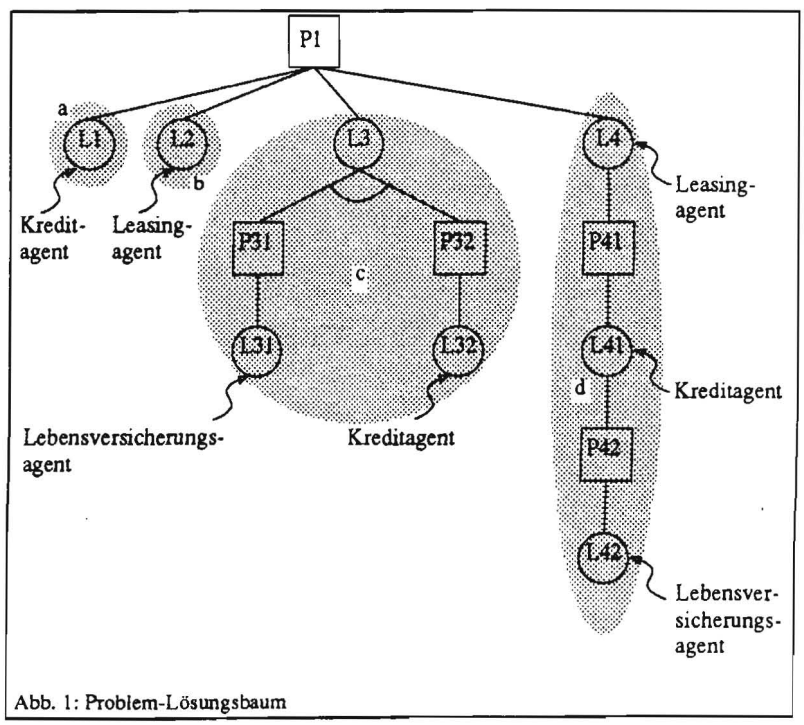


Abb. 1: Problem-Lösungsbaum

ströme, indem sie Finanzprodukte ihrer Domäne selektieren, konfigurieren und/oder kombinieren, wobei sie versuchen, den Anforderungen des Problemvektors möglichst gut zu genügen. Korrespondierend zu Anlage- bzw. Finanzierungsproblemen lassen sich zwei Klassen von Fachagenten zur Lösung von Problemen aus diesen Bereichen unterscheiden: Anlageagenten und Finanzierungsagenten. Gemischte Probleme müssen in der Regel von mindestens einem Fachagenten aus jeweils einer Klasse bearbeitet werden. Wie eingangs erwähnt, sind aber auch bei reinen Anlage- bzw. Finanzierungsproblemen oftmals kombinierte Lösungen (im Sinne domänenübergreifender Lösungen) vorteilhaft. Die Erarbeitung kombinierter Lösungen impliziert die Notwendigkeit zielgerichteter Kooperation der beteiligten Fachagenten. Mit den drei folgenden Kooperationsformen kann der verteilte Problemlösungsprozeß zur Erstellung von Allfinanzangeboten angemessen modelliert werden:

Häufig werden sich zu einem gestellten Problem mehrere Fachagenten in der Lage sehen, eine Lösung zu liefern. Wir sagen, daß Fachagenten, die an alternativen Lösungen zu *einem* Problem arbeiten, ***konkurrieren***.

Der Kredit-Agent konkurriert mit seinem Angebot L1 mit dem Leasing-Agenten (vgl. Abb. 1, a und b). Dieser liefert eine Lösung L2 in Form eines Leasingvertrags mit konstanten Nach-Steuer-Zahlungen, kombiniert mit einer extrem günstigen Kaufoption in (symbolischer) Höhe von DM 1. Dies ergibt folgenden Lösungsvektor

$$L2 = (+\,880.000;\ \underbrace{-\,60.000;\ \ldots;\ -\,60.000}_{24\ mal};\ -\,60.001).$$

(Man beachte: Der Leasingvertrag gewährleistet die Nutzung des Hauses für 25 Jahre und die sichere Möglichkeit, danach das Haus zu DM 1 zu kaufen. In diesem Sinne repräsentiert das erste Element des Vektors "+880.000" keine echte Zahlung; es beschreibt vielmehr formal die Tatsache, daß der Leasingvertrag die eigentliche Bezahlung ersetzt. K muß diesen Betrag nicht aufnehmen (im Gegensatz zum Kreditkauf L1)).

