Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Document
D-94-12

Proceedings des Studentenprogramms der 18. Deutschen Jahrestagung für Künstliche Intelligenz KI-94

Arthur Sehn, Serge Autexier (Hrsg.)

September 1994

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 20 80
67608 Kaiserslautern, FRG
Tel.: (+49 631) 205-3211/13
Fax: (+49 631) 205-3210

Stuhlsatzenhausweg 3
66123 Saarbrücken, 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, and Siemens. 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.

Dr. Dr. D. Ruland
Director

Proceedings des Studentenprogramms der 18. Deutschen Jahrestagung für Künstliche Intelligenz KI-94

Arthur Sehn, Serge Autexier (Hrsg.)

DFKI-D-94-12

Proceedings des Studentenprogramms der 18. Deutschen Jahrestagung für Künstliche Intelligenz

KI–94

herausgegeben von

Arthur Sehn
Fachbereich Informatik
Universität des Saarlandes
Postfach 151150
66041 Saarbrücken
acsehn@cs.uni–sb.de

Serge Autexier
DFKI
Stuhlsatzenhausweg 3
66123 Saarbrücken
serge@dfki.uni–sb.de

21. – 22. September, 1994

Vorwort

Das Gebiet der Künstlichen Intelligenz (KI) hat in den letzten Jahrzehnten einen großen Aufschwung erfahren. Das ursprünglich kleine Gebiet der KI hat sich in viele Teilgebiete aufgefächert, wie zum Beispiel:

- Computerlinguistik,
- Robotik,
- Computersehen,
- Wissensrepräsentation,
- Automatisches Beweisen,
- Planen,
- Expertensysteme.

Wegen der Vielfalt der Teilgebiete und deren Größe, ist es auch für den guten wissenschaftlichen Nachwuchs schwierig, sich zu artikulieren und einen geeigneten Rahmen für die Präsentation eigener Arbeiten auf dem Gebiete der KI zu finden. Deshalb findet im Rahmen der KI-94 am Mittwoch, den 21. September 1994 und Donnerstag, den 22. September 1994 erstmalig ein Studentenprogramm statt. Dieses ist bunt gefächert, und zeigt, wie die KI-94 insgesamt, das breite Spektrum der KI. Es werden einige Beiträge zu den vorher genannten Teilgebieten der KI vorgestellt.

Im ersten Teil des Studentenprogramms, am Mittwoch, den 21. September 1994, präsentieren fünf Studenten ihre aktuellen Arbeiten auf ihrem jeweiligen Teilgebiet der KI. Zur Eröffnung stellt Smailbegovic Fethula in seinem Vortrag "Physical information - the way to intelligent machine?" einige Gedanken über den Zusammenhang zwischen Information und Intelligenz vor. Es folgt ein Vortrag mit dem Thema "Resolving conflicts in distributed legal reasoning" von Stefanie Brüninghaus über ein Expertensystem mit Anwendung in der Rechtswissenschaft. Aus dem Bereich der Computerlinguistik hält Sascha Brawer einen Vortrag über "Mechanismen einer kontextfreien Grammatik für das Deutsche". Danach schließt Katerina Marinagi einen Beitrag über "TimePlan: Towards a time-based domain-independent planning system", einem zeitbasierten Planungssystem, an. Abgerundet wird dieser kurze Ausflug in die unterschiedlichen Anwendungsbereiche der KI durch Guido Lindner mit dem Thema "logikbasiertes Lernen in relationalen Datenbanken", hierbei wird die Anwendung eines logikorientierten Lernverfahrens auf relationale Datenbanken aufgezeigt.

Arthur Sehn und Serge Autexier

to react. Why spectrum? Intention is, with this to term to try "isolate" as much as possible smaller part of entrance information; and after that to explain it and determinate connection of entrance information wand system's respond on that (intelligent behaviour).

Knowing that system con conclude about entrance information that substance consist of informations, and knowing changeable nature of entrance information, only conclusion which strike me is that there is subjectivity of physical information. Does information builds substance we know? Yes, by this conclusions.

Abilities of physical information we don't know yet, but considering the great influence it may have on us, I would call it alpha (basic) key (it could open new horizonts), αk. That means that αk builds concepts we see, like substance, energy, elementary particles, force ...

If we implement algorithm approach instead of mathematical one, then Edvard Fredkin as only conclusion, is information as an elementary part of our world. For most of the scintists today, information is only one kind of energy, but by Fredkin information is more basical than an energy and substace.

αk builds connections and make structures of αk-s, what represents physical informations which have some meaning for some systems. We don't know yet features and activities of αk, but they are primery target of future researches, and especially its mathematical model. If we prove existence of αk we could, on the basis of entrance and outgoing informations, to predict functional connection between these signals. The prove that something exists here are Feinaman's diagramms of interactions between two electrons. They were not in any kind of contact but they reacted!

Can we prove existence of αk? It is very hard to find answer on this question. With todays methods of physics, with accelerators, is very hard to do it. Why? Because it is impossible to isolate αk alone, while there are reactins with equipment which gives us result. But theoretical way, with its proves in experiments, can give us an answer of existence of αk. It's not appropriate moment, at this stage, to speak about precise mathematical formulas. I think, we should accept method of global view by Pascal. The basic for the further work would give a proper computer support, which would consist of graphiv simulation of some parts of modell and executing some algorithms, for example; algorithm which would, on the basis of today knowledge about elementary particles and laws of physics, try to predict what is the next elementary particle in array of all known elementary particles. Purpose of this algorithm is to go deep into structure of our substance, and to try to find and follow processes, like "conversion", structure forming, influence and dealing with outside changes. Todays computers give us good basics for this. Artificial Intelligence, as a science, could bring great changes to our civilization, and bring us at the edge of new and better era. In order to reach intelligent machine, we have to solve great number of problems on our way to it. Problem of intelligent machine, of the whole intelligence, is dynamical, and we have to try to look "wider" the problem, to understand suuroundings and

Resolving Conflicts in Distributed Legal Reasoning

A Preliminary Report

Stefanie Brüninghaus
Lehrstuhl für Praktische Informatik I
Universität Mannheim
D - 68131 Mannheim, Germany
Phone: + 49 (621) 292 - 1196
Fax: + 49 (621) 292 - 5297
steffi@pi1.informatik.uni-mannheim.de

Intelligent Systems Program
University of Pittsburgh
Pittsburgh, PA 15260, USA
steffi@isp.pitt.edu

Abstract

This paper introduces DANIEL, *a* **D**istributed **A**rchitecture for the k**N**owledge-based **I**ntegration of **E**xpert Systems in **L**aw. *In this architecture, the legal sources are mapped to separate problem solvers which follow the appropriate paradigm – case-based or rule-based and are coupled via a blackboard. The integration of the problem solvers cannot be achieved by simply interleaving the respective reasoning chains or even "merging" their solutions without further analysis, since their knowledge widely overlaps and in addition often allows contradictory interpretations. In the proposed approach, the problem solvers work autonomously and concurrently, each of them derives an independent local solution. These are then assessed by an explicit coordination component, which on th eone hand applies legal meta-knowledge to handle conflicting interpretations. Additionally, a novel redundancy concept is introduced, which allows to explicitly represent the interrelationships between the reasoners and to handle conflicts efficiently.*

From an AI perspective, the legal sources can be characterized as follows:

- The statutes consist of a body of abstract rules intended to be applicable to a indefinite number of concrete cases of a certain type. They are therefore phrased in an abstract way and need to be interpreted when being applied to a concrete situation.
- Case law are the decisions of courts in previous cases. Their binding force is restricted to the dispute for which they were decided. However, since a judge will presumably decide the same way in a similar situation (defining this similarity is a non-trivial problem [2]), in particular the decisions of higher courts have a strong prejudicial effect. This phenomenon has been discussed controversely in jurisprudence; nevertheless it undeniably strongly influences legal interpretation.
- In particular in tax law, the administrative (resp. revenue) regulations play an important role in everyday legal practice. They embody an official interpretation of the statutes, mainly applicable in *standard situations*. The revenue regulations are enacted by the government, and not by the legislator, thus they do only bind the revenue offices and not the taxpayers.

Apparently, the legal sources require different representation paradigms, have distinct binding forces, and make use of dissimilar inference mechanisms. Therefore, each legal source has to be represented separately, applying the most suitable paradigm, as comprehensively discussed in [18].

From the characteristics of the legal sources, the proper representation can be easily derived: Case law is best represented in a case base and processed by a case-based reasoner, while the statutes are naturally mapped on a rule base. The administrative regulations are usually phrased in the form of rather general rules and therefore should be represented by (possibly annotated) rules.

2.2 Integration of the Problem Solvers

In order to bring together the problem solvers, they are coupled via a blackboard, which offers several advantages:

- Data can be smoothly exchanged via a blackboard, no particular exchange mechanisms have to be developed.
- Often, the statutes have to be applied recursively, which can be efficiently mapped on a hierarchical blackboard structure.

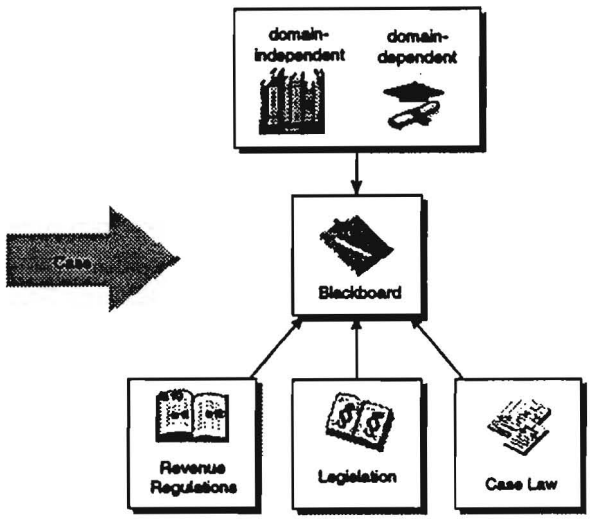

Figure 1: Overview of the DANIEL architecture

In order to illustrate the architecture, a brief, simplified example is given here: From the German Income Tax Law, it can be derived that built-in laundry facilities (i.e., not easily removable, since attached to the concrete) belong to a building, and that therefore the respective expenses must be manufacturing cost. The term manufacturing cost is not exactly defined in all details and therefore *open-textured*. In 1970, however, the German Supreme Tax Court decided that a washing machine screwed to the concrete is not included in the manufacturing cost of the building. Since the definition in the statutes is rather open-textured, while the cited case is an exact match, case law has to be applied.

