Abschlußbericht

DISCO
Dialogsystem für autonome kooperierende Agenten

BMFT Förderkennzeichen ITW 9002 0

Dipl.-Inform. Rolf Backofen
Dr. Stephan Busemann
Dipl.-Inform. Abdel Kader Diagne
Dr. Elizabeth A. Hinkelman
Dr. Walter Kasper
Dipl.-Inform. Bernd Kiefer
Dipl.-Inform. Hans-Ulrich Krieger
Klaus Netter, M.A.
Dipl.-Inform. Günter Neumann
Stephan Oepen, M.A.
Stephen Spackman, MCompSc
Prof. Dr. Hans Uszkoreit

Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI)
Stuhlsatzenhausweg 3, 66123 Saarbrücken
e-mail: {nachname}@dfki.uni-sb.de

30. November 1994

Inhaltsverzeichnis

1 Projektergebnisse und Projektablauf **1**

1.1 Aufgabenstellung und Ergebnisse . . . 1
1.2 Planung und Ablauf des Vorhabens . . . 3
1.2.1 Arbeitspakete und Ergebnisse . . . 3
1.2.2 Weitere Aktivitäten . . . 5
1.2.3 Personalia . . . 7

2 Die linguistische Kernmaschine **9**

2.1 Grammatikformalismus . . . 9
2.1.1 Einführung . . . 10
2.1.2 Motivation . . . 10
2.1.3 Der DISCO-Kernformalismus . . . 11
2.1.4 Der Merkmalsconstraintlöser UDiNe . . . 12
2.1.5 Intelligentes Backtracking . . . 14
2.1.6 *TDL*—Ein Überblick . . . 15
2.1.7 Typhierarchie . . . 16
2.1.8 Simplifikation und Memoisierung . . . 18
2.1.9 Typexpansion . . . 19
2.2 Theoretische Arbeiten im Bereich Formalismus . . . 20
2.3 Morphologie . . . 21
2.4 Grammatik . . . 22
2.4.1 Merkmalsgeometrie . . . 22
2.4.2 Architektur der Grammatik . . . 24
2.4.3 Linguistische Abdeckung der Grammatik . . . 31
2.5 Semantik . . . 33
2.5.1 Konzeption . . . 33
2.5.2 Semantikkonstruktion in der Grammatik . . . 34
2.5.3 NLL . . . 35
2.5.4 Syntax-Semantik-Schnittstelle . . . 36
2.5.5 Referenzauflösung und Diskursgedächtnis . . . 37
2.6 Parsing . . . 38
2.6.1 Möglichkeiten zur Parametrisierung . . . 39
2.6.2 Chart Display . . . 41

2.6.3 Anwendung im DISCO-System . . . 41
2.7 Generierung . . . 42
2.7.1 Der Algorithmus . . . 42
2.7.2 Das System SEREAL . . . 43
2.7.3 Verbesserungen des Algorithmus . . . 44
2.7.4 Die Laufzeitgrammatik des Generators . . . 45
2.7.5 Implementation . . . 45
2.7.6 Semantikdarstellung und Generierung: Probleme . . . 46
2.8 Lernen von Ableitungsmustern . . . 48

3 Werkzeuge **53**
3.1 Ein Editor für Merkmalsstrukturen: Fegramed . . . 53
3.2 Chart-Display . . . 56
3.3 Datenbank für Testsätze des Deutschen . . . 58
3.3.1 Motivation . . . 58
3.3.2 Realisierung . . . 59
3.3.3 Ergebnisse . . . 60
3.4 Arbeiten an SADAW . . . 61

4 Diskurs und Dialog **63**
4.1 Sprechakttheorie . . . 63
4.2 Integration mit der Grammatik . . . 64
4.3 Gruppen als Agenten . . . 66
4.3.1 Gruppenagenten und Propagierung von propositionalen Einstellungen 67
4.3.2 Teilnehmerintensive Sprechakte . . . 72
4.3.3 Verarbeitung eines N-Pfad-Sprechakts . . . 73
4.3.4 Diskussion . . . 76
4.4 Weitere dialog-unterstützende KI-Aktivitäten . . . 77
4.4.1 **PI SA** . . . 78
4.4.2 Temporale und lokale Schlußfolgerungen . . . 79
4.5 Arbeitspakete und Meilensteine . . . 80

5 Verteilte Terminplanung: Eine Anwendung der Kernmaschine **83**
5.1 Grundlagen . . . 83
5.2 Kooperation zwischen DISCO und AKA-MOD . . . 85
5.3 Übersicht über das COSMA-System . . . 85
5.3.1 Verteilte Terminplanung: Eine Beispielinteraktion . . . 89
5.3.2 Kopplung mit dem Anwendungssystem . . . 91
5.3.3 Template-Generierung . . . 92
5.3.4 Graphische Benutzerschnittstelle . . . 92
5.4 Erfahrungen mit COSMA . . . 93

6 Implementation **97**
6.1 Softwareverwaltung 97
6.2 Die Entwicklungsumgebung 99
6.3 Portierungen 103

Kapitel 1

Projektergebnisse und Projektablauf

Nach einem knappen Überblick über die wesentlichen Projektergebnisse werden organisatorische Einzelheiten des Projektablaufs, wissenschaftliche Kontakte und sonstige wichtige Ereignisse zusammengefaßt.

1.1 Aufgabenstellung und Ergebnisse

In der Entwicklung der Informationstechnologie wird die natürliche Sprache zu einem unentbehrlichen Medium für die Kommunikation zwischen Menschen und ihren autonomen maschinellen Partnern (Agenten). Im Projekt DISCO wurde ein Dialogsystem entwickelt, das maschinelle Dialogpartner in die Lage versetzt, mit ihren menschlichen Partnern in natürlicher Sprache zu kommunizieren. Während existierende Dialogsysteme zur Kommunikation zwischen der Maschine und einem einzigen menschlichen Benutzer gedacht sind, nimmt das DISCO-System an Dialogen zwischen mehr als zwei Teilnehmern teil. Die autonomen kooperierenden Agenten, für die das DISCO-System geschaffen wurde, sind KI-Softwaresysteme auf vernetzten Computern.

DlSCO führte Forschung sowohl in Computerlinguistik als auch in anderen Gebieten der KI durch. Frühere Arbeiten an natürlichsprachlichen Dialogsystemen konzentrierten sich entweder auf moderne Methoden für die linguistische Spezifikation und die Sprachverarbeitung oder auf fortgeschrittene Dialogverarbeitung. Im Unterschied dazu konnte DISCOerfolgreich Resultate aus beiden Gebieten integrieren. Als Ergebnis erhielt das DF-KI ein auf modernsten Technologien aufbauendes, umfangreiches, flexibles, erweiterbares natürlichsprachliches Kernsystem, das für eine Vielzahl von Anwendungsentwicklungen eingesetzt werden kann und bereits in mehreren anderen Projekten eingesetzt wird.

Die Forschung in DISCO wurde in zwei Phasen durchgeführt. Die erste Phase endete 1991. In ihr wurde die linguistische Kernmaschine entwickelt und implementiert, die sich gegenüber früheren Systemen durch eine besonders sorgfältige theoretische Fundierung auf der Grundlage deklarativer constraint-basierter Grammatiken auszeichnet. Diese Fundierung garantiert die Wiederverwendbarkeit und Erweiterbarkeit. Die Kernmaschine umfaßt:

- eine mächtige Beschreibungssprache und einen Inferenzmechanismus für getypte Merk-

malsstrukturen, TDL,

- einen Unifikator für Merkmalsstrukturen, der im Gegensatz zu allen Vorgängern auch disjunktive und negierte Strukturen verarbeitet,
- einen linguistischen Formalismus auf der Grundlage von Head-Driven Phrase Structure Grammars, einer der am weitesten fortgeschrittenen Grammatiktheorien,
- eine umfangreiche Kerngrammatik des Deutschen, in *TDL* geschrieben,
- verschiedene Werkzeuge für die Grammatikentwicklung (u.a. einen Editor für Merkmalsstrukturen),
- ein Modul für die Analyse und Generierung von Wortformen auf der Basis der Zwei-Ebenen-Morphologie,
- einen parametrisierbaren Chart-Parser und einen Funktor-gesteuerten Generator
- eine Methode zur Anwendung von Explanation Based Learning (EBL) zur effizienten Verarbeitung von Subsprachen
- eine flexible Systemarchitektur zur Definition unterschiedlicher Kontrollstrategien

Phase Zwei begann 1992 und erweiterte das Kernsystem um die Fähigkeit, Dialoge zu führen. Das Anwendungssystem COSMA wurde entworfen und implementiert, in dem die linguistischen Kernkomponenten und Komponenten aus anderen DFKI-Projekten getestet und verbessert werden konnten. COSMA übernimmt die Rolle eines Sekretariatsassistenten bei der Planung von Terminen mit mehreren Teilnehmern.

Folgende wesentliche Forschungsergebnisse wurden erreicht:

- die Repräsentation von Gruppen als kollektive Agenten,
- dynamisches Dialogwissen, z.B. gemeinsame Überzeugungen und Einstellungen der Dialogpartner,
- Benutzung dieses Wissens, um die Beziehungen zwischen sprachlichen Ausdrücken und Diskursreferenten aufzubauen sowie um die Interpretation mehrdeutiger Äußerungen zu unterstützen,
- verläßliche Sprechakte für mehrere Partner,
- eine Schnittstelle zur Planungskomponente,
- das Anwendungssystem COSMA (in Zusammenarbeit mit dem BMFT-Projekt AKAMOD).

Neben einem direkten Nachfolgeprojekt PARADICE zur Weiterentwicklung des Kernsystems gingen aus dem Projekt DISCO die folgenden auf Anwendungen gerichtete Projekte, bzw. Aktivitäten hervor.

- EFFENDI (Effiziente Formulierung von Dialogbeiträgen) wird von der Daimler Benz AG finanziert und hat als Ziel die Implementierung einer effizienten Realisierungskomponte. Das Projekt wird in Zusammenarbeit mit der Gruppe von Prof. Dr. Wahlster durchgeführt.

- TSNLP (Test Suites for Natural Language Processing) ist ein EG gefördertes LRE Projekt, das Methodologie und Aufbau von Testkorpora zum Thema hat. Partner in diesem Projekt sind Aerospatiale France (Suresnes), ISSCO (Genf) und die University of Essex.

- TAMIC (Transparent Access to Multiple Information for the Citizen) ist ein EG gefördertes MLAP Projekt, das die Möglichkeit zu natürlichsprachlichem Zugang zu heterogenen Informationssystemen untersucht. Partner in diesem Projekt sind Quinary SpA (Mailand), Cap Gemini Innovation (Boulogne) und IRST (Trento).

- SARDIC wird vom DFKI, der Siemens AG, München und dem Institut für Maschinelle Sprachverarbeitung (Stuttgart) gemeinsam finanziert und hat zum Ziel, aufbauend auf die Ressourcen des am SFB 100, Saarbrücken, entwickelten SADAW Lexikons, eine wiederverwendbare, theorieübergreifende lexikalische Datenbank für den Einsatz in der maschinellen Sprachverarbeitung zu entwickeln.

Das Grammatikentwicklungssystem wird an mehreren anderen F&E Institutionen verwendet.