Für manche Probleme können vorteilhafte Lösungen z.B. aufgrund von theoretisch erarbeitetem oder Erfahrungswissen (domänenübergreifendes "Kombinationswissen") a priori - also zu Beginn der Lösungssuche - bekannt sein. Es sollte möglich sein, daß solche Lösungen erkannt werden und die einzubeziehenden Fachagenten zur Leistung ihrer Lösungsbeiträge angestoßen werden. Wir nennen diese Form der Kooperation ***explizite Kollaboration***, weil die Fachagenten ausdrücklich aufgefordert sind, Beiträge zu einer als gut erkannten, domänenübergreifenden Lösung zu leisten.

Angenommen, folgende Kombination wäre in unserem Beipiel eine gute Lösung: Für einen endfälligen Kredit mit konstanten Zinszahlungen (d.h. Tilgung erfolgt in voller Höhe am Ende der Laufzeit in t=25) ist die Tilgungszahlung über DM 880.000 durch eine Lebensversicherung gedeckt. Um diesen Sachverhalt abbilden zu können, muß P1 in zwei Teilprobleme P31 und P32 zerlegt werden:

$$P31 = (+\,880.000;\ \underbrace{-\,\min\{|b|\};\ \ldots;\ -\,\min\{|b|\}}_{24\ mal};\ -\,\min\{|b|\} - 880.000)$$ *für den Kredit-Agenten,*

$$P32 = (0;\ \underbrace{-\,\min\{|c|\};\ \ldots;\ -\,\min\{|c|\}}_{24\ mal};\ -\,\min\{|c|\} + 880.000)$$ *für den Lebensversicherungs-Agenten.*

Der Kredit-Agent bietet an: $L31 = (+\ 880.000; -\ 44.000; \ldots; -\ 44.000; -\ 44.000 - 880.000)$ *und* (24 mal)

der Lebensversicherungs-Agent: $L32 = (0; -\ 14.000; \ldots; -\ 14.000; -\ 14.000 + 880.000)$. (24 mal)

Die Summe L3 = L31 + L32 ergibt: $L3 = (+\ 880.000; -\ 58.000; \ldots; -\ 58.000)$ *(vgl. Abb. 1, c).* (25 mal)

In vielen Fällen wird aber das zuvor beschriebene Kombinationswissen nicht zur Verfügung stehen. Dies gilt um so mehr, je komplizierter die Problemstellung und je differenzierter die Zielvorstellungen des Kunden sind. Wir verlangen, daß dann die einzelnen Fachagenten das Recht haben, aus ihrer lokalen Sicht gute Lösungen vorzuschlagen, auch wenn diese das gestellte Problem nicht vollständig lösen. Wird das Problem von einem Fachagenten nicht vollständig gelöst, ergibt sich aus dem Vergleich der unvollständigen Lösung mit dem gestellten Problem ein Restproblem, welches dann auf die gleiche Weise wie das ursprüngliche bearbeitet werden kann. Wir sagen, daß ein Fachagent, der an der Lösung eines Restproblems arbeitet, mit dem Fachagent, der das ursprüngliche Problem nur unvollständig gelöst hat, ***implizit kollaboriert.***

Neben seiner Lösung L2 generiert der Leasing-Agent aufgrund einiger Entscheidungskriterien, die er von K abgefragt hat, noch ein zweites Angebot mit einer Einmalzahlung von DM 800.000 zu Beginn des Vertrags und einer Kaufoption zu DM 185.000 25 Jahre später. Wir erhalten

$L4 = (+\ 880.000 - 800.000; 0; \ldots; 0; -\ 185.000)$. (24 mal)

Mit diesem Vertrag ist das Problem P1 noch nicht gelöst; es verbleibt das Restproblem

$P41 = P1 - L4 = (+\ 800.000; -\ min\{|a|\}\ ; \ldots ; -\ min\{|a|\}\ ; -min\{|a|\} + 185.000)$. (24 mal)

Zu P41 findet der Kredit-Agent

$L41 = (+\ 800.000; -\ 40.000; \ldots; -\ 40.000; -\ 40.000 - 800.000)$. (24 mal)

Auch jetzt bleibt ein Restproblem P42 = P41 - L41:

$P42 = (+/-\ 0\ ; -\ min\{|a| + 40.000\}; \ldots ;-\ min\{|a| + 40.000\}\ ; -\ min\{|a| + 40.000\} + 985.000)$. (24 mal)

Der Lebensversicherungs-Agent kann die Deckung der hohen Schlußzahlung in t=25 übernehmen:

$L42 = (0; -\ 15.000; \ldots; -\ 15.000; -\ 15.000 + 985.000)$. (24 mal)

L4 + L41 + L42 ist die Kombinationslösung für P1 aus der impliziten Kollaboration des Kredit-, Leasing und Lebensversicherungs-Agenten (s. Abb. 1, d):

$L4 + L41 + L42 = (+\ 880.000; -\ 55.000; \ldots; -\ 55.000)$ (25 mal)

Dies ist gleichzeitig aus globaler Sicht die beste der vier Lösungen, die die konkurrierenden Agenten bzw. Koalitionen von Agenten erstellt haben (s. Abb. 1, a vs. b vs. c vs. d).