3 Disacussion of the Control Strategy

There are numerous options for integrating case-based and rule-based co-reasoners [20]. This far the general idea has been that all reasoners contribute to a *common, overall* solution, a single thread of control is passed between the reasoners. In DANIEL, however, a different direction has been taken: the reasoners are supervised by a central module, they work concurrently and autonomously; each one of them derives a solution of its own. Then, their solutions are integrated by the central coordination component.

Intuitively, exploiting the flexibility of the blackboard paradigm by an opportunistic control strategy seems to be most advantageous. Moreover, the proposed control

4 Related Work

Normative conflicts in legal reasoning are investigated by a number of researchers, among others [19, 17]. However, these approaches do only focus on the handling of conflicts in statutory interpretation. In legal CBR, the problem of selecting the most appropriate case form a set of retrieved, possibly conflicting cases, is usually not conflict resolution but rather a matter of similarity. Conflicts among case law and statutes have not been investigated this far.

In the AI and Law field, only few approaches to cooperative distributed problem solving (CDPS) exist. Doubtlessly the most sophisticated among these is Skalak's and Rissland's CABARET [18], which is a domain-independent hybrid reasoning shell developed with GBB. In CABARET, a case-based and a rule-based reasoner are coupled via a blackboard. The reasoners are interleaved by central component, which embodies domain-independent control heuristics. It is assumed, that the reasoners complement each other, and consequently, there is no explicit conflict resolution.

Most other systems use cases when rules run out or vice versa, like [22, 15]. In the Dutch system PROLEXS [23], rules and cases are applied in a fixed order, which is determined by situation-specific heuristics.

Another, however "illegal", system concerned with the integration of CBR and RBR is Golding's ANAPRON, which has been developed for the problem of pronouncing names. In ANAPRON, cases are used to discard respectively confirm the result of the rule-based inference process. The knowledge sources have to independent from each other in order apply this model - which is definitely not true in the legal domain.

In distributed artificial intelligence (DAI), negotiation is broadly applied to resolve conflicts among agents, see, e.g., [21, 16]. All these approaches, however are not directly applicable in the legal domain. In law, a certain predicate is either true or false, a concrete state of affairs is either subsumed by a legal rule or not, which is not negotiable. Therefore, in this approach, a strategy for finding the most probable solution among the possible interpretation is developed.

5 Redundancy Concept

A wicked side effect of the concurrent application of the reasoners is that conflicts are much more likely to occur - more than one (partial) solution is derived. Coordination

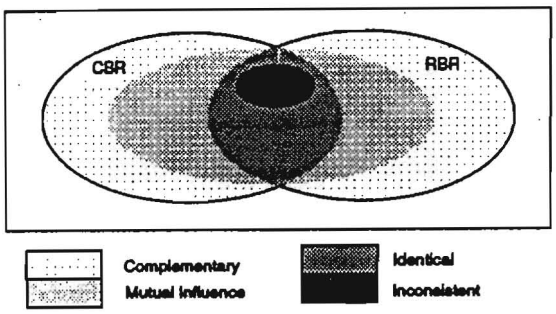

Figure 2: Overlapping of Knowledge Sources

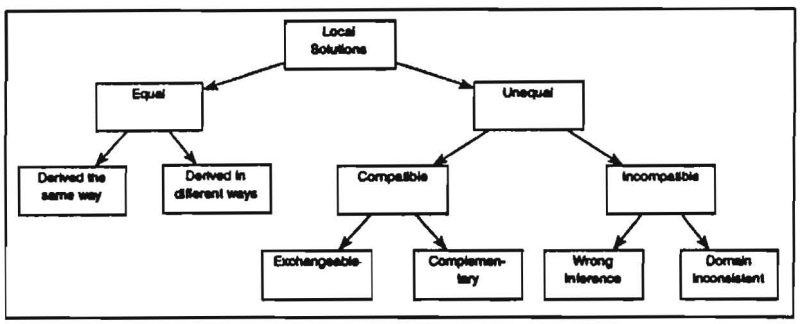

Figure 3: Taxonomy of local solutions

Generally, the local solutions can be *equal* or *unequal*, which does *not necessarily* mean that they are contradictory. If they are equal, they can be derived either in the same way (following the same argumentation) or in different way. In the first case, the application of both knowledge sources was merely superfluous, while in the latter case the confidence in the solution is increased, because the correctness of the problem solvers is *proven*. Anyway, no further coordination is needed.

More effort has to be spent, however, if the local solutions are unequal. If they are *incompatible*, the conflict can not be resolved by the system itself. In order to achieve an assessment of the system status, two possible situations have to be distinguished. On the one hand, there can be inconsistencies caused by *inconsistent domain knowledge*, e.g., if a court explicitly contradicts a statute for constitutional reasons. Problems like this exceed the competence of a system and should cause a corresponding message to the user. In the following, it will be assumed that cases like this were eliminated in the knowledge acquisition phase.

On the other hand, the reasoners the reasoners do not always work accurately and can derive incorrect solutions. In this case, their concurrent execution does not serve

Moreover, the inherent limitations of both reasoning modes, case-based and rule-based, can be minimized by their concurrent and autonomous application.

Since the problem of overlapping expertise and non-independent knowledge is not limited to the legal domain, this model can also be adapted and applied to other domains.

7 Background

This work is part my master's thesis, which I am currently preparing at the Chair for Applied Computer Science of Professor Dr. Franz Stetter, whose research focus is on Software Engineering. My supervisor is Dr. Michael Weiss, who graduated with a thesis on distributed AI for design, in which he developed HBBS, a hierarchical blackboard architecture.

After the completion of my Wirtschaftsinformatik degree at the University of Mannheim, I will start as a Ph.D. candidate in the Intelligent Systems Program at the University of Pittsburgh. This program was founded by Bruce G. Buchanan as a center for advanced education and research in artificial intelligence. There, I will continue my work in the field of AI and Law with Prof. Kevin D. Ashley [2, 4, 3].

I already presented my work in a several workshops and conferences [6, 11, 8, 9]. Recently, my student paper for the American National Conference on Artificial Intelligence (AAAI94) was accepted [10]. Furthermore, I could participate in workshops at conferences on object-orientation with my work in object-oriented modeling [12, 7].

References

[1] R. Alexy. *Theorie der juristischem Argumentation.* Suhrkamp, Frankfurt/Main, 1991. Theory of Legal Argument.

[2] K.D. Ashley. *Modeling Legal Argument.* MIT-Press, Cambridge, MA, 1990.

[3] K.D. Ashley and V. Aleven. Generating dialectical examples automatically. In *Proceedings of the Tenth National Conference on Artficial Intelligence*, pages 654 – 660, San Jose, CA, 1992.

[4] K.D. Ashley and V. Aleven. Using logic to reason with cases. In *Proceedings of the First European Workshop on Case-Based Reasoning*, pages 373–378, Kaiserslautern, 1992.

[14] B.G. Buchanan and R.G. Smith. Fundamentals of expert systems. In A. Barr, P.R. Cohen, and E.A. Feigenbaum, editors, *The Handbook of Artificial Intelligence, Vol. IV*, pages 149–192. Addison Wesley, Reading, MA, 1989.

[15] A.v.d.L. Gardner. *An Artificial Intelligence Appraoch to Legal Reasoning*. MIT Press, Cambridge, MA, 1987.

[16] M. Klein, editor. *Proceedings of the IJCAI-93 Workshop on Conflict Management*, Chambery, 1993.

[17] H. Prakken. A tool in modelling disagreement in law: preferring the most specific argument. In *Proceedings of the Third International Conference on Artficial Intelligence and Law, Oxford*, pages 165–174, 1991.

[18] E.L. Rissland and D.B. Skalak. CABARET: Rule Interpretation in a Hybrid Architecture. *International Journal on Man-Machine Studies*, 34(1):839–887, 1991. Also appeared as University of Massachusetts Technical Report COINS-TR 90-97.

[19] G. Sartor. *Artificial Intelligence and Law*. Complex. Tano, Oslo, January 1993.

[20] D. B. Skalak. Options for controlling mixed paradigm systems. In *Proceedings of the 1989 DARPA Case-Based Reasoning Workshop*, pages 318–323, Pensacola Beach, FL, 1989. Morgan Kaufman.

[21] E.P. Sycara. *Resolving Adversarial Conflicts: An Approach Integrating Case-Based and Analytical Methods*. PhD thesis, School of Information and Computer Science, Georgia Institute of Technology, Atlanta, GA, 1987. published as Technical Report GIT-ICS-87/26.

[22] G. Vossos, J. Zeleznikow, T. Dillon, and V. Vossos. An example of integrating legal case-based reasoning with object-oriented rule-based systems: IKBALS II. In *Proceedings of the Second International Conference on Artficial Intelligence and Law, Vancouver*, pages 31 – 41, Vancouver, 1989.

[23] R.F. Walker et al. PROLEXS: creating law and order in a heterogenous domain. *International Journal on Man-Machine Studies*, pages 35–67, 1991.