1.2 Planung und Ablauf des Vorhabens

1.2.1 Arbeitspakete und Ergebnisse

Das Projekt DISCO wurde mit 10 Mitarbeiterstellen zum 01.04.1990 bewilligt. Die Projektlaufzeit sollte vier Jahre bis zum 31.12.1993 betragen. Der geplante Personalaufwand betrug somit 480 Personalmonate. Infolge von Kürzungen wurde das Projekt mit ca. 320 Personalmonaten durchgeführt, d.h. mit durchschnittlich zwei Dritteln der ursprünglich geplanten Stärke. Weitere 21 Personalmonate kamen durch eine kostenneutrale Verlängerung um drei Monate vom 01.01.94 bis zum 31.03.1994 hinzu. Außerdem konnten durch freie Mitarbeiterverträge Projektaufgaben bearbeitet werden.

Die ursprünglich geplante Ausrichtung des Projektes auf die Verbindung zwischen constraintbasierten Ansätzen in der Computerlinguistik und Wissensrepräsentations- und Inferenzmethoden der KI wurde infolge der Kürzungen in Abstimmung mit dem wissenschaftlichen Beirat des DFKI leicht geändert. Indem im Bereich Dialog und Pragmatik weniger Personalmonate eingesetzt wurden, lag der Schwerpunkt des Projekts auf der linguistischen Kernkomponente sowie auf der Beispielanwendung COSMA. Diese Anwendung ergab sich erst im Laufe des Projekts, so daß die dadurch entstehenden Arbeitspakete noch nicht im Antrag enthalten waren (vgl. Kapitel 5).

Da es ein ausdrückliches Ziel von DISCO war, uniforme Formalismen und möglichst hohe Verarbeitungsuniformität anzustreben, wurde das Projekt in die Teilgruppen *Formalismen und Schnittstellen, Linguistische Verarbeitung* und *Linguistische Wissensbasen* unterteilt. In der nachfolgenden Übersicht wird der Bezug zwischen den Teilgruppen mit ihren gemäß Antrag zugeordneten Arbeitspaketen und den in diesem Bericht beschriebenen Forschungsergebnissen hergestellt.

Formalismen und Schnittstellen

- Merkmalsunifikationsformalismus: Abschnitt 2.2
- Grammatikformalismus: Abschnitt 2.2
- Wissensrepräsentationsformalismus: Abschnitt 4.4
- Schnittstellen: Abschnitte 5.3.2, 5.3.4, 6.2
- Werkzeuge: Abschnitte 3.1, 3.2

Linguistische Verarbeitung

- Analyse: Abschnitt 2.6
- Generierung: Abschnitte 2.7, 5.3.3
- Morphologie: Abschnitt 2.3
- Subsprachen: Abschnitt 2.8
- Lexikonzugriff: Abschnitt 2.7

Linguistische Wissensbasen

•Lexikon:Abschnitte2.4,3.4•Syntax:Abschnitte2.4,3.3

- Semantik und Pragmatik: Abschnitt 2.5, Kapitel 4
- Subsprachen: Abschnitt 2.8
- Weltwissen: Abschnitt 4.4

Während der kostenneutralen Verlängerung wurden verschiedene Portierungen der Kernmaschine auf andere Plattformen fertiggestellt (Abschnitt 6.3).

1.2.2 Weitere Aktivitäten

Vom 22. bis 24. Juni 1990 wurde in Saarbrücken an der Universität des Saarlandes und am DFKI ein Grammar Engineering Workshop abgehalten. Hierzu wurden Vertreter aller wichtigen Grammatikentwicklungsprojekte in der Welt in den letzten 30 Jahren und insbesondere aller in Deutschland aktiven Gruppen eingeladen. Hauptthemen waren: Graphikwerkzeuge zur Veranschaulichung von Verarbeitungsprozessen und Wissensbasen, Testwerkzeuge zur Diagnostik und zur Evaluierung von Grammatiken, allgemeine Softwarewerkzeuge zur Unterstützung von Gruppenarbeit, sinnvolle Arbeitsaufteilung unter den Grammatikentwicklern, die Tauglichkeit bzw. Untauglichkeit von verschiedenen linguistischen Theorien und linguistischen Beschreibungen, und die Nützlichkeit von herkömmlichen sprachwissenschaftlichen Wissensquellen (wie Wörterbüchern und Konkordanzen). Der Workshop wurde von Hans Uszkoreit organisiert.

Im Dezember 1990 wurden Fernsehaufnahmen über das Projekt DISCO im Rahmen einer Sendung über das DFKI im SWF-3 Fernsehen gemacht.

Im August 1991 fand ein Workshop zum Thema "Deutsche Grammatik im Rahmen der HPSG" statt, an dem eine Reihe von internationalen Experten auf dem Gebiet der HPSG teilnahmen und Beiträge leisteten. Eine Auswahl dieser Papiere wurde in einem Sammelband veröffentlicht [116]. Der Workshop wurde von John Nerbonne, Klaus Netter und Carl Pollard organisiert.

Die Generaldirektion XIII der EG-Kommission nimmt den Merkmalseditor Fegramed in die EG-geförderte Grammatikentwicklungsumgebung ALEP auf.

Hans Uszkoreit wurde zum Mitglied des Management Boards der EG für die Expert Advisory Groups for Language Engineering Standards (EAGLES) berufen. Ihm wurde auch die Leitung einer dieser Expertengruppen ("Linguistic Formalisms") übertragen.

Durch die Einstellung von Elizabeth A. Hinkelman in DISCO wurde der Katalog für natürlichsprachliche Software (Natural Language Software Registry) ans DFKI gebracht. Frau Hinkelman betreut diesen Katalog, der Beschreibungen von Systemen enthält, die auf kommerzieller oder nichtkommerzieller Basis erhältlich sind. Die aktuelle Version umfaßt

- Systeme zur Sprachsignalverarbeitung, z.B. Computerized Speech Lab (Kay Electronics),
- morphologische Analysekomponenten, z.B. PC-KIMMO (Summer Institute for Linguistics),
- Parser, z.B. Alveytools (University of Edinburgh),
- Wissensrepräsentationssysteme, z.B. Rhet (University of Rochester),
- Multikomponentensysteme, z.B. ELU (ISSCO), PENMAN (ISI), Pundit (UNISYS), SNePS (SUNY Buffalo),
- verschiedene Anwendungsprogramme.

Der Katalog ist on-line erhältlich über anonymes FTP bei ftp.dfki.uni-sb.de und über E-Mail an registry@dfki.uni-sb.de. Er wird laufend auf dem neuesten Stand gehalten.

Prof. Carl Pollard (The Ohio State University), Mitentwickler von HPSG, der im DISCO-Projekt eingesetzten und zur Zeit wichtigsten Grammatiktheorie in der Computerlinguistik, war drei Wochen im Juli und August 1992 Gastwissenschaftler bei DISCO. Während dieser Zeit wurde ein Treffen über die mathematische Interpretation des HPSG-Formalismus veranstaltet, an dem als Gäste außer Pollard auch Dr. Drew Moshier (UCLA) und Dr. Paul King (Tübingen) teilnahmen. Die wichtigsten Themen während des Besuchs von Pollard waren

- die Weiterentwicklung des Lexikonmodells im Stile von Krieger und Nerbonne,
- die Einbindung der Semantik in HPSG und
- die Herausgabe eines Buches über Anwendungen der HPSG im Deutschen (mit Nerbonne und Netter [116]).

Bob Kasper (Ohio State University) war im Sommer 1992 zwei Wochen zu Besuch am DFKI und stellte in dieser Zeit Mitarbeitern von DISCO und WIP seinen Ansatz zur Übersetzung vom HPSG-Grammatikformalismus in TAG-ähnliche Strukturen vor. Eine Implementierung dieses Ansatzes wurde dann gemeinsam begonnen und wird fortgesetzt.

Stephan Busemann veranstaltete zusammen mit Karin Harbusch (Projekt WIP) am 16.10.92 einen DFKI-internen Workshop zum Thema "Natürlichsprachliche Systeme: Wiederverwendbarkeit und Modularität". Es wurden zehn Vorträge zum Themenbereich aus den mit Sprachverarbeitung befaßten DFKI-Projekten gehalten [31].

Das Kooperationsprojekt EFFENDI wurde von der Daimler-Benz AG und dem DFKI genehmigt mit einer Laufzeit von 01.06.93 bis 31.12.94. Die beteiligten Projekte seitens des DFKIs waren VERBMOBIL und DISCO. Seitens des DISCO-Projekts ist Günter Neumann der Ansprechpartner für EFFENDI.

Am DFKI wurde vom 28.02.-03.03.93 ein EAGLES (Expert Advisory Group on Linguistic Engineering Standards) Workshop über implementierte linguistische Formalismen durchgeführt. Die Tagung wurde von Mitarbeitern der Projekte DISCO und ASL/VERBMOBIL organisiert. Auf dem Workshop wurden 15 Systeme vorgeführt; darunter die derzeit am weitesten fortgeschrittenen in diesem Gebiet. Das TDL-System von DISCO wurde von den Gutachtern sehr gelobt und bekam ausgezeichnete Noten, sowohl für das eigentliche System als auch hinsichtlich der linguistischen Abdeckung der mit *TDL* geschriebenen Grammatik des Disco-Projektes.

Vom 01.10.93 bis 30.06.94 war Prof. K. Vijay-Shanker von der University of Delaware zu Gast am DFKI. Die Zusammenarbeit mit ihm erwies sich als sehr fruchtbar für DISCO, besonders hinsichtlich der Kompilation von HPSG in den TAG-Formalismus.

Portierungen des DISCO-Kernsystems wurden im März 1994 an folgende Institutionen ausgeliefert:

- Fachbereich Informatik der Universität des Saarlandes, SFB 314, Projekt PRACMA, B. Kipper

- Fachbereich Computerlinguistik der Universität des Saarlandes, Dr. Mats Wiren, außerdem Einsatz des Scanners für Korpus-Tagging
- IBM Deutschland, Heidelberg, Dr. Peter Bosch.
- Center for the Study of Language and Information (CSLI), Stanford Universität (USA), Prof. Ivan Sag. Das DISCO-System wurde von den Mitarbeitern und Doktoranden der HPSG-Gruppe am CSLI sowohl bezüglich seiner formalen Mächtigkeit als auch hinsichtlich seiner Effizienz und seiner leistungsfähigen Entwicklungsumgebung sehr positiv beurteilt.

Mitarbeiter des Projekts haben viele Workshops und Tutorien auf internationalen Konferenzen und Sommerschulen organisiert. Durch Mitarbeiter des Projekts wurden auch viele Lehrveranstaltungen an der Universität des Saarlandes abgehalten.

1.2.3 Personalia

Am 17.07.90 promovierte Stephan Busemann zum Dr. rer. nat. an der Universität des Saarlandes. Gutachter waren Wolfgang Wahlster und Hans Uszkoreit.

Harald Trost habilitierte sich an der Universität Wien. Er verließ das Projekt zum 31.10.91.

Stephan Busemann erhielt verschiedene Angebote, C3-Stellen im Bereich Computerlinguistik zu vertreten. Er vertrat eine C3-Stelle an der Universität Hamburg im Sommersemester 1992 und war vom 01.04.92 bis 31.07.92 nur nebenberuflich im Projekt tätig.

John Nerbonne erhielt einen Ruf auf eine C4-Stelle in Groningen (Professor für Computerlinguistik und Leiter der Alfa-Informatica). Er verließ das Projekt zum 31.01.93.

In DISCO haben neben den Autoren dieses Berichts, John Nerbonne und Harald Trost, auch zahlreiche freie Mitarbeiterinnen, Diplomandinnen und studentische Hilfskräfte zeitweise mitgearbeitet: Jan Alexandersson, Paola D'Allesandro, Thomas Fettig, Hannes Fischer, Ralph Flassig, Edmund Grimley Evans, Stefan Haas, Christoph Jung, Judith Klein, Karsten Konrad, Harald Löchert, Ingo Neis, Fred Oberhauser, Ulrich Schäfer, Oliver Scherf, Jörg Steffen, Michael Wein, Christoph Weyers.