In der Regel wird der skizzierte verteilte Problemlösungsprozeß durch die Synthetisierung korrespondierender Teillösungen zu entsprechenden Kombinationslösungen (Allfinanzangeboten) abgeschlossen. Wurden mehrere alternative Lösungen zum Ausgangsproblem erarbeitet, so schließt sich die Generierung einer Rangordnung an, die dem Kunden präsentiert wird.

3. Konzeption eines Blackboardsystems zur Erstellung von Allfinanzangeboten

Realisiert wird der im vorangegangenen Szenario dargestellte verteilte Problemlösungsprozeß zur Erstellung kundenorientierter Allfinanzangebote durch ein blackboardbasiertes System [Nii86], dessen Architektur anhand von Abb. 2 erläutert wird.

Im Blackboard werden sowohl der während des verteilten Problemlösens entstehende Problem-Lösungsbaum als auch die für eine Konsultation spezifischen und benötigten Kundendaten gespeichert. Hierzu wird das Blackboard in die drei Ebenen *Konsultationsebene (KE)*, *Problemebene (PE)* und *Lösungsebene (LE)* gegliedert. In der KE werden die Kundendaten einer Konsultation abgelegt, welche als Kalkulationsgrundlage zur Selektion, Konfiguration und/oder Kombination von Finanzprodukten notwendig sind (z. B. Kalkulationszinssätze), sowie das vom Kunden spezifizierte Bewertungskriterium (z. B. Barwert der Nach-Steuer-Zahlungen). Die Daten der KE bleiben während einer Konsultation unverändert. In der PE werden sowohl das Kundenproblem als initiales Ausgangsproblem des Problemlösungsprozesses als auch die während des Problemlösens durch implizite oder explizite Kollaboration entstehenden Rest- bzw. Teilprobleme durch jeweils einen Problemvektor *einheitlich* repräsentiert. Aufgrund der einheitlichen Problemrepräsentation ist für auf das Blackboard zugreifende Agenten transparent, ob sie ein Ausgangsproblem oder ein Rest- bzw. Teilproblem bearbeiten. Jedes in der PE existierende Problem wird neben dem Problemvektor durch weitere Daten näher