Ursprünglich lagen diese Grammatiken als Transitionsnetzwerke vor, die um Unifikation erweitert waren. Auf diese sogenannten »UTNs« (vgl. [Russi 1990]) wandte man ein an [Pereira/Warren 1980] angelehntes Verfahren an, um vollautomatisch Definit-Klausel-Grammatiken zu erhalten, wie sie auch zum Beispiel die Programmiersprache Prolog kennt. Obwohl DCGs sehr oft direkt in Prolog abgearbeitet werden, wird in SVOX aus Gründen der Effizienz ein in Modula-2 geschriebener Chart-Parser eingesetzt.
Meine Aufgabe war es nun einerseits, die Grammatiken zu verbessern, andererseits sollte die Syntaxanalyse beschleunigt werden. Die Änderungen am Parser sind jedoch nicht Gegenstand dieses Artikels: Im wesentlichen habe ich den Earley-Algorithmus implementiert und durch Techniken wie Überprüfung der First-/Follow-Relationen erweitert, die aus dem Compilerbau bekannt sind.
Es erwies sich als nötig, große Teile der Grammatik neu zu schreiben, wobei ich großes Gewicht auf die Verarbeitungszeit legte. Das Aufwendigste bei einer Unifikationsgrammatik ist das Unifizieren selber — es lohnt sich also, hier anzusetzen, bei dieser Operation, die unter Umständen pro Satz mehrere tausend Mal durchgeführt werden muß. Nun ist es vor allem das Unifizieren von *komplexen* Termen, das viel Zeit in Anspruch nimmt; um zwei Atome miteinander zu unifizieren, braucht es dagegen nicht viel mehr als einen simplen Vergleich.
Es liegt aus diesem Grund nahe, eine Grammatik zu schreiben, die sich mit atomarer Unifikation begnügt. Wie im nächsten Abschnitt gezeigt wird, kann eine solche Grammatik jedoch nur die Klasse der *kontextfreien Sprachen* verarbeiten. Kontextfreie Grammatiken sind nun allerdings in großen Kreisen der Computerlinguistik eher verpönt (vgl. zum Beispiel die Argumente gegen kontextfreie Grammatiken in [Haugeneder/Trost 1993]): Es heißt unter anderem, mit ihnen ließen sich komplexere linguistische Phänomene kaum beschreiben. Daß man mit kontextfreien Grammatiken durchaus einige Tiefe bei der linguistischen Analyse erreichen kann, soll der Rest dieses Artikels zeigen: Nach dem Einführen von zwei nützlichen Erweiterungen des Formalismus, die auf die Sprachklasse aber keinerlei Einfluß haben, wird erläutert, wie die SVOX-Satzgrammatik mit Koordination, Subkategorisierung und freier Wortstellung umgeht.
Es sei aber bereits an dieser Stelle angemerkt, daß im beschriebenen System keine semantische Verarbeitung stattfindet. In diesem Fall wäre eine »richtige« Unifikationsgrammatik, HPSG zum Beispiel, sicherlich ein geeigneteres Werkzeug als ein reiner DCG-Formalismus.
Weiterhin sei festgestellt, daß die nachfolgend vorgestellte Grammatik keinerlei theoretische Ansprüche erheben will: SVOX ist ein Programmpaket, das in der Praxis angewandt werden soll; es kann daher manche Frage einfach ignorieren, die man einer Grammatik-*Theorie* stellen müßte.

Warum ist die Grammatik kontextfrei?

Wie bereits erwähnt wurde, benutzt das System SVOX sowohl für die Darstellung des syntaktischen wie auch des morphologischen Wissens Grammatikregeln, die abgesehen von einer etwas anderen Syntax den Definit-Klausel-Grammatiken (DCGs) von Prolog entsprechen.
Mit Definiten Klauseln können durchaus auch kontext-*sensitive* Sprachen repräsentiert werden. Es sei hier lediglich auf [König/Seiffert 1989] verwiesen, wo sich auf Seite 112 eine

Aus diesem Grund möchte man bestimmte Konstruktionen als wahrscheinlicher als andere einstufen. Die eine Erweiterung des Formalismus ist daher eine Bewertung jeder Regelanwendung in Form von »Strafpunkten«, die über die gesamte Ableitung hinweg aufsummiert werden. Die Ableitung mit der niedrigsten Punktzahl gilt als wahrscheinlichste Alternative und wird als einzige den nachfolgenden Komponenten übergeben.
Die zweite Erweiterung sind als »unsichtbar« markierte Regeln. Sie werden ganz normal verarbeitet, ihre Anwendung erscheint aber nicht im Baum, der an die anderen Module weitergegeben wird.
Nachfolgend werden einige konkrete Anwendungen dieser beiden Erweiterungen gezeigt, die für die Behandlung von Koordination, Subkategorisierung und freier Wortstellung hilfreich sind.

Pseudo-Operatoren

Das Einführen von echten Operatoren bzw. von Funktionen, wie es beispielsweise in HPSG möglich ist, hätte einen gewissen Aufwand bedeutet; insbesondere wäre es nötig geworden, den Modula-Quelltext des Parsers zu ändern, um neue Funktionen zu implementieren — was einerseits sehr unschön wäre und andererseits eine gewisse Vermischung der Wissensrepräsentation mit der Verarbeitung bedeuten würde. Durch die Möglichkeit, eine Regelanwendung nicht im Syntaxbaum erscheinen zu lassen, kann man jedoch »Pseudo-Operatoren« deklarativ einführen, welche zu einem gewissen Grad dieselbe Funktionalität bieten, ohne eine echte Erweiterung des Formalismus vornehmen zu müssen. Ich möchte das Prinzip nachfolgend am Beispiel des logischen »Oder« zeigen.
Für das Erzeugen der Satzmelodie ist es von grundlegender Bedeutung, zu wissen, ob der Satz eine Frage oder eine Aussage darstellt. Taucht irgendwo im Satz eine Konstituente auf, die durch bestimmte Wörter oder durch die Wortstellung auf eine Frage hindeutet, soll der gesamte Satz als Frage betrachtet werden. Die oberste Konstituente trägt daher ein Attribut »Frage«, dessen Wert von unten nach oben »vererbt« wird.[2] Dazu wird ein »Operator« benötigt, der das boolesche »Oder« berechnet. Diese »Funktion« kann folgendermaßen notiert werden:[3]

```
or(f, f, f).                    0 Strafpunkte · unsichtbar
or(t, _, t).                    0 Strafpunkte · unsichtbar
or(_, t, t).                    0 Strafpunkte · unsichtbar
```

Ein Beispiel für die Benutzung:

```
s(F) --> np(F1),
         vp(F2),
         or(F1,F2,F).           1 Strafpunkt · sichtbar
```

Neu ist eine solche deklarative Definition von Funktionen zwar nicht, nur wird sie eher auf dem Gebiet der Logischen Programmierung als beim Schreiben von Grammatiken benutzt. In einer »traditionellen« Definit-Klausel-Grammatik müßte der »Funktionsaufruf« entweder

[2] Es ist eigentlich nicht korrekt, bei einer Unifikationsgrammatik von »Vererbung« zu sprechen — Information wird beim Unifizieren nicht weitergegeben, sondern via *Structure Sharing* geteilt. Dennoch verdeutlicht dieser Sprachgebrauch vielleicht die Funktionsweise der Grammatik.

[3] Die Grammatiken des SVOX-Systems verwenden eine etwas andere Notation; ich habe sie hier zur besseren Verständlichkeit an den DCG-Formalismus von Prolog angeglichen.

Es ist zwar nicht der Fall, daß Konstruktionen wie »Ich sah Anna, Bruno« völlig ungrammatisch wären, sie sind bei Adjektiven sogar sehr häufig: »Das große, schöne Haus«. Sie sollen daher nicht völlig ausgeschlossen, sondern lediglich als eher schlecht bewertet werden, wobei die Zahl der Strafpunkte je nach Wortart variiert.
Betrachten wir nun die Grammatik für eine allgemeine Konstituente X, die koordinierbar sei. Gezeigt werden nur diejenigen Features, die für die Koordination von Bedeutung sind; es ist klar, daß zum Beispiel Agreement ebenfalls behandelt werden muß, oder daß Vorkehrungen zu treffen sind, damit eine Konstruktion wie »weder Anna als auch Bruno« ausgeschlossen wird.

```
x --> x-mult(f, f).          1 Strafpunkt  nicht koordiniert
x --> x-mult(f, t).         30 Strafp.     ohne Konj. koordiniert
x --> x-mult(t, t).          1 Strafp.     mit Konj. koordiniert
```

Der Wert des ersten Attributs ist gleich `t`, wenn die Konstruktion eine Konjunktion enthält; der Wert des zweiten zeigt, ob es sich überhaupt um eine koordinierte Konstruktion handelt. Wie zu sehen ist, bestraft die Grammatik Konstruktionen, die ausschließlich durch Aufzählen koordiniert sind.

```
x-mult(f, f) --> x1.                    1 Strafp.   [Anna]
x-mult(Kj, t) --> x1, x-mult(Kj, _).    1 Strafp.   [Anna] [Bruno und C.]
x-mult(t, t) --> x1, x-konj.            1 Strafp.   [Bruno] [und Cäcilie]
x-mult(t, t) --> x-konj1, x-konj2.      1 Strafp.   [weder A.] [noch B.]

x-konj --> konj, x.                     1 Strafp.   [sowie] [A. und B.]
x-konj1 --> konj1, x.                   1 Strafp.   [weder] [Anna]
x-konj2 --> konj2, x.                   1 Strafp.   [noch] [Anna]
```

Offensichtlich muß die Grammatik rekursiv sein: Man will beliebig lange Aufzählungen zulassen, und es sollen auch beliebig tiefe Verschachtelungen möglich sein.
Ein Problem dieser Implementation sind Konstruktionen wie in »Morgen heiraten Anna und Bruno, Cäcilie und Daniel und Emil und Frieda«, bei denen die richtige Lesart zwar erkannt, aber sehr schlecht bewertet wird. Eine korrekte Analyse (und damit eine verständliche Aussprache) könnte in diesem Fall aber nur mit Hilfe einer semantischen Verarbeitung erreicht werden. In der Mehrzahl der Fälle zeigte sich jedoch, daß Koordinationen vernünftig analysiert werden.

Subkategorisierung, Fernabhängigkeiten und freie Wortstellung

Jedes Verb »verlangt« (subkategorisiert) nach bestimmten Kategorien; das Verb »sehen« erwartet zum Beispiel ein Subjekt und meistens ein Akkusativobjekt: »*Ich* sehe *ein Einhorn* im Garten.« Diese Konstituenten werden Komplemente genannt; freie Zusätze (»im Garten«) heißen Adjunkte. Ein bekanntes Problem beim Entwickeln von Grammatiken für das Deutsche ist die sehr freie Wortstellung der Komplemente und Adjunkte. »Im Garten sah er gestern ein Einhorn«, »Gestern sah er ein Einhorn im Garten«, »Ein Einhorn sah im Garten gestern er« — manche der Varianten mögen seltsam klingen, vollkommen ungrammatisch ist aber keine. Ein System, das von Menschen stammende Sätze des Deutschen verarbeiten soll, muß damit rechnen, Adjunkte und Komplemente in beliebiger Reihenfolge anzutreffen. Ein Formalismus, der Regeln für die hierarchische, vertikale Abhängigkeit (immediate dominance) mit Regeln für die Wortstellung innerhalb einer Konstituente (linear precedence) ver-

Die Wortgrammatik sorgt dafür, daß Verben im Infinitiv nicht nach einem Subjekt, sondern nur nach den anderen möglichen Komplementen subkategorisieren können. Dadurch ist es nicht notwendig, Konstruktionen mit Kontrollverben (Raising-/Equi-Verben) gesondert zu behandeln.
Nun ist bekannt, welche Kategorien vorhanden sein müssen, damit ein Satz vollständig ist; es bleibt die Frage, mit welchem Mechanismus die einzelnen Komplemente abgebunden werden. Betrachten wir dazu zunächst, an welchen Stellen im Satz Komplemente auftreten:

Gestern	hat Maria vor Zeugen behauptet,	ein Einhorn gesehen zu haben.
Vorfeld	**Mittelfeld**	**Nachfeld**
Genau eine Konstituente	*Null bis mehrere Konstituenten in beliebiger Reihenfolge*	

Ein Aussagesatz besteht aus dem Vorfeld (mit genau einer Konstituente), dem finiten Verb, dem Mittelfeld und eventuell aus einem Partizip; die Behandlung des Nachfelds geschieht mittels besonderer Mechanismen, die hier nicht erläutert werden.
Welche Komplemente vorhanden sein müssen, hängt von der Subkategorisierung des Verbs ab; die SVOX-Grammatik trennt daher erst das Vorfeld ab und unterteilt dann den Rest in die drei Teile finites Verb, Mittelfeld und Partizip. Die Hauptaufgabe der Satzgrammatik ist es, die Komplemente im Mittelfeld »aufzusammeln« und mit dem Vorfeld und dem Subkategorisierungsrahmen des Verbs so in Beziehung zu setzen, daß nur solche Sätze akzeptiert werden, denen weder ein Komplement fehlt noch eines überzählig haben.
Angelehnt an die Behandlung der Fernabhängigkeiten (Unbounded Dependencies) in HPSG verwendet die hier beschriebene Grammatik *zwei* Mengen, um diese Aufgabe zu lösen. Zum einen trägt jede Konstituente des Mittelfeldes die Information, welche noch nicht abgebundenen Komplemente sie enthält (im folgenden »HAVE« genannt), und zum anderen gibt eine »NEED« genannte Menge an, welche Komplemente benötigt werden, bisher aber noch nicht abgebunden werden konnten. Anders als HPSG arbeitet die SVOX-Grammatik jedoch ohne Traces.
Das Vorgehen der Grammatik soll nachfolgend anhand des Satzes »Diesen Ball wollte ich Maria gestern geben« erläutert werden; auf der nächsten Seite ist der Syntaxbaum in vereinfachter Form abgedruckt, welchen der Parser als Ergebnis liefert. Insbesondere sind die NEED- und die HAVE-Menge als Mengen dargestellt, obwohl sie wie die Subkategorisierungsinformation der Verben als Bündel von separaten Features implementiert sind.
Das Mittelfeld wird rekursiv aufgebaut: Das Verb »geben« subkategorisiert als Infinitiv nach einem Akkusativobjekt und optional nach einem Dativobjekt; die Wortgrammatik sorgt dafür, daß Infinitive niemals nach einem Subjekt subkategorisieren, damit Konstruktionen mit Kontrollverben behandelbar sind. Somit enthält das aus »geben« alleine bestehende Mittelfeld eine nicht abgebundene Infinitiv-Konstruktion, und es müssen noch ein Akkusativobjekt und ein Dativobjekt gefunden werden.
Ein Adjunkt wie »gestern« ändert nichts an den beiden »Mengen«; die Werte der Attribute werden unverändert übernommen. Zusammen mit der Dativ-NP »Maria« ergibt sich ein Mittelfeld, dessen HAVE-Menge eine Infinitivkonstruktion enthält und dessen NEED-Menge anzeigt, daß noch ein Akkusativobjekt zu finden ist.

Da Mengen durch Attribute mit booleschem Wertebereich simuliert werden, lassen sich diese beiden Regeln mit Hilfe einer Negation zu einer einzigen zusammenfassen: Das neue Mittelfeld hat genau dann einen Dativ überzählig, wenn das alte keinen benötigte. Somit enthält die Grammatik für jedes der möglichen Komplemente eine einzige Regel. Es wird hierbei angenommen, daß kein Dativ auf ein Mittelfeld trifft, das bereits einen nicht abgebundenen, »überzähligen« Dativ enthält. Dies ist eigentlich nicht korrekt, hat aber bisher zu keinen falschen Analysen geführt. Der Grund dafür ist, daß selten zwei gleiche Konstituenten nicht-lokal abgebunden werden.
Ist das Mittelfeld vollständig auf die beschriebene Art aufgebaut worden, wird es mit dem Verb zur »klassischen« Verbalphrase verknüpft. Dies bewerkstelligt ein besonderer Abbinde-Operator:

Verb	Mittelfeld		Verbalphrase	
NEED	NEED	HAVE	NEED	HAVE
t	f	t	f	f
t	f	f	t	f
f	t	f	t	f
f	f	t	f	t
f	f	f	f	f

Benötigt ein Verb beispielsweise einen Dativ und»bietet« das Mittelfeld einen Dativ an, hat die Kombination von beiden weder einen Dativ »zuviel« noch einen »zuwenig«. Die übrigen Zeilen können analog gelesen werden. Manche Kombinationen fehlen in der Tabelle; zum Beispiel ist es ausgeschlossen, daß ein Mittelfeld gleichzeitig einen Dativ überzählig hat und einen braucht, denn in diesem Fall wäre der Dativ schon beim Aufbau des Mittelfeldes abgebunden worden.
Als letztes muß die »Verbalphrase« mit dem Vorfeld verbunden werden: Ein Satz besteht aus einem Dativobjekt und einer Verbalphrase, die noch einen Dativ benötigt, aber nichts überzählig hat. Ähnliches gilt für die anderen möglichen Komplemente, zusätzlich kann im Vorfeld auch ein Adjunkt (»Gestern brannte es«) oder ein Expletivum (»Es brennt ein Feuer im Garten«) stehen; in diesen beiden Fällen müssen sowohl die NEED- als auch die HAVE-Menge leer sein.

Schlußbemerkungen

Das Anwendungsgebiet der Sprachsynthse läßt niedrige Rechenzeiten besonders wünschenswert erscheinen: Die Grammatiken des SVOX-Systems beschränken sich daher auf atomare Unifikation, was die syntaktische Analyse stark beschleunigt. Allerdings hat dies unmittelbar zur Folge, nur noch die Klasse der kontextfreien Sprachen verarbeiten zu können. Entgegen einer in der Computerlinguistik verbreiteten Meinung war es aber durchaus möglich, mit der beschriebenen Grammatik auch komplexere linguistische Phänomene zu behandeln.
Das vorgestellte Verfahren kann für Anwendungsgebiete, die lediglich eine syntaktische Analyse erfordern, einen Mittelweg zwischen einer linguistisch sehr sauberen, aber zeitintensiven Unifikationsgrammatik einerseits und einer nur oberflächlichen, daher qualitätsmindernden Analyse andererseits darstellen.

TimePlan : Towards a time-based domain-independent planning system

C.C.Marinagi

Institute of Informatics and Telecommunications
N.C.S.R. Demokritos
15310 Aghia Paraskevi, Athens, Greece

e-mail : katerina@iit.nrcps.ariadne-t.gr
Tel : +301-6510310, Fax:+301-6532175

Abstract

In this paper we describe an experimental time-based planning system which is in development stage. Existing planning systems are compared with TimePlan as well as with its possible extension. TRL temporal reference language can be incorporated in TimePlan to provide techniques for temporal deduction. Additionally, TGS time graph management system can be used to handle temporal information fast and efficiently.

1. Introduction

A problem in the field of Artificial Intelligence is the design of systems, called planning systems, that can describe a sequence of actions to be executed by an agent, such as an intelligent robot.

Planning has been extensively studied during the last thirty years, and has been applied onto various areas such as process control, production control, project management, circuit design, spacecraft activities, etc...

However, realistic planning systems bear great temporal and conceptual complexity and major technical obstacles exist that still remain to be solved. To overcome such problems, issues such as time, space, uncertainty, and conceptual structures must be incorporated into planning systems. New representation schemes and planning algorithms could be designed and developed to support such issues.

During the long period of planning research, a lot of planning systems have been developed and a great progress has been made comparing to the

reach our aims, but consists a basis for further testing of ideas resulting from research. In order to provide the reader with the most essential background knowledge, in section 2, we briefly give the definitions of some basic planning terms as well as some of the important characteristics of existing planning systems. Old fashion techniques that have been abandoned are not referred at all. In section 3, it is explained which of these characteristics are considered desirable to be included to our system and in section 4 which of them have already been implemented. Finally, in section 5 we conclude and give future perspectives.

2. Background knowledge and previous work

2.1 Basic planning terms

There are several surveys in the literature of planning systems, [4],[15],[19] that can introduce the reader in planning issues. In the following we give the definitions of the basic planning terms :

A *planner* has to find a collection of temporally related actions that should be executed in order to achieve its goal.

An AI planning system generates an action sequence which can achieve the explicitly stated goals. The essential *inputs* of a planner are :

a) the current state of the world (the planning environment),

b) the operator schemata (or else the action specifications), which describe actions in terms of their preconditions and effects and

c) a set of goals ,which is a description of desired future conditions.

The *output* of a planner is a plan which satisfies the goals. Thus, a *plan* is a representation of a course of actions that is a solution to a given problem.

2.1 Classification of planning systems

When the plan representation language and plan generation algorithms are expected to work for a large variety of application domains the planner is characterised as *domain-independent*. On the other hand, when specific heuristics are used for the control of the operations of a planner, the planner is characterised *domain-dependent*. The planning research that focuses on

window and duration associated with the activity. FORBIN [10] uses a 'task network' for plan representation where there are not nodes and arcs, but time lines that are split and joined.

2.3 Domain Representation

As far as *domain representation* is concerned, planning systems have introduced different formalisms.

At first, *action representation* is concerned. Researchers usually distinguish the notion of action from the notion of event, [34]. In the following the term action will be used to refer to both notions, because a common formalism is used for representing both of them.

An action is called *'primitive'* when it can be executed directly by the planner or it is called *'macro'* when it can be decomposed in more detailed actions.

The basic idea of STRIPS [13] to model actions as *operators schemata,* having preconditions and postconditions which can be added or deleted, has generally be accepted and slightly changed by consequent systems. What partially-ordered systems need more are *action reduction schemata* to represent the instructions for action or goal expansion to lower level actions. These can be separated from operator schemata (e.g. NOAH's SOUP code [27], FORBIN's methods[10]) or incorporated in a unified formalism that can represent both of them [e.g. NONLIN's [33], O-PLAN's [8] Task Formalism (TF)].

Secondly, there is also need to keep record of the *effects* of each action and their *'scope'*. The term *'scope'* used in SIPE [35] , is analogous to the terms : *'protection interval'* used in HACKER [31], *'holding period'* used in INTERPLAN [32], *'range'* used in NONLIN [33], and means the time between when a goal is achieved and the point at which it was required to satisfy a condition on a later node. Planning systems need to record effects along with their scope, in order to keep the *'purpose'* of an action, that is why the particular action has been selected.