Sekretariatsarbeiten wurden von Corinna Johanns und Astrid Thoenes übernommen.

Kapitel 2

Die linguistische Kernmaschine

In diesem Kapitel werden die Basistechniken der Repräsentation und Verarbeitung natürli-
cherSprachebeschrieben,diediewiederverwendbarelinguistischeKernmaschinevonDISCObilden.Angesichts
zu repräsentieren wurde besonderes Augenmerk auf die Grammatikentwicklungsumgebung gelegt. Die linguistische Kernmaschine stellt komfortable Methoden zur Entwicklung und Wartung constraintbasierter Grammatiken zur Verfügung. Daneben dient sie als Kernmodul in natürlichsprachlichen Systemen, indem grundlegende Verfahren zur Analyse und Generierung von Sprache zur Verfügung gestellt werden.

Abschnitt 2.2 beschreibt den allgemeinen Unifikations- und Inferenzmechanismus, mit dem in der Kernmaschine Grammatiken definiert und Sprache verarbeitet wird. Das sprachliche Wissen (Morphologie, Syntax und Semantik) wird in den Abschnitten 2.3, 2.4 und 2.5 beschrieben. Die Verarbeitungsverfahren (Parsing und Generierung) finden sich in den Abschnitten 2.6 und 2.7. Neuartige Verfahren, die Analyse durch Verwendung von zuvor geparsten und gelernten Ableitungen zu beschleunigen, werden in Abschnitt 2.8 dargestellt.

2.1 Grammatikformalismus

TDL [87, 89, 90, 91] ist eine getypte merkmalsbasierte Sprache und Inferenzsystem, das in Hinblick auf lexikalisierte Grammatiktheorien entwickelt wurde. *TDL* bietet die Möglichkeit, (potentiell rekursive) Typen mittels Typconstraints und Merkmalsconstraints über den Standardkonnektiven $\land$, $\lor$ und zu definieren, wobei die Typen selber in einer Typsubsumtionshierarchie angeordnet sind. *TDL* unterscheidet zwischen AVM-Typen (offene Welt) und Sorten-Typen (geschlossene Welt) und erlaubt die Deklaration von Partitionen und unverträglichen Typen. Die Arbeit mit partiellen als auch mit voll expandierten Typen ist möglich, sowohl zur Definitions- wie auch zur Laufzeit. *TDL* ist inkrementell, d.h. die Redefinition von Typen ist möglich als auch die Verwendung von Undefinierten Typen. Die effiziente Verarbeitung wird in *TDL* durch die Verwendung von vier spezialisierten Inferenzmodulen erreicht.

TDL basiert auf *UDiNe* [17], einem hochentwickelten Merkmalsconstraintlöser. *UDiNe*

stellt die wesentlichen fortgeschrittenen Hilfsmittel zur Verfügung, die in der Literatur beschrieben sind oder aber in praktischen Systemen verwendet werden, z.B. volle Disjunktion inkl. verteilte Disjunktionen, volle Negation als auch funktionale und relationale Constraints.

TDL und *UDiNe* zusammen stellen sowohl eine hochentwickelte Grammatikentwicklungsumgebung als auch ein getyptes Laufzeitsystem zur Verfügung, das zudem noch über verzögerte Typexpansion verfügt. *TDL,* und UDiNe sind über eine flexible Schnittstelle verbunden, die zudem Raum für Erweiterungen und Experimente läßt.

2.1.1 Einführung

Unifikationsgrammatiken haben sich in den letzten Jahren zum vorherrschenden Paradigma in der Verarbeitung natürlicher Sprache und der Computerlinguistik entwickelt. Ihr Erfolg resultiert aus der Tatsache, daß man die zugrundeliegenden Formalismen als monotone, hochabstrahierende Repräsentationssprachen für linguistisches Wissen ansehen kann, die zudem eine präzise mathematische Semantik besitzen. Die Hauptidee ist dabei, so viel wie möglich linguistisches Wissen mittels eines einzigen Datentyps darzustellen, den man i.a. *Merkmalsstruktur* nennt. Dies erlaubt es einem dann, verschiedenste Beschreibungsebenen zu integrieren, ohne auf Schnittstellenprobleme acht zu geben. Während die ersten Ansätze noch auf annotierten Phrasenstrukturregeln aufbauten (z.B. PATR-II), versuchen moderne Formalismen grammatisches Wissen, als auch Lexikoneinträge nur noch mit Hilfe von Merkmalsstrukturen darzustellen. Um dieses Ziel zu erreichen, wurde die Ausdruckskraft der ersten unifikationsbasierten Formalismen durch verschiedenste disjunktive Beschreibungsmittel erweitert. Später kamen andere Operationen hinzu, etwa Negation. Die wichtigste Erweiterung jedoch betraf die Einführung von Typen, die man in modernen Systemen wie TFS, CUF oder *TDL,* findet. Typen sind hierarchisch angeordnet, wie man es etwa von objektorientierten Programmiersprachen her kennt. Dies führt zu multiplen Vererbung in der Beschreibung linguistischer Einheiten. Rekursive Typen sind schließlich notwendig, um z.B. Phrasenstruktur oder relationale Constraints ausdrücken zu können.

2.1.2 Motivation

Moderne getypte unifikationsbasierte Grammatikformalismen unterscheiden sich von frühen ungetypten Formalismen dadurch, daß sie den Begriff des *Merkmalstyps* herausstellen. Typen können hierarchisch angeordnet werden, wobei ein Subtyp alle Informationen seiner Supertypen erbt und Unifikation hier die primäre informationsbildende Operation darstellt. Eine *Typdefinition* kann zuallererst als ein Repräsentant für einen komplexen Ausdruck angesehen werden, der aus Typconstraints und Merkmalsconstraints über den Konnektiven A (log. Und), V (log. Oder) und -, (log. Negation) besteht. Typen dienen hier als Abkürzungen für Lexikoneinträge, ID Regelschemata, als auch für universelle bzw. sprachspezifische Prinzipien. Neben der Verwendung von Typen als ein Mittel der Abkürzung—wie man es etwa von Makros her kennt—haben Typen aber noch weitere Vorteile zu bieten, die jedoch nicht mit Hilfe von Makros erzielt werden können:

- STRUKTURIERUNG VON WISSEN
  Die Verwendung von Typen, plus die Möglichkeit sie hierarchisch anordnen zu können, erlaubt auf modulare und saubere Weise linguistisches Wissen adäquat darzustellen.

- EFFIZIENTE VERARBEITUNG
  Bestimmte Typconstraints können in effizientere Darstellungen, wie etwa Bitvektoren, compiliert werden, wobei GLB (größte untere Schranke), LUB (kleinste obere Schranke) und $\leq$ (Typsubsumtion) Berechnungen mit Bitmanipulationen korrespondieren. Zusätzlich befreien Typen die ungetypte Unifikation von teuren Berechnungen durch die Möglichkeit, Unverträglichkeiten zwischen Typen zu deklarieren. Desweiteren erlaubt der Einsatz von Typnamen oder partiell expandierten Typen den Aufwand des Kopierens von Strukturen während der Verarbeitung zu minimieren. Dies ist aber nur möglich, wie im Fall von *TDL,* wenn das System einen flexiblen Mechanismus zur Typexpansion zur Verfügung stellt.

- TYPÜBERPRÜFUNG
  Typdefinitionen geben einem Linguisten die Möglichkeit, Attribute plus die zugehörigen Attributwerte zu spezifizieren. Typdefinitionen sind daher ein Mittel, inkonsistente Beschreibungen zu vermeiden. Typexpansion ist wieder nötig, um die globale Erfüllbarkeit einer Struktur zu testen.

- REKURSIVE TYPEN
  Gewisse Funktionen oder Relationen können mit Hilfe von rekursiven Typen einfach dargestellt werden. Innerhalb des Typdeduktionsparadigmas wird das kontextfreie Phrasenstrukturgerüst durch rekursive Typen ersetzt. Parametrisierbare verzögerte Typexpansion ist hier das Tor zur Welt der kontrollierten linguistischen Deduktion [155].

2.1.3 Der DISCO—Kernfo

Der Kernformalismus von DISCO besteht aus *TDL* und *UDiNe* [14]. *TDL* ist eine unifikationsbasierte Grammatikentwicklungsumgebung als auch ein Inferenzsystem (Laufzeitsystem), das lexikalisierte Grammatiken im Stil von FUG, CUG, LFG und HPSG unterstützt. Die Arbeit an *TDL* begann innerhalb des DISCO-Projektes am DFKI [88]. Die DISCO-Grammatik besteht z.Zt. aus mehr als 900 Typdefinitionen in *TDL* und ist die größte HPSG-Grammatik des Deutschen [121]. Innerhalb des DISCO-Systems kommt auf den Merkmalsconstraintlöser *UDiNe* die Hauptarbeit zu. Die typische Größe einer zu verarbeitenden Struktur beträgt etwa 1000 Knoten und 140 Koreferenzen (das entspricht etwa 185000 Knoten in einer PROLOG Baumnotation).

Beide Module kommunizieren über eine extrem flexible Schnittstelle. Diese Kommunikation spiegelt exakt die Vorgehensweise eines abstrakten Unifikationsalgorithmus wieder: Zwei getypte Merkmalsstrukturen können nur dann unifizieren, wenn ihre Typen definitiv verträglich sind. Dies wird vom Unifikator dadurch gehandhabt, daß *UDiNe* an *TDL* zwei getypte Merkmalsstrukturen übergibt, während *TDL* selber eine simplifizierte (gelöste)

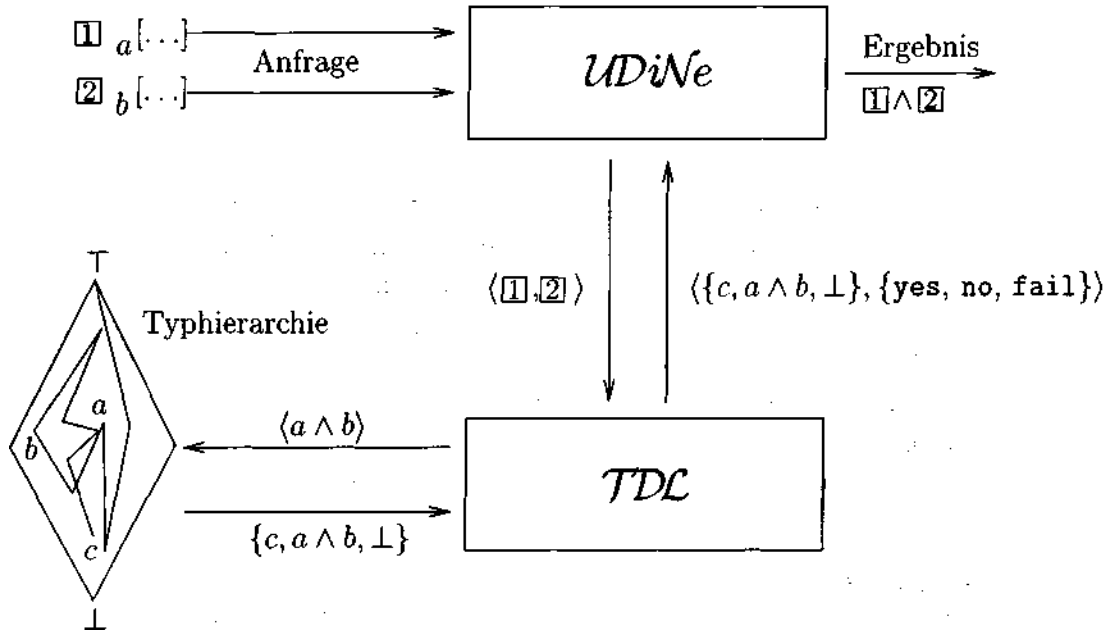


Abbildung 2.1: *Schnittstelle zwischen TDL und UDiNe. In Abhängigkeit von der Typhierarchie und dem Typ von* Gfl *und* [2] *gibt TDL entweder c zurück (c ist definitiv der GLB von a und b) oder liefert a^b im Falle einer offenen Welt bzw. _|_ (geschlossene Welt), falls es keinen ausgezeichneten Typ gibt, der der GLB von a und b ist. Zusätzlich bestimmt TDL, ob UDiNe Merkmalstermunifikation durchführen muß (yes) oder nicht (no), d.h. der Rückgabetyp enthält alle Information, um korrekt weiterarbeiten zu können* fail *signalisiert einen globalen Unifikationsfehler).*

Form zurückgibt (plus weitere Information; siehe Abb. 2.1). Die Motivation für die Trennung von Merkmals- und Typconstraints ist vielfältig. Zuallererst wird die Komplexität des Gesamtsystem verringert, wodurch die Architektur überschaubarer wird. Desweiteren wird die Performanz des Kernformalismus (und damit auch des Gesamtsystems) durch diese Maßnahme gesteigert, da jedes Modul nur speziell für eine sehr restriktive Domäne konzipiert wurde. Zudem erlaubt diese Technik auch andere Unifikatoren und Typsysteme anzuschließen.