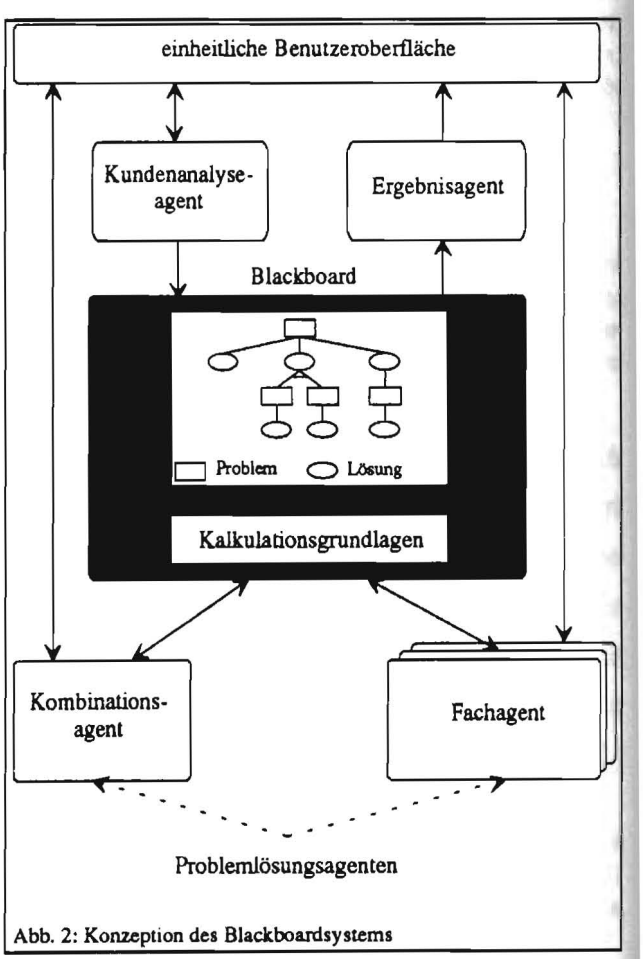


Abb. 2: Konzeption des Blackboardsystems

beschrieben, zum Beispiel durch einen eindeutigen Identifikator. In der LE werden von den Fachagenten angebotene Finanzprodukte durch einen Lösungsvektor einheitlich repräsentiert. Jeder Lösungsvektor stellt jeweils die Lösung eines Problems der PE dar. Die Darstellung eines Finanzangebots umfaßt neben dem Lösungsvektor ebenfalls weitere spezifizierende Informationen, insbesondere einen Verweis auf die mit dem Angebot verbundene konkrete Vertragsspezifikation.

Über das Blackboard, das auf einer relationalen Datenbank basiert, können mehrere Kundenkonsultationen nebenläufig bearbeitet werden. Die drei Ebenen sind jeweils in Form mehrerer relationaler Tabellen (3 NF) modelliert. Die alternierende Verknüpfung von PE und LE erfolgt über referentielle Integrität.

Über das Blackboard interagieren vier Typen von Agenten: ein Kundenanalyseagent, ein Ergebnisagent, ein Kombinationsagent und mehrere finanzwirtschaftliche Fachagenten. Alle Agenten können mit dem Benutzer über eine einheitliche Benutzeroberfläche kommunizieren; die logische und physische Verteilung des Problemlösungsprozesses ist somit für den Benutzer transparent.

Der *Kundenanalyseagent* ermittelt die Zielsetzung sowie die konsultationsrelevanten Daten des Kunden und trägt sie in die KE ein. Weiterhin präzisiert der Kundenanalyseagent das Kundenproblem, formalisiert es durch den Problemvektor und trägt es als Ausgangsproblem in die PE ein; hierbei ist unsicheres und unscharfes Wissen zu verarbeiten. Der Kundenanalyseagent hat über datenbank- und rechnerübergreifende virtuelle (Fachdomänen-) Schemata Zugriff [Köni93] auf alle in den Server-Datenbanken des Finanzdienstleisters dezentral gespeicherten Kundendaten. Hierdurch sind sowohl die Daten vergangener Konsultationen als auch die operativen Daten des Finanzdienstleisters unternehmungsweit verfügbar.

Nachdem der Kundenanalyseagent das Kundenproblem formalisiert in der PE des Blackboards eingetragen hat, können sowohl der Kombinationsagent als auch die Fachagenten auf dieses Problem zugreifen, wodurch der verteilte Problemlösungsprozeß initiiert wird. Kombinationsagent und Fachagenten werden als *Problemlösungsagenten* bezeichnet, da diese Agenten konkurrierend zueinander in der PE repräsentierte finanzwirtschaftliche Probleme bearbeiten. Sie beinhalten einen Konditionen- und einen Aktionen-Teil. Im *Konditionen-Teil* sind die Voraussetzungen spezifiziert, die erfüllt sein müssen, damit ein Problemlösungsagent ein Problem der PE bearbeiten kann. Durch die Evaluierung ihres Konditionen-Teils können Problemlösungsagenten autonom [DuLe89, 66f] und reflektierend entscheiden, ob sie aufgrund ihrer Problemlösungskompetenz ein in der PE eingetragenes Problem bearbeiten und damit zum Gesamtlösungsprozeß beitragen können;

Problemlösungsagenten sind also selbstselektierend. Die Problembearbeitung selbst erfolgt im Aktionen-Teil eines Problemlösungsagenten. Aktionen-Teil und Konditionen-Teil können unabhängig voneinander ausgeführt werden.

Fachagenten erarbeiten durch Auswertung ihres jeweiligen Aktionen-Teils Vertragsangebote für Finanzprodukte als Lösungen von Problemen der PE und tragen den zugehörigen Lösungsvektor in die LE ein. Ein Fachagent kann dabei aufgrund seiner Kompetenz selbständig entscheiden, ein Problem vollständig oder nicht vollständig zu lösen. Bei unvollständiger Problemlösung trägt er den Lösungsvektor für das unvollständig gelöste Problem in der LE ein und bestimmt zudem das verbleibende Restproblem, welches er in der PE einträgt. Durch das Ausschreiben des Restproblems wird die implizite Kollaboration angestoßen.