NOAH uses the Table of Multiple Effects (TOME) to keep the instantiated effects of an action at each node. This information is used during conflict detection and resolution. NOAH does not distinguish between important effects which achieve a precondition on some later node and other

correction of interactions. In planning systems that resources are handled, the mechanism that allocate and deallocate them check also for usage conflicts. If such conflicts occur they can be avoided.

a) An unsolved goal is chosen from the goal list to be satisfied.
b) An operator is chosen to expand the goal and the goal is expanded. *(Node Expansion)*
c) Check if the effects of the chosen operator generate inconsistencies and correct them by ordering. *(Conflict detection & correction)*
d) Remove the achieved goal from the list of unsolved goals and add the preconditions of the operator in the list . Preconditions are now the new goals.
e) Go to step a).

Figure 2.a The basic steps of a Planning Algorithm

2.4.2 Guiding search

During the execution of planning algorithms some choices must be made that can be kept as *backtrack choice points.* These are:

a) which action reduction schema to use to expand a node
b) how to order two conflicting nodes

NOAH performs deterministically, without backtracking at all. Most of the other hierarchical partially-ordered planners keep choice points for backtracking. They just use different techniques to search the space of plans. Backtracking can be classical chronological [13],[32],[34],[35], or dependency-directed [28],[6] when the system backtracks directly to the choice point that is responsible for failure.

On one side, avoiding backtracking can cause loss of solutions and on the other side uncontrolled backtracking can be very expensive. Therefore, planning systems employ various techniques in order to prune the search space. *'Meta-rules'* (e.g. MOLGEN [29],[30]) and *human interaction* (e.g. SIPE [35]) can manage control decisions. *Time* and *resource usage constraints* can also play an important role in taking decisions. Finally, *heuristic techniques* can be used to guide search which may be domain-dependent or independent. For example, *temporal coherence* (TC) [12] and

representing and reasoning about temporal information. Temporal planners (FORBIN [10], TRIPTIC [26]) are based on TMM. Another planning system that employs a time network management system (TNMS), which is similar to TMM, is O-PLAN2 [11]. Despite the fact that the metric temporal reasoning and incremental updates supported by TMM are useful in planning , the problem is that it is not certain that persistence, projection and overlap chaining are suitable forms of inference for current planning techniques. This problem is pointed out in [23] by Nebel and Backstrom and is also admitted by Boddy [5] who comments that " the correct level of integration between a planner and a temporal reasoning system such as the TMM is an open question".

3. Design Requirements of the TimePlan system

TimePlan system is an hierarchical partially-ordered planner based on the class of NOAH-NONLIN planners. In this planning system time is incorporated in a DEVISER-like manner.

After evaluating the characteristics of the existing planning systems we have decided to include a union of the most necessary characteristics. We use Tate's [33] terminology to describe most of our issues. In the following, we will describe the next TimePlan's basic design requirements :

- the operator schema, that is how actions are represented
- the plan network, that is the data structure that is used for plan representation
- the planning algorithm, that is the basic steps that the planner follows to produce a plan
- how time requirements are incorporated into the system.

3.1 Operator representation

The desirable operator schemata that we adopt for action description includes information that is illustrated in figure 3.a.

The *operator's name* is a unique name that characterises the operator. The *operators identification* is the parameterised action description. Several schemas that may have the same operator_id, are alternatives for expanding a node.

Effects are also separated into *primary* and *general effects* to differentiate between those that may be used to satisfy conditions and those that are general and considered as side-effects. The effects are not separated into added and deleted as STRIPS postconditions. They are expressed as propositions which can be negated or not.

3.2 Plan network

The *plan network* is a dynamically changing collection of nodes. In the network appear three types of nodes : *action nodes, goal nodes* and *phantom nodes.* There are also two special nodes the *start* and *stop* nodes that correspond to initial and final state respectively.

A goal node represents a condition which must be satisfied. In order to satisfy this condition an appropriate action is selected and an action node along with its subgoals replaces the initial goal.

Therefore each action node in the network is linked with its parent action node and children goal nodes (action subgoals). These goal nodes will be consequently replaced by other action nodes, etc. This operation is called *node expansion.*

Each action node has an associated TOME assertion, where its instantiated effects (both primary and general) are kept. Additionally, when an effect is protected to hold up to a particular node, then it is kept in GOST along with its protection range. A subgoal is protected to hold up to the action node that needs it as precondition.

Node expansion finishes when the condition corresponding to a goal node is satisfied by an initial condition. In this case this goal node is linked

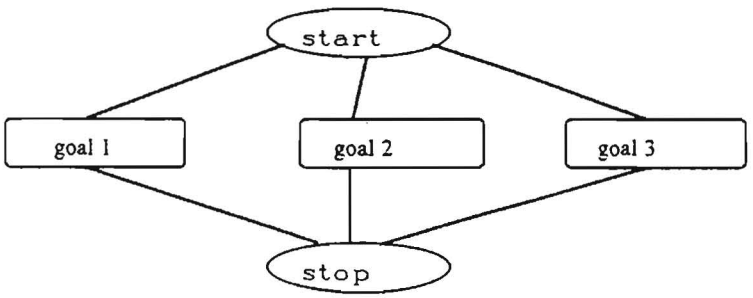

Figure 3.b Beginning plan generation

true at final state. Top goals are kept in a goal list. The system takes to satisfy one goal after the other. If a solution can not be found the system can replan by reordering goals and try the new sequence.

As a result, the planner can be characterized as *goal driven* that does a *depth-first search* through the space of possible plans.

As planning proceeds the system always works at a node, referred as 'current node' in the following. Each node has an associated number. This node number is unique for every goal node and every phantom node. An action node takes the same number node with the goal node that has previously been expanded by selecting this action node.

As it is already mentioned in paragraph 2.4, according to Tate's terminology, three fundamental operations are adopted here : *linking*, *node expansion* and *conflict resolution & ordering*. The details of the implemented algorithm are explained below :

The first top goal is selected from goal list to be satisfied. If it can not be linked to initial state , it has to be expanded. The planner selects an action with a primary effect that matches the goal and the subgoals of this action become the new goals which are put on top of goal list. The effects of the action become the TOME assertions of the action node. The primary effect that matched the subgoal becomes a GOST assertion.

When an action that has been selected, has more than one subgoals, then parallel branches are created for each one. Interactions, being helpful or harmful, may be appeared between parallel nodes. For this reason, each time a goal or subgoal is selected from goal list, all parallel nodes are checked for any harmful interactions (conflicts). In order to detect them, the planner compares the goal with TOME assertions of parallel actions nodes. Conflicts are resolved by ordering conflicting nodes while respecting protections. A primary effect is protected to hold up to the beginning of the actions of which it is a precondition and in this case is also a GOST assertion.

After conflict detection and resolution of a goal node, the planner looks if any helpful interactions exist, in other words, tries linking operation. The current goal node, may be true somewhere in the existing plan or else at initial state. Such a case appears when the goal matches with a TOME assertion of a parallel action node or else when the goal matches with one of the initial conditions.

Then if linking fails, the goal is expanded by selecting an action. Candidate actions are these that have a primary effect that matches with the goal. Time and resource constraints, assumptions, as well as heuristic rules, may reject some alternatives. If more than one candidate action remain, one is selected and the others are kept for backtracking.

When an action is selected, the planner compares its TOME assertions with those of parallel action nodes. If a conflict is found an ordering has to be decided, respecting GOST assertions.

Actually, each time a conflict is resolved there are two possible orderings that can be applied. Time and resource constraints must be evaluated and heuristics may be applied to help rejecting one of the two possible orderings. Then, if still two alternative orderings remain, this becomes a choice point for backtracking.

The plan construction continues by applying the three fundamental operations from the beginning, having selected the next unsolved goal from the goal list.

3.4 Design issues on incorporating time requirements

Each operator schema contains temporal information related to the duration of the action, the starting time point and the finishing time point of the action. Such information can be used in cases of real-world planning applications where time plays an important role.

The temporal parameters of the operator schema incorporate some additional complexity into the algorithm of the planner. The operator schema, the list of goals, and the planning algorithm must take temporal information into consideration. Some of the points that this should be done are the following :

- Operator schema contains information about the duration of an action. The start or the end time of the execution of an action may also be defined.
- Each goal in the goal list is augmented with a temporal template indicating the expected time that this goal must be achieved. The expected time of the achievement of top goals is defined from the beginning. As planning proceeds new subgoals will have to be true at the start time of the actions of which they are preconditions.

```
topgoals ([G1,G2,... Gn]).
initialstate ([ F1,F2,....Fn]).

planner :- topgoals(G), take_to_satisfy(G).

take_to_satisfy([G|Goals]) :- linking (G),!,
                              take_to_satisfy(Goals).
take_to_satisfy([G|Goals]) :-
                        expand(G, Action, Subgoal, Effects),
                        tomeassert(Action),
                        gostassert(Action),
                        detect_resolve_action_conflict( Action),
                        detect_resolve_subgoal_conflict( Subgoals),
                        take_to_satisfy(Subgoals),
                        take_to_satisfy(Goals) .

linking(G) :- satisfied(G, Node),
              order(G,Node),
              propagate_changes_of_time_bounds(G),
              assert(phantom(G, at(Node))).

detect_resolve_action_conflict( Action, Node) :-
                         check_tome(Action, Node),
                         check_gost(Action, Node),
                         resolve_by_ordering(Node,Paral_Node),
                         propagate_changes_of_time_bounds(Action).

detect_resolve_subgoal_conflict( [S|Subgoals]) :-
                          check_tome(S, Node),
                          check_gost(S, Node),
                          resolve_by_ordering(Node, Paral_Node),
                          propagate_changes_of_time_bounds(S),
                          detect_resolve_subgoal_conflict(Subgoals).
```

of the operator schema (as in SIPE [35]). Alternatively, resources may be specified as conditions, that must be reserved as long as the particular action proceeds or for a definite period of time [7],[10],[34]. We intend to incorporate resources into TimePlan in one or the other way. TGS's [16] suggestion that a resource can belong to the set of resource types {shared, unary shared, dedicated, unary dedicated, consumable, partially consumable}, will be considered.

Acknowledgements

I want to thank Dr.C.D.Spyropoulos, Dr.T.Panayiotopoulos for their supervision and support of this work.