2.1.4 Der Merkmalsconstraintlöser *UDiNe*

UDiNe ist ein moderner Constraintlöser für eine Merkmalssprache, die als Ausdrucksmittel Konjunktion, Disjunktion, verteilte Disjunktionen, negative Koreferenzen und volle Negation zuläßt. Es ist der erste (und unseres Wissens nach einzige) implementierte Merkmalsconstraintlöser in dem sowohl verteilte Disjunktionen als auch volle Negation zulässig sind. Desweiteren ist eine Erweiterung von *UDiNe* um relationale Constraints implementiert worden.

UDiNe arbeitet auf einer internen Repräsentation von Merkmalsstrukturen, in der Koreferenzen durch Strukturgleichheit repräsentiert werden. Für die Übersetzung der internen Repräsentation in eine (gut lesbare) externe Repräsentation und umgekehrt stehen verschieden Input/Output-Funktionen zur Verfügung. Desweiteren wurde im Rahmen von DISCO ein fensterbasierter Merkmalseditor (FEGRAMED) implementiert, der es erlaubt, Merkmalsstrukturen zu definieren, zu drucken und zu editieren.

Während der Übersetzung der externen Repräsentation in die interne werden verschiedene Normalisierungen durchgeführt und syntaktische Fehler erkannt. Einer der Normalisierungsschritte betrifft volle Negation. Wir verwenden die Methode von Gert Smolka [149], welche eine existentielle Quantifikation der Variablen innerhalb des Skopus der Negation einführt. Damit kann dann volle Negation auf Disjunktion, negative Koreferenzen und negative Atome/Typen zurückgeführt werden.

UDiNe verwendet verteilte Disjunktionen [8, 12, 13, 40, 39, 96] nicht nur in internen Verarbeitungschritten zur Effizienssteigerung. Sie sind auch Teil der Eingabesyntax, was eine sehr kompakte Darstellung der linguistischen Daten erlaubt. Im Gegensatz zu anderen Systemen, die verteilte Disjunktionen verwenden, ist die Anzahl der Alternativen nicht auf zwei beschränkt (weder in der Eingabe noch in der Verarbeitung). Dies reduziert erheblich die Größe der Eingabedaten.

Der Merkmalsconstraintlöser U*DiNe* ist so aufgebaut, daß er mit verschiedenen Typsystemen kombiniert werden kann. Die Unifikation von Merkmalsstrukturen erfolgt destruktiv, wobei die Methode von Hassan Ait-Kaci [1] verwendet wird. Diese Methode verzögert das Kopieren von Unterstrukturen, es werden nur diejenigen Strukturen kopiert, die während der Unifikation auch tatsächlich bearbeitet werden. Für nicht-destruktive Unifikation stellt *UDiNe* Kopierfunktionen zur Verfügung. *Keime* ist die Basismaschine des Projektes DISCO und wurde dort erfolgreich für verschieden Aufgaben eingesetzt, so zum Beispiel für Parsing, Generierung, erweiterte Zwei-Stufen Morphologie und Verarbeitung oberflächennaher Sprechaktrepräsentation.

Die Funktionalität von *UDiNe* wird durch eine Anzahl von Hilfsfunktionen vervollständigt, die Voraussetzungen für die Kopplung von *UDiNe* mit anderen Systemkomponenten bilden. So ist es möglich, inkonsistente Alternativen zu entfernen, Merkmalsstrukturen zu simplifizieren, Subterme zu extrahieren und funktionale Constraints zu definieren und auszuwerten. Es existiert eine allgemeine Besuchsfunktion, die es erlaubt, benutzerspezifische Erweiterungen zu implementieren. Diese Funktion macht die interne Repräsentation der Merkmalsstrukturen für das Anwenderprogramm transparent (was bei Verwendung verteilter Disjunktionen besonders wichtig ist, da hierbei die auch eine „verteilte" Repräsentation der Merkmalsstrukturen eingesetzt wird). So ist zum Beispiel die *TDL*-Typexpansion unter Verwendung diese Funktion als benutzerspezifische Erweiterung geschrieben. Eine weitere Hilfsfunktion erstellt die disjunktive Normalform einer Merkmalsstruktur. Diese Werkzeug wird für Systemkomponenten benötigt, die nicht mit verteilten Disjunktionen umgehen können. Man kann in dieser Funktion optional angeben, ob verteilten Disjunktionen bis zu ihrem gemeinsamen Präfix oder zur vollen disjunktiven Normalform expandiert werden sollen (letzteres entspricht der Expansion bis zum leeren Pfad). In beiden Fällen enthält die externen Repräsentation des Resultats keine verteilten Disjunktionen. Die erste Option

liefert im allgemeinen eine kleinere Struktur als Resultat, wogegen die zweite ein schnelleres Laufzeitverhalten besitzt.

Desweiteren ist eine Debugversion von *UDiNe* implementiert worden, die es erlaubt, Unifikation zu tracen und somit die Ursache für fehlschlagende Unifikationen zu finden. Dies ist besonders beim Debuggen der Grammatik wichtig.

Der nächste Abschnitt beschreibt eine prototypische Erweiterung von *UDiNe*, die eine flexible Kontrolle bei der Auswertung von verteilten Disjunktionen erlaubt.

2.1.5 Intelligentes Backtracking

Uszkoreit führte in [155] eine neue Strategie für die Verarbeitung linguistischem Wissen ein, die *kontrollierte linguistische Deduktion* genannt wird. Sie sieht sowohl eine Kontrolle der Auswertung von konjunktiver als auch von disjunktiver Information vor. Konjunktive verknüpfte Merkmalsconstraints werden dabei in der Reihenfolge der Fehlerwahrscheinlichkeit ausgewertet. Bei disjunktive verknüpften Constraints ist Erfolgswahrscheinlichkeit der Alternativen das Selektionskriterium. Diejenige Alternative mit der höchsten Erfolgswahrscheinlichkeit wird solange verwendet, bis ein Fehler bei der Unifikation auftritt. In diesem Fall wird die nächste beste Alternative ausgewählt. Uszkoreit beschreibt neben komplexeren auch eine Strategie, die statische Werte für die Erfolgswahrscheinlichkeit von Disjunktionsalternativen verwendet. Im folgenden werden wir diese Strategie *intelligentes Backtracking* nennen, da die Hauptaufgabe bei der Implementation dieser Strategie das gezielte Zurücksetzen und Nachführen von Unifikationsschritten ist. Ähnliche Probleme sind auch bei dem aus PROLOG bekannten intelligentem Backtracking zu lösen. Wegen dieser Ähnlichkeit können wir die folgenden Kriterien aufstellen, die ein Unifikator mit intelligentem Backtracking erfüllen muß:

- UNABHÄNGIGKEIT
  Das Backtracking muß unabhängig von der Berechnungshistorie sein, d. h. das Backtracking darf nicht nur auf die letzte verarbeitete Disjunktion beschränkt sein.

- KONFLIKTERKENNUNG
  Der Unifikator muß die an einen Unifikationsfehler beteiligten Disjunktionen bestimmen können. Es ist unerläßlich, daß die Menge der Kandidaten für das Backtracking so stark wie möglich eingeschränkt wird.

- KONFLIKTDEDUKTION
  Die Information über konfliktäre Kombinationen von Disjunktionen aus verschiedenen Unifikationen muß kombiniert werden, damit weitere konfliktäre Kombination berechnet werden können.

- VOLLSTÄNDIGKEIT
  Falls konsistente Kombinationen von Disjunktionsalternativen existieren, müssen diese auch erkannt werden.

Unifikatoren wie *Keime,* die verteilte Disjunktionen verwenden, sind vielversprechende Kandidaten für die Erweiterung um intelligentes Backtracking, da die meisten der obengenannten Konzepte bereits für die verteilten Disjunktion realisiert werden müssen. Dabei spielt das Konzept des Kontextes, das allen Implementation von verteilten Disjunktionen gemeinsam ist, eine zentrale Rolle. Ein *Kontext* ist eine Funktion, die Disjunktionen auf entsprechende Alternativen abbildet. Jeder Knoten einer Merkmalstruktur hat einen eindeutigen Kontext, der beschreibt, in welchen Disjunktionen und unter welchen Alternativen dieser Disjunktionen der Knoten gefunden werden kann. Falls eine Unifikation fehlschlägt, so wird der Kontext der an der fehlschlagenden Unifikation beteiligten Knoten *inkonsistenter Kontext* genannt. Diese inkonsistenten Kontexte werden gespeichert werden, um minimale inkonsistente Kontexte zu berechnen und globale Inkonsistenten zu erkennen. Die Berechnung von minimalen inkonsistenten Kontexten entspricht der Konfliktdeduktion. Der Test auf globale Inkonsistenz kann für die Sicherstellung der Vollständigkeit verwendet werden.

Wir haben eine prototypische Erweiterung von *UDiNe* um intelligentes Backtracking implementiert, die folgendermaßen arbeitet. Falls während der Unifikation eine Disjunktion bearbeitet wird, so wird die Alternative mit der höchsten Erfolgswahrscheinlichkeit ausgewählt. Die folgenden Unifikation werden nur mit dieser Alternative durchgeführt. Falls eine Unifikation fehlschlägt, so werden die beteiligten Disjunktionen anhand des inkonsistenten Kontextes bestimmt. Danach wird eine der beteiligten Disjunktionen für das Backtracking ausgewählt. Es gibt hier nun zwei Möglichkeiten: (1) die Disjunktion wird anhand der statischen Erfolgswahrscheinlichkeiten für die Alternativen ausgewählt und (2) der Unifikator ruft ein Benutzerprogramm auf, das die Disjunktion auswählt. Es ist daran gedacht, die zweite Möglichkeit zur Implementation von komplexeren Kontrollmethoden zu verwenden. Z.B. kann man dem Benutzerprogramm auch die Definitionen der Disjunktionen und die konjunktiven Merkmalsstrukturen, mit denen die Disjunktionen unifiziert wurde, aufrufen, damit man bessere Selektionskriterien erhält.