Der *Kombinationsagent* verfügt über globales domänenübergreifendes Kombinationswissen und kann eine explizite Kollaboration von Fachagenten initiieren: Er kennt für wesentliche finanzwirtschaftliche Probleme vorteilhafte Kombinationen von Finanzdienstleistungen und kann diese Probleme explizit in im allgemeinen weniger komplexe Teilprobleme zerlegen. Diese werden vom Kombinationsagenten in der PE ausgeschrieben und anschließend von Problemlösungsagenten gelöst. Die dargestellten Interaktionen von Problemlösungsagenten über das Blackboard ermöglichen ein inkrementelles und kooperatives Problemlösen und erlauben sowohl die Auswertung von vorhandenem Kombinationswissen als auch die Erstellung innovativer Finanzproduktkombinationen. Fehlende Detailinformationen können die Problemlösungsagenten direkt über die einheitliche Benutzeroberfläche vom Kunden erfragen.

Der *Ergebnisagent* aggregiert einander ergänzende Finanzprodukte zu Allfinanzangeboten, bewertet diese anhand des in der KE spezifizierten Bewertungskriteriums und bietet die besten Allfinanzangebote dem Kunden an. Realisiert werden die Agenten mit einer hybriden, Objektorientierung und Regelbasierung umfassenden Entwicklungsumgebung (Trinzic ADS 6.11).

Nach der Darstellung von Blackboardkonzeption und Agenten sowie deren Interaktionen soll anhand von Abb. 3 die Konzeption der Steuerung des verteilten Problemlösens vorgestellt werden. Eintragungen von finanzwirtschaftlichen Problemen in der PE sind als Blackboardereignisse (Events) definiert, welche die Problemlösungsagenten triggern. Zur Verwaltung dieser Events verfügt das Blackboard über eine aktive Komponente, den *Event-Manager*. Nach der Eintragung eines Problems in der PE übermittelt der eintragende Agent Informationen über den Problemeintrag (u. a. Identifikation und Typ des eingetragenen Problems, wie z.B. Anlageproblem) an den Event-Manager. Dieser verfügt über Wissen,

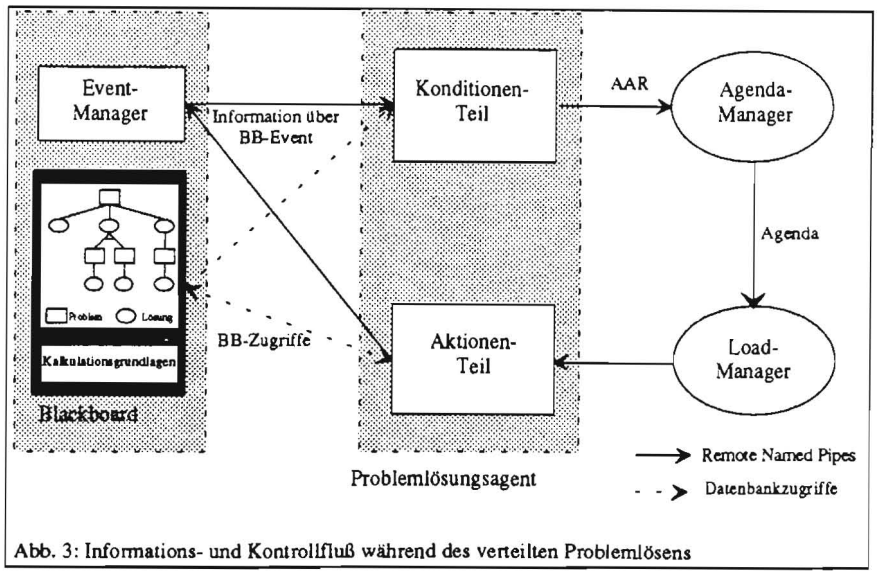


Abb. 3: Informations- und Kontrollfluß während des verteilten Problemlösens

welche Problemlösungsagenten für die Bearbeitung des neu eingetragenen Problems grundsätzlich geeignet sind, und informiert deren Konditionen-Teile. Durch die zentrale Dienstleistungsfunktion des Event-Managers werden die Problemlösungsagenten davon entlastet, die PE ständig zu überwachen und zu prüfen, ob ein neues Problem eingetragen wurde. Darüber hinaus brauchen sich eintragende Agenten nicht darum zu kümmern, welche Problemlösungsagenten über einen Event zu informieren sind.