References

[1] J. F. Allen, "Towards a General Theory of Action and Time", Artificial Intelligence 23, pp.123-154, 1984.

[2] J. F. Allen, "Planning as Temporal Reasoning", In Proceedings of the Conference on Principles of Knowledge Representation and Reasoning, pp.3-14, 1991.

[3] J. F. Allen and J. A. Koomen, "Planning Using a Temporal World Model", in Proceedings of the 8th IJCAI, Karlsruhe, Germany, pp.741-747, 1983.

[4] A. Barr and E. A. Feigenbaum, "The Handbook of Artificial Intelligence", VolII, Morgan Kaufmann, Palo Alto, Calf., pp. 515-562, 1981.

[5] M. Boddy, "Temporal Reasoning for Planning and Scheduling", SIGART Bulletin, Vol 4, No 3, 1992.

[6] D. Chapman, "Planning for Conjunctive Goals" Artificial Intelligence 32, pp.333-377, 1987.[TWEAK]

[7] P. Cheesman, "A Representation of Time for Automatic Planning", IEEE, pp. 513-518.

[8] K. Currie and A. Tate, "O-PLAN: the open system architecture", Artificial Intelligence 52, pp.49-86, 1991. [O-PLAN]

[9] T. Dean and D. McDermott, "Temporal Data Base Management", Artificial Intelligence 32, pp.1-55, 1987.

[22] D. P. Miller, "Planning by search through simulations" , PhD Thesis, Yale University, Dec. 1985.

[23] B. Nebel, C. Backstrom, "On the Complexity of Temporal Projection and Plan Validation", Proceedings AAAI-92, 10th National Conference on Artificial Intelligence, 1992, pp.748-753.

[24] T. Panayiotopoulos, C.D. Spyropoulos, E.V. Ioannidis, "TRL: A formal Language for Temporal References Part I : Specifications and Semantics", DEMO, 93/10, N.C.S.R. Demokritos, June 1993.

[25] E. Pednault, "Solving Multi-Agent Dynamic World Problems in the Classical Planning Framework" ,In Reasoning about Actions and Plans: Proceedings of the 1986 Workshop, eds. M.Georgeff, A.Lansky, San Mateo,Calif, Morgan Kaufmann, 1987.

[26] E. Rutten, "A temporal non-linear planner: TRIPTIC", Arbeitpapier, 582, GMD, 1991. [TRIPTIC]

[27] E. D. Sacerdoti, "A Structure for Plans and Behaviour", Amsterdam: Elsevier-North Holland, 1977. [NOAH]

[28] R. M. Stallman and G. J. Sussman, "Forward reasoning and Dependency-Directed Backtracking in a System for Computer-Aided Circuit Analysis", Artificial Intelligence 9, pp.135-196, 1977.

[29] M. J. Stefik, "Planning with Constraints", Artificial Intelligence 16 , pp.111-140, 1981. [MOLGEN]

[30] M. J. Stefik, "Planning and Metaplanning", Artificial Intelligence 16 , pp.141-169, 1981. [MOLGEN]

[31] G. A. Sussman, "A computational Model of Skill Acquisition", MIT AI Lab Memo, AI-TR-297, AI Lab., Massachusetts Institute of Technology, 1973.[HACKER]

[32] A. Tate, "Interacting Goals and their use", In Proceedings of the 4th IJCAI , Menlo Park, Calif, pp.215-218, 1975. [INTERPLAN]

[33] A. Tate, "Generating Project Networks", In Proceedings of 5th IJCAI, Menlo Park, Calif., 1977. [NONLIN]

[34] S. A. Vere, "Planning in Time: Windows and Durations for Activities and Goals", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.PAMI-5, No.3, pp.246-267, May 1983. [DEVISER]

[35] D. E. Wilkins, "Domain-independent Planning:Representation and Plan Generation ", Artificial Intelligence 22, pp.269-301, 1984. [SIPE]

1 Einleitung

Im Rahmen der Wissensentdeckung in relationalen Datenbanken versucht man immer mehr auch Verfahren des maschinellen Lernens einzusetzen. Hier bieten sich logikbasierte Verfahren aufgrund ihrer Leistungsfähigkeit an. Im logikbasiertes Lernen liegt das Lernergebnis in eingeschränkter Prädikatenlogik vor und ist von einem System interpretierbar. Andererseits besteht auch von der Seite des maschinellen Lernens ein großes Interesse, durch eine Anbindung an Datenbanken, sich größere Beispielmengen zugänglich und handhabbar zu machen.

Im nächsten Abschnitt beschreibe ich deshalb erst einmal die Berührungspunkte der verschiedenen Disziplinen, die ein Interesse am Einsatz von Lernverfahren in relationalen Datenbanken haben. Im weiteren werde ich die Anwendung des logikbasierten Lernverfahrens RDT [Kietz und Wrobel, 1992] auf relationale Datenbanken beschreiben. Im Abschnitt 3 stelle ich das Verfahren vor und beschreibe dann in den folgenden Abschnitten besondere Punkte bei der Anpassung an relationale Datenbanken. Der Abschnitt 6 zeigt dann die Unterschiede zwischen RDT, angewandt auf relationale Datenbanken (RDT/DB), und dem ursprünglichen RDT auf.

Im Abschnitt 7 beschreibe ich erste experimentelle Vergleichstests zwischen RDT und RDT/DB, an Hand von zwei Testszenarien. Zum einen wurde hierfür von mir der im maschinellen Lernen bekannte Sachbereich KRK [Quinlan, 1983] bearbeitet, und zum anderen eine reale Anwendung aus dem Bereich der Roboternavigation.

2 Induktives Lernen und relationale Datenbanken

2.1 Maschinelles Lernen im Knowledge Discovery

Im Rahmen des Knowledge Discovery (KDD) erhält das Maschinelle Lernen (ML) eine immer größere Bedeutung. Aus der Definition für Knowledge Discovery von Frawley, Piatetsky-Shapiro und Matheus wird die Ähnlichkeit der Zielsetzung deutlich[Frawley *et al.*, 1991].

Definition 1 (Knowledge Discovery/Data Mining)
Knowledge Discovery ist das Entdecken von potentiell nützlichen Informationen aus Daten.

2.2 Induktives Lernen in deduktiven Datenbanken

Eine weitere Anwendung und damit eine weitere Motivation, ein Verfahren des induktiven Lernens auf relationale Datenbanken anzuwenden, ist die Möglichkeit, diese als induktive Komponente in deduktiven Datenbanken einzusetzen. Deduktive Datenbanken bestehen aus explizieten Daten und implizieten Daten in Form von Regeln. Eine solche induktive Komponente könnte beim Aufbau und der Pflege einer deduktiven Datenbank durch das Entdecken von bisher nicht bekannten Abhängigkeiten hilfreich sein. So beschreiben Džeroski und Lavrač den Einsatz von LINUS [Lavrač und Džeroski, 1992] zum Entdecken von virtuellen Relationen in deduktiven Datenbanken [Džeroski und Lavrač, 1993].

Bei den bisher in den Abschnitten 2.1 und 2.2 genannten Verfahren handelt es sich ausschließlich um Attributwerte-Verfahren. Hier ist es nun von Interesse, ein logikorientiertes Verfahren wie RDT in Zusammenhang mit relationalen Datenbanken einzusetzen.

3 Das Lernverfahren RDT

RDT [Kietz und Wrobel, 1992] ist Bestandteil des Modellierungssystems MOBAL [Morik *et al.*, 1993] und wurde an der GMD entwickelt.

Bevor ich auf die Besonderheiten beim Lernen mit RDT über relationale Datenbanken eingehen, will ich das Verfahren RDT aus der Sicht des maschinellen Lernens charakterisieren.

RDT ist ein Verfahren aus dem *inductive logic programming* (ILP). Die Lernaufgabe von RDT kann wie folgt beschrieben werden:

Gegeben: Hintergrundwissen und eine Menge positiver und negativer Beispielen für einen zu lernenden Begriff C in funktionsfreier Klausellogik.

Ziel: Finde eine Hypothese H in funktionsfreier Klausellogik, die einem vom Benutzer definierten Akzeptanzkriterium genügt.

Das Akzeptanzkriterium setzt sich aus den folgenden Faktoren zusammmen:

Sei $P(X_1, \ldots, X_m)$ eine Menge von Prädikaten, die von den Attributen $X_1, \ldots, X_m$ abhängen. Weiterhin sei dann H eine Hypothese der Form $H : P(X_1, \ldots, X_m) \rightarrow q(X_1, \ldots, X_n)$. Dann sind mögliche Faktoren des Akzeptanzkriteriums:

solchen Verknüpfungsbeziehung mit den bereits belegten Prädikatvariablen stehen (siehe Beispiel Sortentaxonomie).

Weiterhin können von RDT zur Instanziierung der Prädikatvariablen die Prädikattopologie und die Sortentaxonomie ausgenutzt werden.

Prädikatentopologie
Prädikate können strukturiert, in Form eines Graphen, dessen Knoten Prädikate zugeordnet sind, repräsentiert werden. Eine Hypothese ist mit einer Prädikattopologie kompatibel, wenn die Prädikate der Prämisse in einem Vorgängerknoten oder im Knoten des Prädikats der Konklusion sind. Daraus folgt, daß die Suche nach Instanzen für die Prädikatvariablen auf wenige Topologieknoten beschränkt werden kann. Eine solche Topologie berechnet in MOBAL die Systemkomponente PST ([Klingspor, 1991],[Morik *et al.*, 1993, S.149-168]).

Sortentaxonomie
Es ist effektiv, nur sortenverträgliche Hypothesen zu testen. Ich will das an einem kleinen Beispiel deutlich machen.
Gegen sei folgendes Regelschema:

```
m_example(P1,P2,C):P1(Y) & P2(X,Y) --> C(X)
```

Nach der *relation chain* wird erst `P2` instanziiert, da nur die Variable `X` durch den Zielbegriff gebunden ist. Für `P2` können nur Prädikate ausgewählt werden, die die Sorte von `X` als als Sorte ihres ersten Arguments haben. Die Sortenbeziehungen werden in MOBAL durch die Systemkomponente STT berechnet ([Morik *et al.*, 1993, S.109-148])

Nachdem ich nun RDT beschrieben habe, will ich die Vorgehensweise noch einmal anschaulich skizzieren.

Zur Umsetzung der Relationen in Prädikate gibt es zahlreiche Möglichkeiten, von denen ich die folgenden drei in Betracht gezogen habe.