Während des Backtrackings der ausgewählten Disjunktion werden alle Unifikation mit der zuvor gewählten Alternative zurückgesetzt. Wir haben die existierende Rücksetzroutine derart verändert, daß ein lokales Rücksetzten ermöglicht wird. Danach werden die zurückgesetzten Unifikationen mit der neuen Alternative nachgeführt. Der Algorithmus garantiert, das die Unifikation auf die Substrukturen unterhalb der ausgewählten Disjunktion beschränkt ist.

2.1.6 *TDL*—Ein Überblick

TDL unterstützt Typdefinitionen, die aus Typ- und Merkmalsconstraints mit Hilfe der Operatoren $\wedge$, $\vee$, $\neg$ und $\oplus$ (log. Xor) gebildet werden. Diese Operatoren sind dahingehend generalisiert, daß sie Merkmalsbeschreibungen, Koreferenzen (log. Variablen) als auch Typen verknüpfen können. *TDL* unterscheidet zwischen *AVM-Typen* (Schließen in einer offenen Welt), *Sorten-Typen* (geschlossene Welt), *Built-In-Datentypen* (vom zugrundeliegenden Wirtssystem) und *Atomen.* Fragt man nach dem GLB von zwei AVM-Typen *a* und 6, die keinen gemeinsamen Subtyp besitzen, so gibt *TDL* im Fall einer offenen Welt immer aA6

und nicht_l_zurück. Der entgegengesetzte Fall gilt für Sorten-Typen. Zusätzlich unterscheiden sich Sorten-Typen von AVM-Typen dahingehend, daß sie nicht mehr weiter strukturiert sind, was auch für die Atome gilt. Desweiteren bietet *TDL* die Möglichkeit, exhaustive und disjunkte Partitionen von Typen zu deklarieren. So drückt etwa *sign = word® phrase* die Tatsache aus, daß (i) es keine anderen Subtypen von *sign* gibt als *word* oder *phrase,* (ii) die Mengen der Objekte, die durch *phrase* und *word* denotiert werden, disjunkt sind, und (iii) die Disjunktion von *phrase* und *word* (während der Verarbeitung) durch *sign* ersetzt werden kann.

TDL erlaubt es einem Grammatikschreiber parametrisierbare Templates (Makros) zu definieren und zu benutzen. Das Schreiben von Lexikoneinträgen (Instanzen) wird durch einen speziellen Mechanismus unterstützt. Typdefinitionen, sowie andere Eingaben werden durch einen von ZEBU generierten LALR(l)-Parser verarbeitet, der einem Benutzer eine intuitive Hochsprache zur Formulierung seines Wissens zu Verfügung stellt.

Im folgenden werden wir nun die Inferenzmodule von *TDL* kurz vorstellen und ihre besonderen Eigenschaften herausstellen. Eine globale Sicht auf die Architektur von *TDL* gibt Abb. 2.2.

2.1.7 Typhierarchie

Die Implementierung der Typhierarchie basiert auf der von Ait-Kaci vorgeschlagenen Bitvektorcodierungstechnik für partiellen Ordnungen (eine Bit-Und/-Oder Operation korrespondiert mit einer LUB-/GLB-Berechnung). Ait-Kacis Methode wurde um offenes Schließen über AVM-Typen dahingehend erweitert, das nun potentielle GLB/LUB-Kandidaten mit Hilfe einer Graphensuche über der Typhierarchie verifiziert werden müssen (diese Information kann jedoch vorcompiliert werden). GLB, LUB und < Berechnungen haben innerhalb dieses Rahmens die angenehme Eigenschaft, daß sie eine Zeitkomplexität von $O(n)$ besitzen, wobei n die Anzahl der Typen darstellt (die zur Laufzeit konstant ist). In Abhängigkeit von der Codierungsmethode benötigt die Typhierarchie $O(n \log n)$ ("compact encoding") bzw. $O(n^2)$ Bits ("transitive closure encoding").

Der Codierungsalgorithmus wurde auch hingehend der Redefinition von Typen erweitert, eine wichtige Eigenschaft eines inkrementellen Grammatik-/Lexikonentwicklungssystems. Einen Typ rezudefinieren bedeutet hier nicht nur lokale Änderungen an diesem Typ durchzuführen. Vielmehr müssen alle "Abhängigen" dieses Typs redefiniert werden— alle Subtypen im Falle eine konjunktiven Typdefinition bzw. alle Alternativen falls eine disjunktive Spezifikation vorliegt, plus, in beiden Fällen die Typen, die diese Typen in ihrer Definition erwähnen. Die abhängigen Typen eines Typs t können graphentheoretisch mittels der strengen Zusammenhangskomponenten von t bzgl. der Abhängigkeitsrelation charakterisiert werden.

Konjunktive (z.B. $x := y \wedge z$) und disjunktive Typspezifikationen (z.B. $x' := y' \text{ A } z'$) werden unterschiedlich in die Typhierarchie eingetragen: x erbt von seinen Supertypen y und z, während sich x' durch seine Alternativen y' und z' definiert. Diese Unterscheidung

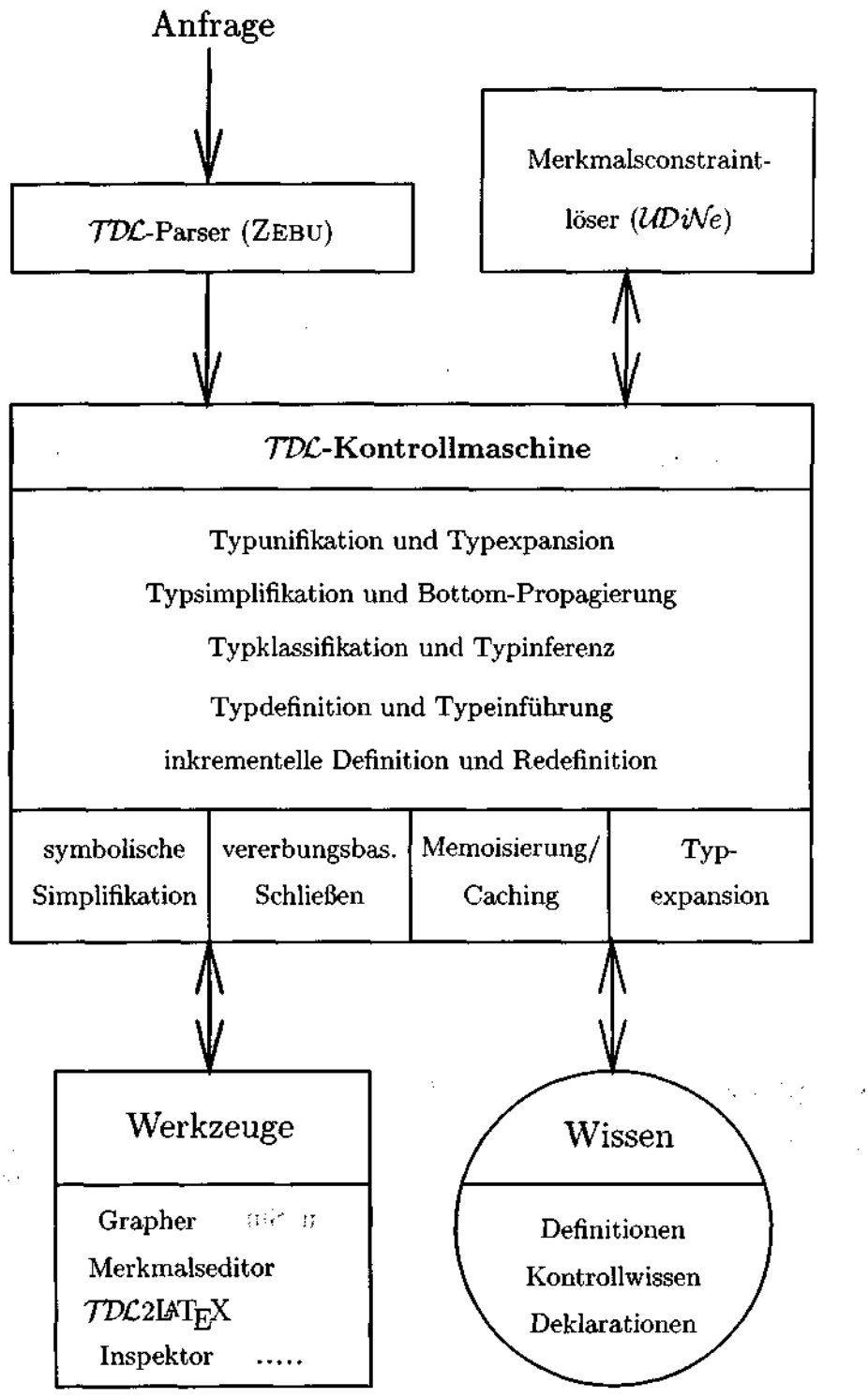


Abbildung 2.2: **Die Architektur von TDL**. Die *Kontrollmaschine von TDL* wird entweder *vom Unifikator UDiNe oder vom Benutzer während der Definitionszeit (etwa beim Konsistenzcheck des Lexikons und der Grammatik) bzw. zur Laufzeit (durch die Typexpansion) aufgerufen.*

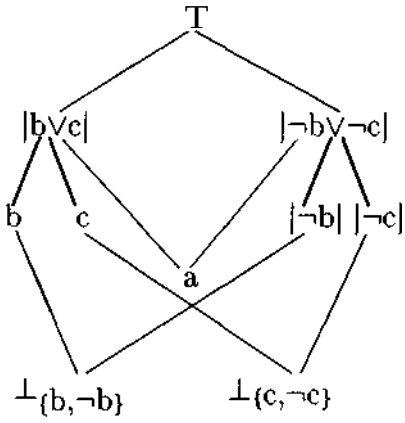


Abbildung 2.3: *Die Dekomposition von a := b © c erfolgt mit Hilfe der disjunktiven Zwischentypen \b* V c und |-b V *-c, von denen a erbt.*

wird in der Typhierarchie durch verschiedene Arten von Kanten dargestellt (siehe Abb. 2.3). Beide Formen drücken jedoch stets Subsumtion aus.

TDL dekomponiert komplexe Ausdrücke über A, V und -i unter Verwendung von (synthetischen) Zwischentypen, so daß der resultierende Ausdruck entweder eine pure Konjunktion oder Disjunktion von Typausdrücken ist. Die gleiche Technik wird bei der Verwendung von ® angewendet . ® wird in A, V und -> sowie zusätzlichen Zwischentypen entfaltet. Für jeden negativen Typ ->*t* führt *TDL* ein neues Typsymbol -<*t*\ ein, dessen Definition ->*t* ist und erklärt \->*t*\ unverträglich mit *t.* Unverträgliche Typen führen zudem spezielle Bottomsymbole ein (vgl. Abb. 2.3), die jedoch in der zugrundeliegenden Logik identifiziert werden. Diese Bottomsymbole werden zur Laufzeit durch einen Mechanismus, den wir Bottompropagierung nennen, abwärts transportiert.

2.1.8 Simplifikation und Memoisierung

Der Simplifikator operiert auf beliebigen *TDL*-Ausdrücken. Simplifikation wird sowohl zur Definitionszeit als auch zur Laufzeit durchgeführt, wenn getypte Unifikation stattfindet (Abb. 2.1). Die Hauptaufgabe der symbolischen Simplifikation besteht darin, unnötige Merkmalstermunifikationen und Anfragen an die Typhierarchie durch Anwendung von syntaktischen Reduktionsregeln zu verhindern.