Evaluiert der getriggerte Konditionen-Teil eines Agenten, daß der Agent eine finanzwirtschaftliche Problemstellung erfolgreich bearbeiten kann, generiert er einen Agent Application Record (AAR), als Bewerbung des Agenten zur Lösung der Problemstellung. Der AAR wird an den Agenda-Manager übermittelt. Der *Agenda-Manager* dient der Koordination des Problemlösungsprozesses. Die Koordination erfolgt über eine Priorisierung der eingehenden AARs, die zu einer Agenda der Konsultation zusammengefaßt werden. So wird festgelegt, welches Problem von welchem Agenten vorrangig bearbeitet werden soll. Diese Steuerungsmöglichkeit erlaubt insbesondere die Priorisierung des Kombinationsagenten gegenüber Fachagenten, die sich konkurrierend um die Lösung desselben Problems bewerben. Weiterhin kann der Agenda-Manager über die Terminierung einzelner Äste des Problem-Lösungsbaums entscheiden sowie über die Terminierung des gesamten Problemlösungsprozesses und die Aktivierung des Ergebnisagenten.

Die vom Agenda-Manager erstellte Agenda wird vom *Load-Manager* gelesen. Dieser übermittelt die einzelnen Aufgaben der Agenda an die Aktionen-Teile der entsprechenden Problemlösungsagenten. Die Aktionen-Teile werden durch den Load-Manager aktiviert, bearbeiten die zugeordnete Problemstellung, tragen die erstellten Finanzprodukte in die LE und eventuell entstehende Teil- oder Restprobleme in die PE des Blackboards ein.

Agenten und Manager werden nicht nur logisch verteilt auf einem Rechner operieren, sondern auch physisch verteilt in einem LAN. Es ist deshalb geplant, die Funktionalität des Load-

Managers zu erweitern: Er soll abhängig von der Lastverteilung in einem LAN[3] entscheiden, auf welcher Maschine im LAN die Aktionen-Teile von Problemlösungsagenten auszuführen sind; er führt somit eine Lastbalancierung im LAN durch. Event-Manager, Agenda-Manager und Load-Manager sollen konventionell mit C realisiert werden. Die Interprozeßkommunikation zwischen Agenten und Managern erfolgt über Remote Named Pipes und basiert auf einem schichtenorientierten Programmiermodell [Mack93].

4. Bewertung und Ausblick

Finanzwirtschaftliche Problemstellungen zeichnen sich durch ihre hohe Komplexität und schlechte Strukturiertheit aus, die vor allem aus der Breite und Tiefe einer umfangreichen Produktpalette sowie der situativen und kundenspezifischen Individualität der Problemstellung resultieren. Daraus folgt, daß zur Lösung dieser Problemstellungen keine vorgegebenen "Lösungspfade" existieren [MaRo93]. Eine befriedigende kundenindividuelle Lösung erfordert die flexible Verarbeitung großer Mengen von domänenspezifischem und domänenübergreifendem finanzwirtschaftlichem Wissen sowie die Auswertung einer Vielzahl von Daten, z. B. über Kunden und Finanzprodukte. Voraussetzung hierfür ist die Verfügbarkeit von Problemlösungswissen und problemadäquaten Lösungsmethoden. Wissen und Fakten sind aber (nicht nur bei geographisch verteilt operierenden Unternehmungen) nur punktuell bei verschiedenen menschlichen Experten oder in Systeminseln verfügbar.

Für eine technisch überzeugende und betriebswirtschaftlich fundierte Unterstützung von Allfinanz-Angebotsprozessen ist deshalb eine logisch integrierte und flexible Verarbeitung des verteilt vorliegenden Sach- und Problemlösungswissens notwendig. Hierfür geeignet sind die in der VKI entwickelten Ansätze des verteilten Problemlösens und der Multi-Agenten-Systeme. Wie anhand der vorgestellten Architektur eines Blackboardsystems gezeigt wurde, kann das Wissen entsprechend den hier in einem Szenario abgeleiteten Anforderungen an den Problemlösungsprozeß partitioniert und modular ausgewertet werden: eine wesentliche Voraussetzung für die Erweiterbarkeit, Wartbarkeit und Robustheit komplexer Systeme [KiSc91, 190]. Problemlösungsagenten können mit ausreichenden lokalen Fähigkeiten ausgestattet werden, um eine komplexe Problemstellung explizit oder implizit in einfachere Probleme zu zerlegen und diese weitgehend autonom zu lösen. Indem die Agenten während des Problemlösens interagieren, können auch solche Problemstellungen gelöst werden, welche die Kompetenz jedes einzelnen Agenten übersteigen. Damit kann die Problemlösungskompetenz eines Finanzdienstleisters integriert, aber dezentral an der

---

[3] Als Entwicklungs- und Einsatzplattform dient ein Ethernet mit PS/2 Maschinen unter OS/2 2.0, OS/2 LAN Services und OS/2 Database Server.