Repräsentation I:

Man übernimmt den Tabellennamen als Prädikatnamen und die Attribute der Relation als Prädikatattribute.

Abbildungsvorschrift: R über $a_1, ..., a_n \rightarrow R(a_1, ..., a_n)$.

Repräsentation II:

Eine andere Möglichkeit ist die Zerlegung der Relation in $n+1$ Prädikate der Form:

Abbildungsvorschrift: R über $a_1, ..., a_n \rightarrow \forall(a_x(R, a_j, ..., a_l, a_x))$,
$R(a_j, ..., a_l)$,
$a_j, ..., a_l$ sind Primärschlüsselattribute der Relation R, und x läuft im Intervall $[1, ..., n]$ ohne $j \ldots l$.

Repräsentation III:

Die dritte Form ist ähnlich der Repräsentation II. Auch hier erhält man $n+1$ Prädikate, nur mit dem Unterschied, daß die Relationsbezeichnung (Tabellenname) mit in den Prädikatnamen gezogen wird.

Abbildungsvorschrift: R über $a_1, ..., a_n \rightarrow \forall R\_a_x(a_j, ..., a_l, a_x)$,
$R(a_j, ..., a_l)$,
$a_j, ..., a_l$ sind Primärschlüsselattribute der Relation R, und x läuft im Intervall $[1, ..., n]$ ohne $j \ldots l$.

4.2 Diskussion der Repräsentationen

Der Vorteil der Repräsentation I liegt in der sehr einfachen Überführungsvorschrift. Dem stehen allerdings zwei gravierende Nachteile gegenüber. Der erste Punkt ist, daß nicht mehr auf einzelne Attribute der Relation zugegriffen und somit nur die ganze Relation als Zielbegriff verwendet werden kann. Es dürfte aber gerade von Interesse sein, auch über einzelne Attribute zu lernen. Der zweite Nachteil ergibt sich aus der angestrebten Anwendung.

In dem logikorientierten Lernverfahren RDT wird die Beschreibung für einen Begriff gesucht. Als Beschreibungssprache dienen dabei die dem System bekannten Prädikate. Ist nun ein Attribut „frei" wählbar bei der Beschreibung des zu lernenden Begriffs, so erhöht sich das Kriterium *total* von RDT um den Faktor der Anzahl der Sortenelemente. Diese hätte zur Folge, daß das Kriterium *pred* als Akzeptanzkriterium von RDT nicht mehr handhabbar wäre. Man verliert dadurch ein Kriterium.

Regel auftreten können und Prädikate nicht doppelt dargestellt werden. Jeweils einen dieser Punkte erfüllen die anderen Repräsentationen nicht. In Tabelle 1 sind die angesprochenen Eigenschaften der Repräsentationen gegenübergestellt.

Allerdings reicht die Repräsentation III alleine nicht aus um z.B auch Verkettungen über Attribute, die keine Schlüsselattribute sind, zu lernen. An einem Beispiel will ich die Problematik veranschaulichen.

Gegeben sind folgende zwei Metaprädikate[3]:

m1(P1,P2,C):P1(S,T,U,V) & P2(S,T,V,X) → C(S,T,U)
m2(P1,P2,P3,C):P1(S,T,U,V) & P2(S,T,V,X) & P2(S,T,X,Z) → C(S,T,U)

RDT/DB würde m2 nicht als mögliche Hypothese betrachten, wenn m1 akzeptiert wurde, da m1 bzgl. der θ–Subsumption genereller ist als m2. Damit dieser Fall, nicht auftritt muß das Ende der Verkettung im Konklusionsbegriff gebunden sein. Eine Lösung ist, die Repräsentationen I und III zu kombinieren. Das Problem der ungebundenen Variablen tritt hierbei nicht mehr auf, da die Schlüsselattribute des Konklusionsbegriffes gebunden sind.

5 Umsetzung des Akzeptanzkriteriums in SQL – Anfragen

Um das Verfahren RDT auf relationale Datenbanken anwenden zu können, muß man geeignete SQL–Anfragen bestimmen, die die einzelnen Faktoren des Akzeptanzkriteriums berechnen.

Im folgenden sei H definiert wie in Abschnitt 3. Die Prädikate der Hypothese H sind von der Form $R\_a_x(PS, a_x)$, bzw. $R(PS)$, wobei R die Relationenbezeichnung (Tabellenname) in der relationalen Datenbank, a_x ein Attribut der Relation R und PS der Primärschlüssel der Relation R ist. Weiterhin sei $ps(Q)$ eine Funktion, die den Primärschlüssel des Prädikats Q bestimmt, $tabelle(Q)$ eine Funktion, die den zugehörigen Tabellennamen zum Prädikat Q bestimmt und $v(P)$ eine Funktion, die die Prämisen P in SQL–Bedingungen überführt.

Die Bestandteile des Akzeptanzkriteriums können wie folgt angesetzt werden:

- $\mid pos(H) \mid \Rightarrow$
  select count($ps(q)$) from $tabelle(q)$, $tabelle(P)$

[3]Die Schlüsselattribute sind unterstrichen.

aufgrund der zu erwartenden Größenordnung der einzelnen Sorten zu aufwendig wäre.

Stattdessen teste ich die Datentypverträglichkeit der gebundenen Attributvariablen. Eine Hypothese, in der eine Variable an zwei verschiedene Datentypen der Datenbank gebunden ist, kann kein positives Beispiel mehr abdecken.

Eine weitere Einschränkung des Hypothesenraums erreiche ich durch die Streichung redundanter Prädikate, ohne den θ-Subsumptionstest von RDT zu verwenden. Aufgrund der von Repräsentation der Datenbank als Prädikate und den Informationen über die Primärschlüssel der Datenbank kann man leicht verhindern, daß ein Prädikat mit gleicher Variablenbelegung in einer Regel zweimal instanziiert wird.

Neben den Unterschieden in der Hypothesenraumeinschränkung enthält RDT/DB einen zusätzlichen Parameter. Mittels dieses Parameters ist es dem Benutzer überlassen zu bestimmen, ob Hypothesen, die als zu speziell erkannt wurden, im Redundanztest berücksichtigt werden. Dann kann es passieren, daß Speziallisierungen der Hypothese noch getestet werden, obwohl die Hypothese als zu speziell bekannt ist. Das zusätzliche Testen der Hypothesen ist in vielen Anwendungen weniger rechenaufwendig als die Redundanztests (siehe Abschnitt 7).

6.2 Negative Beispiele in RDT/DB

In relationalen Datenbanken sind keine negativen Beispiele gegeben, diese können aber für das Lernen von großer Bedeutung sein, da ein negatives Beispiel oft stärker abgrenzt. Durch negative Beispiele vermeidet man es zu allgemeine Regeln zu lernen. An dieser Stelle habe ich ausgenutzt, daß RDT/DB als externes Tool in das System MOBAL mit eingebunden ist. Der Benutzer kann so über das System MOBAL negative Beispiele, die beim Lernen über die Datenbank berücksichtigt werden (siehe Abschnitt 5), eingeben.

6.3 Algorithmus

Bevor ich im nächsten Abschnitt die ersten experimentellen Tests mit RDT/DB vorstelle, möchte ich die Vorgehensweise von RDT/DB skizzieren.

white king x	white king y	white rook x	white rook y	black king x	black king y	ID
4	5	2	2	2	4	452224
...	...	...	...	...	...	...

Tabelle 2: illegal mit ID

x	y
1	1
1	2
...	...

Tabelle 3: adjacent

Dieser Sachbereich wurde von mir aufgrund seiner Überschaubarkeit und relativen Einfachheit ausgewählt. Weiterhin ist dieser Sachbereich ein oft verwendetes Testszenario im maschinellen Lernen, so daß man Vergleiche mit anderen Verfahren anstellen kann. So wurde der KRK–Sachbereich zu Vergleichtests von LINUS/FOIL [Lavrač und Džeroski, 1992], und RDT/FOIL [Lindner und Robers, 1994] verwendet. Aus der Sicht des Knowledge Discovery ist diese Anwendung allerdings nicht akzeptabel. Hier wird besonderer Wert auf die „reale"Anwendung gelegt. Um diesem Punkt gerecht zu werden habe ich ein Testszenario aus dem Bereich der Roboternavigation (siehe Abschnitt 7.2) für weitere experimentelle Tests mit RDT/DB ausgewählt.

7.1.2 Testaufbau für KRK

Der Sachbereich ist in der Datenbank durch die Relation illegal (siehe Tabelle 2) mit ca. 3500 Einträgen repräsentiert, wobei das Schachbrett in ein numerisches Koordinatensystem eingeteilt ist.

Weiterhin ist in der Datenbank als Hintergrundwissen die Relation adjacent (siehe Tabelle 3) enthalten. Die Relation adjacent ist vollinstanziiert.

Um mit RDT/DB lernen zu können wurden folgende Metaprädikate vorgegeben:

```
m1(P1,P2,P3,P4,C)    :P1(E,X1) & P2(E,Y1) & P3(E,X1) & P4(E,Y2)
                      ne(Y1,Y2) --> C(E).
m2(P1,P2,P3,P4,C)    :P1(E,X1) & P2(E,Y1) & P3(E,X1) & P4(E,Y1)
                      --> C(E).
```

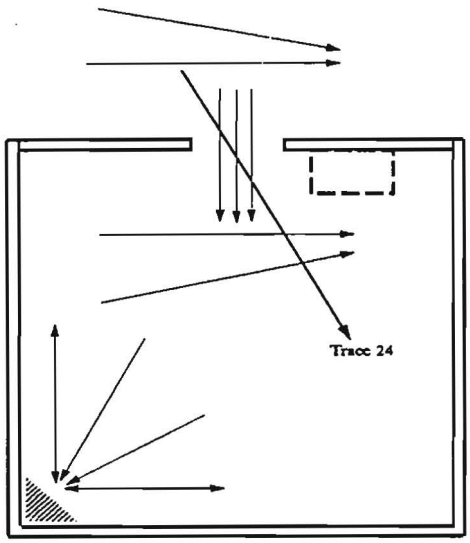

Abbildung 1: Roboterbewegung in einem Raum

Die *basic features* haben als Argumente die Tracenummer[5], die relative Orientierung des Sensors, die Sensorbezeichnung, Start- und Endzeit der Messung und den Gradienten. Jedes *basic feature* ist als Relation in der Datenbank repräsentiert, insgesamt gibt es 9 verschiedene *basic features*. Neben den *basic features* gibt es noch *sensor features*, die eine von einem Sensor gemessene Kantenkonstellation des Raums beschreibt.