Die Simplifikationsschemata sind aus der Aussagenlogik wohl bekannt (De Morgan, Idempotenz, Absorption etc.) und lassen sich auf unsere Situation leicht anpassen. Sie sind in COMMON LISP "fest verdrahtet", um die Verarbeitung zu beschleunigen. Formal gesehen kann Typsimplifikation in *TDL* als ein Termersetzungssystem charakterisiert werden. Konfluenz und Terminierung wird durch Erzwingung einer verallgemeinerten lexikograpischen Normalform auf Termen erzielt (entweder konjunktive oder disjunktive Normalform). Diese Ordnung hat zudem die angenehme Eigenschaft, daß das Kommutativitätsschema

nicht mehr angewendet werden muß, welches i.a. teuer ist und zu Terminierungsproblemen führt—es gibt nur einen Stellvertreter (d.h. eine Äquivalenzklasse) für die Permutationen einer Formel.

Aus diesem Grunde ist auch das Caching (Memoisierung) von Formeln billig und wird in *TDL* extensiv genutzt. Die Idee dabei ist es, das Ergebnis teurer Berechnungen abzuspeichern, um später darauf Zugriff zu haben. Die verallgemeinerte Normalform erlaubt es uns dann, nur einen Stellvertreter für eine ganze Klasse von komplexen Formeln betrachten zu müssen. Zudem erlaubt die Memoisierung auch, neue Typen zu "lernen", indem man die Frequenz der Zugriffe verzeichnet und nach einem Trainingslauf neue Typen für komplexe Ausdrücke einführt.

2.1.9 Typexpansion

Wie wir in Abschnitt 2.1.2 anmerkten, erlauben es uns Typen, über ihren Namen auf komplexe Strukturen zugreifen zu können. Die Rekonstruktion der mit dem Typ assoziierten Constraints verlangt eine komplexe Operation, die wir *Typexpansion* nennen wollen.

Die in TDL implementierte Typexpansion ist im Kontext des DISCO-Systems vielseitig verwendbar. So wird sie etwa als stand-alone Modul zum Konsistenzcheck des Lexikons und der Grammatik eingesetzt. Während der Unifikation kann sie (falls gewünscht) von *UDiNe* aufgerufen werden, falls eine der beiden Strukturen nur partiell expandiert sein sollte. Ebenso kann der DISCO-Chartparser die Typexpansion an wohldefinierten Stellen aufrufen (typischerweise maximale Projektionen), um die Konsistenz "unsicherer" Kanten festzustellen. Die Idee hierbei ist, mit partiell expandierten Strukturen zu arbeiten und sie nur von Zeit zu Zeit explizit zu machen, um ihre Konsistenz zu testen. Danach wird mit den kleinen Strukturen weitergerechnet und die expandierten Strukturen für den späteren Gebrauch verzeichnet. Natürlich kann Typexpansion auch im Typdeduktionsparadigma eingesetzt werden.

Der Typexpansionsmechanismus kann auf vielfältigste Weise gesteuert werden, etwa durch die Verwendung der (Miß-)Erfolgswahrscheinlichkeiten der Constraints (s. Abschnitt 2.1.5). Daneben gibt es aber auch noch andere Möglichkeiten, die Typexpansion zu beeinflussen, so etwa durch die Festlegung der Expansionstiefe oder durch die Angabe von Pfaden unter denen expandiert werden soll. Zudem ist es möglich nur Strukturen zu betrachten, die von einem gewissen Typ subsumiert werden.

Wie auch bei der Memoisierung ist es möglich, (partiell) expandierte Strukturen unter einem Index abzuspeichern, um sie später aus Effizienzgründen wieder nutzen zu können. Ein anderes Verfahren Effizienz zu gewinnen, stellt die Auswahl der globalen Expansionsstrategie dar. So kann der Benutzer etwa die an den Pfaden vermerkten Typen vor den Typen auf dem Top-Level expandieren lassen. Diese Methode ist aber grammatischspezifisch, da sie stark von der Filterwirkung der korrespondierenden Typhierarchie abhängt. Beide Verfahren beschleunigen die Typexpansion drastisch. Um den Suchraum während der Expansion zu beschneiden, werden Strukturen grundsätzlich markiert, ob sie voll expandiert sind oder nicht.

Da die Erfüllbarkeit von Merkmalsbeschreibungen mit rekursiven Typen i.a. unentscheidbar ist, haben wir in *TDL* zwei verschiedene Expansionsalgorithmen implementiert: ein vollständigen Verfahren, das möglicherweise nicht terminiert und ein nicht vollständiges, das natürlich in gewissen Fällen nicht korrekt ist. Das letztere Verfahren basiert auf dem Begriff der *Resolviertheit* und stellt eine Variante der verzögerten Typexpansion dar. Die fundamentale Idee dabei ist, nur die Constraints explizit zu machen, die wirklich neu sind. Anders ausgedrückt: solange keine (globale oder lokale) Kontrollinformation verfügbar ist, die die Expansion steuert, werden rekursive Typen unter allen ihren Pfaden expandiert (lokale plus ererbte), bis man einen Punkt erreicht, wo die Information mit der in einem Präfixpfad übereinstimmt. In beiden Verfahren kann jedoch immer die Expansionstiefe angegeben werden, um eine Terminierung zu erzwingen.

2.2 Theoretische Arbeiten im Bereich Formalismus

Es wurden wesentliche Beiträge zu dem Gebiet der Merkmalslogik geleistet. Zum einen wurde in Zusammenarbeit mit Gert Smolka eine vollständige Axiomatisierung der Theorie der Merkmalslogik über dem Standardmodell der Merkmalsgraphen aufgestellt [15]. Damit kann man die Gültigkeit von beliebige Termen aus der Merkmalssprache im Standardmodell entscheiden. Die zuvor bekannten Algorithmen waren auf das existentielle Fragment beschränkt, das nur eine kleinen Teil dieser Theorie darstellt. Bisher war keine vollständige und entscheidbare Theorie für Merkmalsterme bekannt. Der Algorithmus könnte zum Beispiel verwendet werden, um allgemeine Eigenschaften von Grammatiken zu bestimmen, wie die Subsumtion von verschiedenen Grammatikregeln. Da Grammatikregel auch Negation enthalten können, gab es bisher dafür keinen Entscheidungsalgorithmus.

Desweiteren wurde das offene Problem gelöst, ob Konjunktionen von sogenannten „functional uncertainty" Constraints erfüllbar sind. Diese Constraints sind Verallgemeinerungen von Merkmalsconstraints, in denen nicht ein einzelnes Merkmal, sondern eine Menge von Merkmalsketten verwendet werden, die durch einen regulären Ausdruck beschrieben sind. Diese Konzept wurde von [63] und [61] im Rahmen des Grammatikformalismus LFG [62] eingeführt, um sogenannte „long-distance dependencies" zu lösen. Ein detaillierte Beschreibung kann in [63] gefunden werden. Weitere Anwendungen sind in [71] beschrieben. Kaplan und Maxwell [61] haben ein partielles Resultat für diese Problem gefunden. Ihr Algorithmus beschränkt sich aber auf nicht-zirkuläre Strukturen. Diese Bedingung kann aber nicht im allgemeinen aufrecht gehalten werden. Ein negatives Resultat wurde in [7] angegeben, wo gezeigt wurde, das das Problems unentscheidbar ist, falls man Negation hinzufügt. Das Problem für beliebige Konjunktionen von sogenannten functional uncertainty Constraints war aber immer noch offen.

Wir haben in [11] gezeigt, daß diese Problem entscheidbar ist, und einen Entscheidungsalgorithmus angegeben. Der Entscheidungsalgorithmus ist auch eine Optimierung des Algorithmus von [61] im nicht-zirkulären Fall, da er eine flexiblere Kontrolle bei der Auswertung der functional uncertainty Constraints zuläßt, was in [61] als wesentliches Mittel zur Effizienssteigerung angegeben wird.

2.3 Morphologie

Sowohl für Parsing als auch für Generierung kann ein Effizienzgewinn durch die Verwendung eines Stammformenlexikons erreicht werden. Zur Analyse bzw. Generierung flektierter Wortformen muß dann eine separate Morphologiekomponente zur Verfügung stehen.

Als morphologische Komponente wurde das System X2MORF implementiert, das eine Erweiterung der Zweistufenmorphologie darstellt [152, 153, 154].

Um eine bessere Integration der morphologischen Komponente in das unifikationsbasierte Gesamtsystem zu ermöglichen, sowie um eine adäquatere Beschreibung verschiedener morphologischer Phänomene zu ermöglichen, wurden die Fortsetzungsklassen durch eine unifikationsbasierte Wortgrammatik ersetzt. Die Anwendung der Zweistufenregeln wird vom erfolgreichen Testen von Filtern gegen die von der Wortgrammatik erzeugten Merkmalsstrukturen abhängig gemacht. Das System erlaubt die deklarative Beschreibung morphologischer Prozesse, insbesondere auch solcher, die nicht auf Konkatenation beruhen, wie etwa Umlautung. Es wurde eine morphologische Grammatik erstellt, die die Flexions- sowie Teile der Derivationsmorphologie des Deutschen abdeckt.

Die merkmalsbasierte Wortbildungsgrammatik der Morphologiekomponente X2MORF wurde an die Grammatik auf Satzebene angepaßt. Zur Darstellung syntaktischen und semantischen Wissens wurde ein Lexemlexikon erstellt. Es wurde eine Schnittstelle zwischen X2MORF und Lexemlexikon definiert, die diese beiden Komponenten integriert. Diese Schnittstelle erlaubt eine flexible Verteilung des morphosyntaktischen Wissens zwischen Morphlexikon und Lexemlexikon. Grundlage und Voraussetzung dafür ist eine einheitlich Lexikonarchitektur, die die Eigenschaften morphbasierter und lexembasierter Ansätze vereinigt.

Im Bereich Derivation wurden im Rahmen von X2MORF erste Versuche zur kompositionalen syntaktisch/semantischen Verarbeitung von mittels Derivation gebildeten Wörtern unternommen. Dazu wurde eine entsprechende Syntax und Semantik bestimmter Derivationsendungen entwickelt. Eine komplexe Schnittstelle zwischen morphologischer Verarbeitung und Lexemlexikon übernimmt dabei die Anwendung und Verarbeitung des syntaktisch/semantischen Wissens.

Als Alternative zu diesem Ansatz wurden im Bereich der Derivation theoretische Grundlagen zu einer vollständigen Inkorporation der Morphotaktik ins Lexikon gelegt [85, 83]. Damit ist es möglich, die Morphotaktik vollständig in Form von Merkmalsstrukturen zu beschreiben. Dies hat den Vorteil, das (i) die zugrundeliegende Logik der Beschreibung klar ist und das (ii) es für einen Parser bzw. Generator egal ist, ob er auf der Satzebene oder auf der Wortebene arbeitet. Für die Derivation wurde gezeigt, daß man auf komplexe funktionale Abhängigkeiten verzichten kann und alleine mit reversiblen Funktionen auskommt. Dies ist besonders in Hinblick auf reversible Grammatiken wichtig. Für die Derivation konnte weiter gezeigt werden, daß es, analog zu hierarchischen Lexika für (freie) Wörter, Sinn macht, gleiches für Präfixe und Suffixe zu postulieren. Ebenso ist es, im Hinblick auf das HPSG-Paradigma, sinnvoll, Prinzipien und Regelschemata für die Derivation anzunehmen. Die Forschungsergebnisse zur Derivation wurden auf mehreren internationalen Workshops

präsentiert und sehr positiv aufgenommen und diskutiert.