Kundenschnittstelle bereitgestellt werden, um innovative Allfinanzdienstleistungen jedem Kunden individuell anzubieten.

Es erscheint uns somit möglich, mit Hilfe von Konzepten und Methoden der VKI die betriebswirtschaftlich fundierte Lösung eines aktuellen Problems vieler Finanzdienstleister informationstechnisch überzeugend zu unterstützen. Mit einer zunächst prototypischen Implementation und dem Test des Systems soll diese These auch einer ersten empirischen Überprüfung unterzogen werden.

Beispielhaft steht die Erstellung von Allfinanzangeboten für eine Reihe betriebswirtschaftlicher Probleme, zu deren Lösung verteilt vorliegendes Wissen in einem zeitkritischen Prozeß zu verknüpfen ist. So rückt auch in anderen Bereichen im Zuge einer zunehmenden Wettbewerbs- und Kundenorientierung eine Individualisierung des Produktangebots in den Blickpunkt des Interesses [ShSt88]. Die daraus erwachsenden Anforderungen an eine Systemunterstützung sowohl in der Kundenberatung als auch beispielsweise im produktionswirtschaftlichen Bereich [Alba92] lassen erwarten, daß die Bedeutung der VKI zur Lösung betriebswirtschaftlicher Problemstellungen weiter wachsen wird.

5. Literaturverzeichnis

[Alba92] *Albayrak, S.*: TUBKOM-Projekt: Blackboard-DEC - Verteilte kooperierende wissensbasierte Systeme zu der Fertigungssteuerung. In: Künstliche Intelligenz 6 (1992) 1, S. 64 - 68.

[BuHa93] *Buhl, H.U.; Hasenkamp, U.; Müller-Wünsch, M.; Roßbach, P.; Sandbiller, K.*: WI - State of the Art: IT-Unterstützung in der Finanzberatung. Erscheint in: Wirtschaftsinformatik (1993) 3.

[DuLe89] *Durfee, E. H.; Lesser, V. R.; Corkill, D.D.*: Trends in Cooperative Distributed Problem Solving. In: IEEE Transactions on Knowledge and Data Engineering 1 (1989) 1, S. 63 - 83.

[KiSc91] *Kirn, S.; Scherer, A.; Schlageter, G.*: The FRESCO Agent Model: Cooperative Behavior in Federative Environments. In: Proceedings of IJCAI-91 Workshop. Sydney 1991.

[Köni93] *König, H.-J.*: Dezentrale Datenhaltung in der Allfinanzkundenberatung. In Informationstechnik it-ti 1 (1993), S. 45 -54

[Mack93] *Mack, B.*: Synchronisation und Kommunikation in nebenläufigen, verteilten Anwendungssystemen unter dem Betriebssystem OS/2 2.0 - Grundlage und Entwicklung eines Programmiermodelles. Dokumentation 8/1993, Professur für BWL-Wirtschaftsinformatik, Universität Gießen 1993.

[MaRo93] *Mack, B.; Roemer, M.; Sandbiller, K.; Will, A.*: Problemlösungsmodelle der verteilten künstlichen Intelligenz für die Allfinanz-Kundenberatung. Disc.Paper WI 43/1993, Professur für BWL-Wirtschaftsinformatik, Universität Gießen 1993.

[Nii86] *Nii, H.P.*: Blackboard Systems: The Blackbaord Modell of Problem Solving and the Evolution of Blackboard Architectures. In: AI Magazine, S. 37 - 53, Summer 1986.

[SaWe92] *Sandbiller, K.; Weinhardt, C.; Will, A.*: Cooperating Agents Solving Financial Problems - A Scenario. Beitrag für das 5. Arbeitstreffen der GI-Fachgruppe "Verteilte Künstliche Intelligenz", Erlangen 17./18. 12. 1992.

[ShSt88] *Shaw, R., Stone, M.*: Database-Marketing, Aldershot 1988.

[Zhang92] *Zhang, C.*: Cooperation under Uncertainty in Distributed Expert Systems. In: Artificial Intelligence 56 (1992) S. 21 -69.