Lernziel ist, Abfolgen von *basic features* zu lernen, die ein *sensor feature* beschreiben, d.h ein *sensor features* beschreiben das Verhalten eines Sensors vom Zeitpunkt t1 bis tn. Insgesamt gibt es 4 *sensor features*. Die Argumente der *sensor features* sind die Tracenummer, die Sensorbezeichnung, die Start- und Endzeit der Abfolge und die Bewegungsrichtung.

7.3 Testaufbau

Neben den *basic features* und den *sensor features* wurden fünf Metaprädikate für die Lernläufe verwendet, von denen ich exemplarisch drei angeben möchte:

```
m1(P1,P2,M,C)       :P1(T,S,X1,X2) & P2(T,S,X2,X3) & M(0)
                     --> C(T,S,X1,X3,0)
m2(P1,P2.P3,M,C)    :P1(T,S,X1,X2) & P2(T,S,X2,X3)
                     & P3(T,S,X3,X4) & M(0)
```

[5]Schlüsselattribute sind unterstrichen.

ich noch weitere Test mit RDT/DB, RDT und vielleicht einigen anderen Verfahren aus dem maschinellen Lernen durchführen. Dazu gehört dann auch eine Analyse der Vor- und Nachteile der Verfahren. Anfang August wird dann eine ausführliche Beschreibung über die Anwendung des Lernverfahrens RDT auf relationale Datenbanken als Diplomarbeit an der Universität Dortmund vorliegen.

Literatur

[Clark und Niblett, 1989] P. Clark und T. Niblett. The CN2 Induction Algorithm. *Machine Learning*, 3(4):261–284, 1989.

[Džeroski und Lavrač, 1993] S. Džeroski und N. Lavrač. Inductive Learning in Deductive Databases. *IEEE Transactions on Knowledge and Data Engineering*, 5(6):939–949, Dezember 1993.

[Frawley *et al.*, 1991] W. Frawley, G. Piatetsky-Shapiro und C. Matheus. Knowledge Discovery in Databases: An Overview. In G. Piatetsky-Shapiro und W. Frawley (Hrsg.), *Knowledge Discovery in Databases*, S.1–27, Cambridge, Mass., 1991. AAAI/MIT Press.

[Holsheimer und Siebes, 1993] M. Holsheimer und A. Siebes. Data Mininig: The Search for Knowledge in Databases. CS-R9406, CWI, P.O. Box 94079, 1090 GB Amsterdam, Netherland, 1993. ISSN 0169-118X.

[Kietz und Wrobel, 1992] Jörg-Uwe Kietz und Stefan Wrobel. Controlling the Complexity of Learning in Logic through Syntactic and Task-Oriented Models. In Stephen Muggleton (Hrsg.), *Inductive Logic Programming*, Kapitel 16, S. 335 - 360. Academic Press, London, 1992. Also available as Arbeitspapiere der GMD No. 503, 1991.

[Klingspor, 1991] Volker Klingspor. Abstraktion von Inferenzstrukturen in MOBAL. Diplomarbeit, Univ. Bonn, 1991.

[Klingspor, 1994] V. Klingspor. GRDT: Enhancing Model-Based Learning for Its Application in Robot Navigation. Forschungsbericht 518/94, Universität Dortmund, Fachbereich Informatik, 1994. ISSN 0933-6192.

[Lavrač und Džeroski, 1992] N. Lavrač und S. Džeroski. Inductive Learning of Relations from Noisy Data. In S. Muggleton (Hrsg.), *Inductive Logic Programming*, Kapitel 25, S. 495–516. Academic Press, San Diego, CA 92101, 1992.

RR-93-29
Armin Laux: Representing Belief in Multi-Agent Worlds viaTerminological Logics
35 pages

RR-93-30
Stephen P. Spackman, Elizabeth A. Hinkelman: Corporate Agents
14 pages

RR-93-31
Elizabeth A. Hinkelman, Stephen P. Spackman: Abductive Speech Act Recognition, Corporate Agents and the COSMA System
34 pages

RR-93-32
David R. Traum, Elizabeth A. Hinkelman: Conversation Acts in Task-Oriented Spoken Dialogue
28 pages

RR-93-33
Bernhard Nebel, Jana Koehler: Plan Reuse versus Plan Generation: A Theoretical and Empirical Analysis
33 pages

RR-93-34
Wolfgang Wahlster: Verbmobil Translation of Face-To-Face Dialogs
10 pages

RR-93-35
Harold Boley, François Bry, Ulrich Geske (Eds.): Neuere Entwicklungen der deklarativen KI-Programmierung — *Proceedings*
150 Seiten
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

RR-93-36
Michael M. Richter, Bernd Bachmann, Ansgar Bernardi, Christoph Klauck, Ralf Legleitner, Gabriele Schmidt: Von IDA bis IMCOD: Expertensysteme im CIM-Umfeld
13 Seiten

RR-93-38
Stephan Baumann: Document Recognition of Printed Scores and Transformation into MIDI
24 pages

RR-93-40
Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, Werner Nutt, Andrea Schaerf: Queries, Rules and Definitions as Epistemic Statements in Concept Languages
23 pages

RR-93-41
Winfried H. Graf: LAYLAB: A Constraint-Based Layout Manager for Multimedia Presentations
9 pages

RR-93-42
Hubert Comon, Ralf Treinen: The First-Order Theory of Lexicographic Path Orderings is Undecidable
9 pages

RR-93-43
M. Bauer, G. Paul: Logic-based Plan Recognition for Intelligent Help Systems
15 pages

RR-93-44
Martin Buchheit, Manfred A. Jeusfeld, Werner Nutt, Martin Staudt: Subsumption between Queries to Object-Oriented Databases
36 pages

RR-93-45
Rainer Hoch: On Virtual Partitioning of Large Dictionaries for Contextual Post-Processing to Improve Character Recognition
21 pages

RR-93-46
Philipp Hanschke: A Declarative Integration of Terminological, Constraint-based, Data-driven, and Goal-directed Reasoning
81 pages

RR-93-48
Franz Baader, Martin Buchheit, Bernhard Hollunder: Cardinality Restrictions on Concepts
20 pages

RR-94-01
Elisabeth André, Thomas Rist: Multimedia Presentations: The Support of Passive and Active Viewing
15 pages

RR-94-02
Elisabeth André, Thomas Rist: Von Textgeneratoren zu Intellimedia-Präsentationssystemen
22 Seiten

RR-94-03
Gert Smolka: A Calculus for Higher-Order Concurrent Constraint Programming with Deep Guards
34 pages

RR-94-05
Franz Schmalhofer, J.Stuart Aitken, Lyle E. Bourne jr.: Beyond the Knowledge Level: Descriptions of Rational Behavior for Sharing and Reuse
81 pages

RR-94-06
Dietmar Dengler: An Adaptive Deductive Planning System
17 pages

DFKI Documents

D-93-14
Manfred Meyer (Ed.): Constraint Processing – Proceedings of the International Workshop at CSAM'93, July 20-21, 1993
264 pages
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

D-93-15
Robert Laux:
Untersuchung maschineller Lernverfahren und heuristischer Methoden im Hinblick auf deren Kombination zur Unterstützung eines Chart-Parsers
86 Seiten

D-93-16
Bernd Bachmann, Ansgar Bernardi, Christoph Klauck, Gabriele Schmidt: Design & KI
74 Seiten

D-93-20
Bernhard Herbig:
Eine homogene Implementierungsebene für einen hybriden Wissensrepräsentationsformalismus
97 Seiten

D-93-21
Dennis Drollinger:
Intelligentes Backtracking in Inferenzsystemen am Beispiel Terminologischer Logiken
53 Seiten

D-93-22
Andreas Abecker:
Implementierung graphischer Benutzungsoberflächen mit Tcl/Tk und Common Lisp
44 Seiten

D-93-24
Brigitte Krenn, Martin Volk:
DiTo-Datenbank: Datendokumentation zu Funktionsverbgefügen und Relativsätzen
66 Seiten

D-93-25
Hans-Jürgen Bürckert, Werner Nutt (Eds.):
Modeling Epistemic Propositions
118 pages
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

D-93-26
Frank Peters: Unterstützung des Experten bei der Formalisierung von Textwissen
INFOCOM:
Eine interaktive Formalisierungskomponente
58 Seiten

D-93-27
Rolf Backofen, Hans-Ulrich Krieger, Stephen P. Spackman, Hans Uszkoreit (Eds.):
Report of theEAGLES Workshop on Implemented Formalisms at DFKI, Saarbrücken
110 pages

D-94-01
Josua Boon (Ed.):
DFKI-Publications: The First Four Years
1990 - 1993
75 pages

D-94-02
Markus Steffens: Wissenserhebung und Analyse zum Entwicklungsprozeß eines Druckbehälters aus Faserverbundstoff
90 pages

D-94-03
Franz Schmalhofer: Maschinelles Lernen: Eine kognitionswissenschaftliche Betrachtung
54 pages

D-94-04
Franz Schmalhofer, Ludger van Elst:
Entwicklung von Expertensystemen: Prototypen, Tiefenmodellierung und kooperative Wissensevolution
22 pages

D-94-06
Ulrich Buhrmann:
Erstellung einer deklarativen Wissensbasis über recyclingrelevante Materialien
117 pages

D-94-07
Claudia Wenzel, Rainer Hoch:
Eine Übersicht über Information Retrieval (IR) und NLP-Verfahren zur Klassifikation von Texten
25 Seiten

D-94-08
Harald Feibel: IGLOO 1.0 - Eine grafikunterstützte Beweisentwicklungsumgebung
58 Seiten

D-94-09
DFKI Wissenschaftlich-Technischer Jahresbericht 1993
145 Seiten

D-94-10
F. Baader, M. Lenzerini, W. Nutt, P. F. Patel-Schneider (Eds.): Working Notes of the 1994 International Workshop on Description Logics
118 pages
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

D-94-11
F. Baader, M. Buchheit, M. A. Jeusfeld, W. Nutt (Eds.):
Working Notes of the KI'94 Workshop: KRDB'94 - Reasoning about Structured Objects: Knowledge Representation Meets Databases
65 Seiten

D-94-12
Arthur Sehn, Serge Autexier (Hrsg.): Proceedings des Studentenprogramms der 18. Deutschen Jahrestagung für Künstliche Intelligenz KI-94
69 Seiten

Proceedings des Studentenprogramms der 18. Deutschen Jahrestagung für Künstliche Intelligenz KI-94

Arthur Sehn, Serge Autexier

D-94-12
Document