Ebenso konnte gezeigt werden, wie man Allomorphy (z.B. Koskenniemis Zwei-Stufen-Morphologie) vollständig innerhalb eines unifikations-basierten Merkmalsformalismus mit Morphotaktik integriert [86]. Die Idee dabei ist, Koskenniemis 2-Stufen Automaten explizit innerhalb des Merkmalsformalismus zu codieren [84]. Alternative Codierungen, wie etwa Birds "1-level phonology" sind hier aber auch vorstellbar.

Der merkmalsbasierte Teil der Morphologiekomponente X2MORF wurde im TDL-Formalismus reimplementiert. Gleichzeitig wurde die Mikro-Architektur dieser Komponente derart modifiziert, daß sich wesentliche Effizienzsteigerungen ergaben. Durch diese Verbesserungen wurde es möglich, X2MORF als Laufzeitmodul in das System zu integrieren (anstatt es, wie bisher, als Off-line-Komponente zur Erzeugung eines morphologischen Vollformenlexikons zu verwenden). Durch eine Reorganisation der Sub-Module von X2MORF und die Erstellung eines Benutzermanuals wurde es möglich, die Modifikation und Wartung des Lexikons neben dem Entwickler auch "naiven" Benutzern zu übertragen. Mit der Erstellung eines Benutzer- und Programmiererhandbuchs wurde begonnen. Des weiteren wurde der generische Kern zur Erzeugung von weiteren Parsern in der Morphologie und der Oberflächen-Sprechakterkennung eingesetzt.

2.4 Grammatik

Im Rahmen des Projekts wurde eine Deutsche Grammatik entwickelt, die im wesentlichen auf die Theorie der Head Driven Phrase Structure Grammar ([139], [140]) aufbaut. Diese Grammatik wurde auf der Basis des TDL Formalismus und der Entwicklungsumgebung implementiert. Forschungsergebnisse wurden dabei vor allem im Bereich Grammatikarchitektur und in der formal-theoretischen Beschreibung des Deutschen erzielt.

2.4.1 Merkmalsgeometrie

Alle Teile der Grammatik sind als getypte Merkmalstrukturen im TDL Formalismus spezifiziert. Mithilfe von Typenhierarchien werden Zeichen im Sinne der HPSG definiert. Die Grammatik weicht dabei in einigen Punkten von der Merkmalsgeometrie der Standard-HPSG ab, wobei sich für sprachliche Zeichen grob die Struktur in Abb.2.4 ergibt.

Das CAT-Attribut einer Struktur repräsentiert die Kategorie einer Struktur, d.h. den Mutterknoten einer phrasalen Struktur oder die Kategorie einer lexikalischen Struktur. Im letzteren Fall fehlt das DTRS Attribut, das ansonsten die Phrasenstruktur mit entsprechenden Namen für grammatische Funktionen bezeichnet.

Als Wert von CAT treten die verschiedenen grammatischen Abstraktionsebenen parallel zueinander auf. Unter SYN wird die syntaktische Struktur und unter SEM die semantische Struktur aufgeführt. Im Gegensatz zur HPSG ist parallel dazu eine morphologische MORPH Struktur, die die lineare Ordnung von terminalen morphologischen Einheiten wiederspiegelt. Der Vorteil dieser Organisation ist, daß Selektion morphologische Information bzw. die Form des Wortes miteinbeziehen kann und sich nicht nur auf syntaktische und semantische

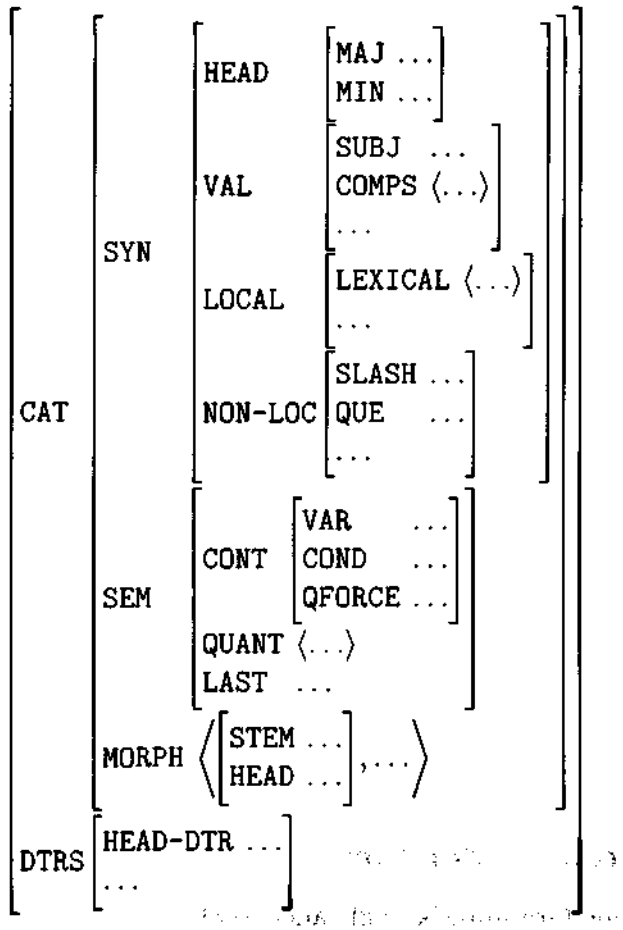


Abbildung 2.4: Merkmalsgeometrie

Merkmale beschränkt. Dadurch wird die Definition von diskontinuierlichen Wortstrukturen (z.B. Präfix-Verben) und idiomatischen und semi-idiomatischen Strukturen wesentlich erleichtert.

Die unter SYN aufgeführten Merkmale HEAD, VAL, LOCAL und NON-LOCAL definieren Teilinformation, die unterschiedlichen Lokalitätsbedingungen unterworfen sind. HEAD-Merkmale unterliegen Head-Feature-Prinzip und definieren z.B. Kongruenzeigenschaften von Projektionen. Die Trennung in MAJOR und MINOR Merkmale dient zur Unterscheidung von funktionalen und substantiellen Eigenschaften von Kategorien. Unter VAL werden verschiedene Valenzmerkmale aufgelistet, die bei der Selektion von Komplementen eine Rolle spielen. LOCAL Merkmale sind echt lokal in dem Sinne, daß sie nicht prinzipiell über den lokalen Bereich einer Regel hinaus weitergereicht werden. NON-LOCAL Merkmale dienen zur Behandlung von nicht-lokalen oder Fernabhängigkeiten und werden auf der Basis von verschiedenen Prinzipien perkoliert. Der Vorteil dieser Geometrie im Vergleich zur Standard-HPSG liegt vor allem in etwas flacheren Merkmalstrukturen und in der erwähnten klareren Strukturierung von Lokalitätsdomänen.

Unter dem SEM-Merkmal wird eine partielle semantische Repräsentation kompositioneil aufgebaut. Die wesentlichen Teilstrukturen sind die CONTENT- und QUANT-Attribute, während LAST lediglich ein für die semantische Komposition verwendetes Hilfsmerkmal ist, welches die Verwendung von append-Operationen erspart und damit eine effizientere Analyse ermöglicht. CONTENT repräsentiert die Semantik des Kopfes. Dabei wird der durch den Kopf eingeführte und als logische Variable repräsentierte Diskursreferent als Index unter VAR herausgehoben, während COND eine atomare Formel (Prädikat-Argument-Struktur) oder eine komplexe logische Formel (Konjunktion) als Merkmalsstruktur darstellt. Quantorenstrukturen enthalten zusätzlich das QFORCE-Merkmal mit dem die Variable bindenden Quantor. QUANT ist eine *Liste* von Quantorenstrukturen, aber auch anderen skopustragenden Operatoren, in der Reihenfolge ihres Vorkommens in der Oberflächenstruktur. Dies erlaubt der semantischen Verarbeitung später die Berechnung von Präzedenzbeziehungen und des Skopus, ohne nochmals auf den syntaktischen Ableitungsbaum zurückgreifen zu müssen.

2.4.2 Architektur der Grammatik

Die Architektur der Grammatik (vgl. auch [121]) sieht verschiedene Typen von Zeichen oder linguistischen Einheiten vor, die in verschiedenen Verarbeitungsschritten herangezogen werden. Diese umfassen die folgenden Entitäten:

- Morphologische Strukturen
- Lexikalische Strukturen
- Lexikalische Regeln
- Grammatikregeln
- Regeln zur Sprechakterkennung

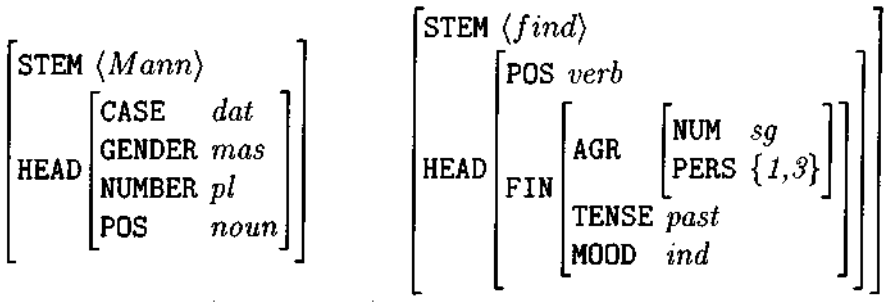


Abbildung 2.5: Morphologische Merkmalstrukturen

Morphologische Strukturen

Morphologische Strukturen definieren den Output, der von der Morphologiekomponte erwartet wird. Diese Strukturen treten als Elemente der Liste auf, die den Wert des Attributs MORPH darstellt. Die Strukturen enthalten zwei Attribute, die den Stamm der Wortform bezeichnen(STEMund die relevanten morpho-syntaktischen Merkmale enthalten. Die Formen *Männern* und *fand* werden z.B. wie in Abbildung 2.5 dargestellt.

Für spezielle Formen, wie Ordinalzahlen oder numerische Datums- oder Zeitangaben, berechnet bereits der der Morphologie vorgeschaltete Scanner die morphologischen Strukturen.

Lexikalische Strukturen

Das eigentliche Lexikon ist in Form eines Lexem-Lexikons strukturiert, das von morphosyntaktischen Variationen und möglichen syntaktischen Alternanzen, wie z.B., Argumentreduktion oder Diathesen, abstrahiert. Die Lexeme sind entsprechend in vollständige lexikalische Einträge und sogenannte Arche-Lexeme unterteilt. Letztere sind dadurch charakterisiert, daß sie, bevor sie als Terminalknoten in die Phrasenstruktur eingesetzt werden können, noch lexikalischen Ableitungsregeln unterworfen werden. Daneben sind Typen für Mehrwortlexeme und leere Terminalknoten vorgesehen.

Vollständige Lexeme Als Index der Lexeme in Bezug auf die Morphologie dient das STEM Merkmal. Nach der morphologischen Analyse wird über dieses Merkmal wird auf einzelne Lexeme zugegriffen und die morphologischen Strukturen werden unter dem MORPH-Merkmal eingesetzt. Durch Koindizierung mit der syntaktischen Struktur können morphosyntaktische Merkmale dann abgegriffen werden. Der Lexem-Eintrag für *Mann* hat entsprechend in etwa die Struktur in Abb.2.6, in die die morphologische Struktur Abb.2.5 eingesetzt werden kann.