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 published 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-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
Zhaohui 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-92-40
Philipp Hanschke, Knut Hinkelmann: Combining Terminological and Rule-based Reasoning for Abstraction Processes
17 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-92-47
Frank Bomarius: A Multi-Agent Approach towards Modeling Urban Traffic Scenarios
24 pages

RR-92-48
Bernhard Nebel, Jana Koehler:
Plan Modifications versus Plan Generation: A Complexity-Theoretic Perspective
15 pages

RR-92-49
Christoph Klauck, Ralf Legleitner, Ansgar Bernardi:
Heuristic Classification for Automated CAPP
15 pages

RR-92-50
Stephan Busemann:
Generierung natürlicher Sprache
61 Seiten

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

RR-92-52
Mathias Bauer, Susanne Biundo, Dietmar Dengler, Jana Koehler, Gabriele Paul: PHI - A Logic-Based Tool for Intelligent Help Systems
14 pages

RR-92-54
Harold Boley: A Direkt Semantic Characterization of RELFUN
30 pages

RR-92-55
John Nerbonne, Joachim Laubsch, Abdel Kader Diagne, Stephan Oepen: Natural Language Semantics and Compiler Technology
17 pages

RR-92-56
Armin Laux: Integrating a Modal Logic of Knowledge into Terminological Logics
34 pages

RR-92-58
Franz Baader, Bernhard Hollunder:
How to Prefer More Specific Defaults in Terminological Default Logic
31 pages

RR-92-59
Karl Schlechta and David Makinson: On Principles and Problems of Defeasible Inheritance
13 pages

RR-92-60
Karl Schlechta: Defaults, Preorder Semantics and Circumscription
19 pages

RR-93-02
Wolfgang Wahlster, Elisabeth André, Wolfgang Finkler, Hans-Jürgen Profitlich, Thomas Rist: Plan-based Integration of Natural Language and Graphics Generation
50 pages

RR-93-03
Franz Baader, Berhard Hollunder, Bernhard Nebel, Hans-Jürgen Profitlich, Enrico Franconi: An Empirical Analysis of Optimization Techniques for Terminological Representation Systems
28 pages

RR-93-04
Christoph Klauck, Johannes Schwagereit: GGD: Graph Grammar Developer for features in CAD/CAM
13 pages

RR-93-05
Franz Baader, Klaus Schulz: Combination Techniques and Decision Problems for Disunification
29 pages

RR-93-08
Harold Boley, Philipp Hanschke, Knut Hinkelmann, Manfred Meyer: COLAB: A Hybrid Knowledge Representation and Compilation Laboratory
64 pages

RR-93-09
Philipp Hanschke, Jörg Würtz: Satisfiability of the Smallest Binary Program
8 Seiten

RR-93-11
Bernhard Nebel, Hans-Juergen Buerckert: Reasoning about Temporal Relations: A Maximal Tractable Subclass of Allen's Interval Algebra
28 pages

RR-93-12
Pierre Sablayrolles: A Two-Level Semantics for French Expressions of Motion
51 pages

RR-93-13
Franz Baader, Karl Schlechta: A Semantics for Open Normal Defaults via a Modified Preferential Approach
25 pages

DFKI Technical Memos

TM-91-12
Klaus Becker, Christoph Klauck, Johannes Schwagereit: FEAT-PATR: Eine Erweiterung des D-PATR zur Feature-Erkennung in CAD/CAM
33 Seiten

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 Busemann: 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
27 pages

DFKI Documents

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-92-22
Werner Stein: Indexing Principles for Relational Languages Applied to PROLOG Code Generation
80 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

D-92-28
Klaus-Peter Gores, Rainer Bleisinger: Ein Modell zur Repräsentation von Nachrichtentypen
56 Seiten

D-93-01
Philipp Hanschke, Thom Frühwirth: Terminological Reasoning with Constraint Handling Rules
12 pages

D-93-02
Gabriele Schmidt, Frank Peters, Gernod Laufkötter: User Manual of COKAM+
23 pages

D-93-03
Stephan Busemann, Karin Harbusch(Eds.): DFKI Workshop on Natural Language Systems: Reusability and Modularity - Proceedings
74 pages

D-93-04
DFKI Wissenschaftlich-Technischer Jahresbericht 1992
194 Seiten

D-93-06
Jürgen Müller (Hrsg.):
Beiträge zum Gründungsworkshop der Fachgruppe Verteilte Künstliche Intelligenz Saarbrücken 29.-30. April 1993
235 Seiten

Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

Beiträge zum Gründungsworkshop der Fachgruppe Verteilte Künstliche Intelligenz Saarbrücken 29.-30. April 1993

Jürgen Müller (Hrsg.)

D-93-06
Document