Arche-Lexeme Während der Eintrag in Abb.2.6 als vollständiger Eintrag interpretiert wird, gibt es daneben Einträge für Archelexeme, die auf jeden Fall über eine lexikalische

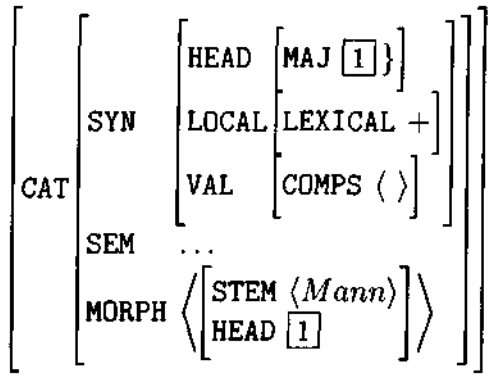


Abbildung 2.6: Lexem-Eintrag für Nomen *Mann*

Regel abgeleitet werden müssen. Beispiele hierfür sind vor allem Verben und Adjektive, bei denen starke Variationen auftreten, die von der morphologischen Form abhängen, wie z.B. finite vs. nicht-finite, Imperativ- oder Passiv-Formen bei Verben oder prädikative, attributive oder adverbiale Verwendung bei Adjektiven.

So enthält der Arche-Lexem-Eintrag für das Verbum *finden* die volle Spezifikation für ein Subjekt und ein Objekt (siehe Abbildung 2.7), obwohl ersteres bei Imperativen oder Passivierung nicht realisiert wird.

Der Vorteil dieser Art von Strukturierung ist natürlich ganz offensichtlich eine Vermeidung von systematischen Redundanzen, die in den lexikalischen Regeln erfaßt werden können.

Mehrwortlexeme Der dritte Typ von Lexikoneinträgen sind Mehrwortlexeme. Diese unterscheiden sich von einfachen Lexemen dadurch, daß sie morphologisch nicht nur eine einzelne Struktur, sondern eine ganze Liste von morphologischen Strukturen überspannen können. Dadurch können Sequenzen von Wortformen zusammengefaßt werden, ohne daß jede einzelne Form syntaktisch-semantisch interpretiert werden muß. Die Anwendung für dieses Konzept sind vor allem standardisierte und nicht semantische kompositioneile Strukturen, wie z.B. Datums- oder Zeitangaben *(Dienstag, der 13. Januar 1994),* komplexe Titel- und Eigennamen *(Herr Prof. Dr. A. v. Müller),* Höflichkeitsfloskeln *(Sehr geehrte Damen und Herren)* und andere idiomatische oder semi-idiomatische Kollokationen. Die Struktur dieser Ausdrücke folgt meist keinem der regulären syntaktisch-semantischen Kompositionsprinzipien in der Regel sondern ist lokal an bestimmte Wort-Formen gebunden.

Das Beispiel in Abb.2.8 zeigt (vereinfacht), wie Uhrzeiten der Form *14 Uhr 33* analysiert und zu einer Einheit zusammengefaßt werden, die von dem Parser wie ein Terminalknoten behandelt werden können. Zwar sind in diesem Beispiel alle morphologischen Teilstrukturen zumindest teilweise spezifiziert, diese Einschränkung gilt jedoch nicht generell: für die Verarbeitung in der Lexikonkomponente genügt es, wenn nur eine Struktur ein spezifiziertes STEM-Merkmal enthält.

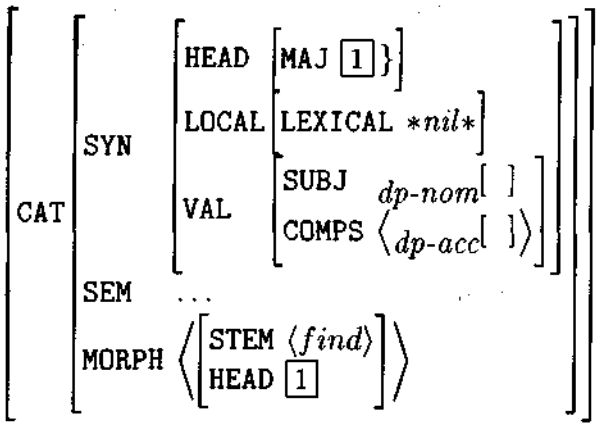


Abbildung 2.7: Arche-Lexem-Eintrag für Verb *finden*

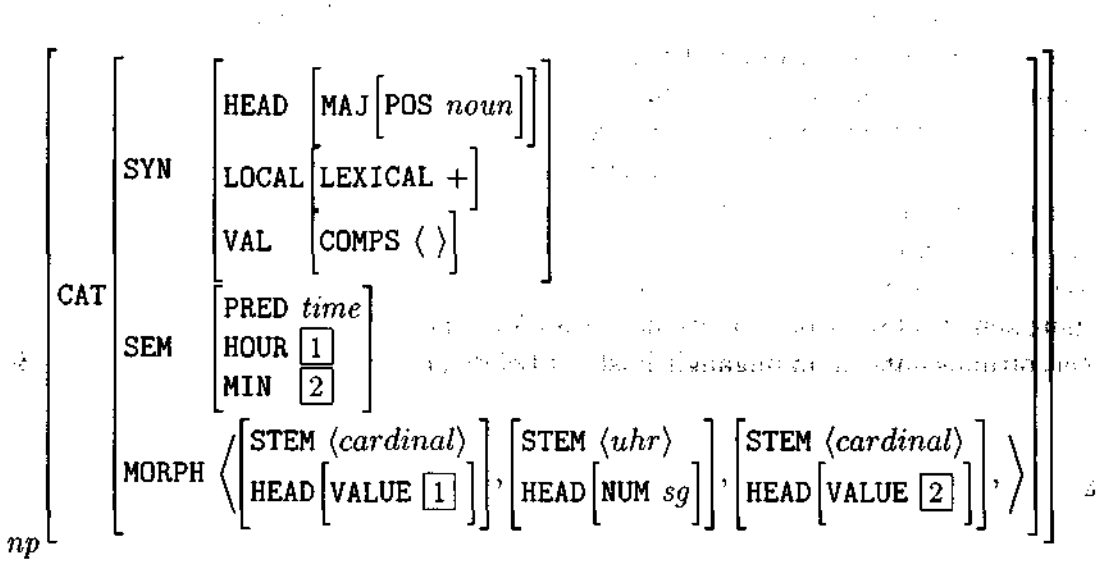


Abbildung 2.8: Mehrwort-Lexem für Zeitangabe

Der Vorteil der Behandlung von idiomatisierten Ausdrücken in Form von Mehrwortlexemen liegt zum einen darin, daß dadurch der Phrasenstrukturteil der Grammatik von eindeutig nicht-rekursiven und nicht-produktiven Regeln freigehalten werden kann, und zum anderen keine semantische Interpretation für Formen notwendig wird, die außerhalb eines spezifischen Kontextes praktisch bedeutungslos sind, bzw. eine andere "wörtliche" Bedeutung haben.

Leere Terminalknoten Als ein Konzept zur Behandlung von speziellen Phänomenen sieht die Architektur der Grammatik auch leere Terminalknoten vor. Diese sind von der Struktur her anderen Lexikoneinträgen vergleichbar, sie werden jedoch kontrolliert vom Parser erst bei Bedarf eingefügt. In der Grammatik werden leere Knoten nur sehr eingeschränkt zur Behandlung von speziellen N-Ellipsen und für Initialstellung des finiten Verbs eingesetzt.

Lexikalische Regeln

Lexikalische Regeln sind in der Grammatik als unäre Ableitungsregeln implementiert, die als Eingabe ein Arche-Lexem, verbunden mit dem Ergebnis der morphologischen Analyse, nehmen und diese auf einen Terminalknoten abbilden, der der Parsingkomponente als Eingabe dient. Lexikalische Regeln werden u.a. zur Ableitung von verschiedenen Verbformen (finit, nicht-finit, Imperativisch) und von Adjektivformen (flektiert attributiv, nicht-flektiert prädikativ oder adverbial) verwendet. Als Beispiel ist hier die Imperativregel (Abb.2.9) aufgeführt, die ein Lexem mit einem nicht-leeren Subjekt in Verbindung mit einer morphologischen Imperativform als Eingabe nimmt (diese Verbindung ergibt sich automatisch aus dem Lexikonzugriff auf der Basis der morphologischen Struktur) und daraus eine um das Subjekt reduzierte Struktur mit einer entsprechenden Semantik liefert.

Während traditionelle lexikalische Regeln eher generativ operieren, indem sie auf ein Lexem angewandt eine Wortform erzeugen, die dann mit der Eingabekette abgeglichen werden kann, sind diese Regeln eher interpretativ, indem sie der Verbindung einer Wortform (der Eingabekette) mit einem Lexems eine Interpretation auf einer höheren Parsingebene zuordnen. Während bei traditionellen lexikalischen Regeln eine Berechnung zur Compile-Zeit oftmals nahezu unumgänglich ist, ist bei den hier verwendeten Regeln eine Berechnung zur Laufzeit viel naheliegender (auch wenn sie theoretisch auch zur Compilezeit eingesetzt werden können). Die Anwendung der lexikalischen Regeln ist damit grundsätzlich datengesteuert und kann im wesentlich nahezu deterministisch erfolgen.

Phrasenstrukturregeln

In der Standard-HPSG werden Phrasenstrukturregeln als eine Konjunktion von abstrakten Regelschemata, die unmittelbare Dominanz kodieren, von Linearen Präzedenzregeln und von Prinzipien. Die Grammatik folgt diesem grundsätzlichen Aufbau, kodiert jedoch die Verbindung der einzelnen Komponenten als Vererbungsrelation in einem Typverband. Die

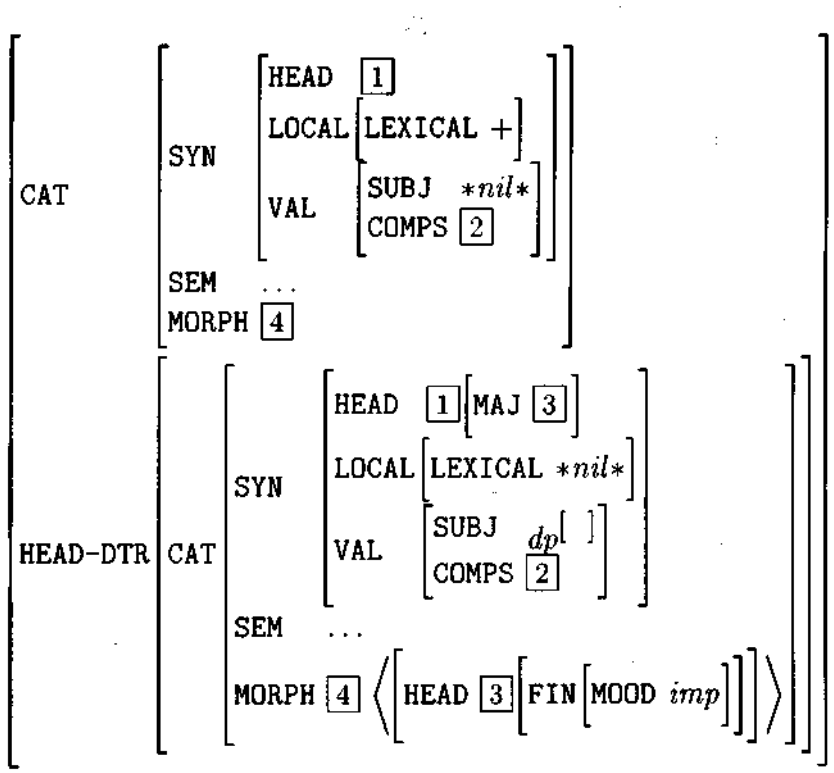


Abbildung 2.9: Lexikalische Regel für Imperative

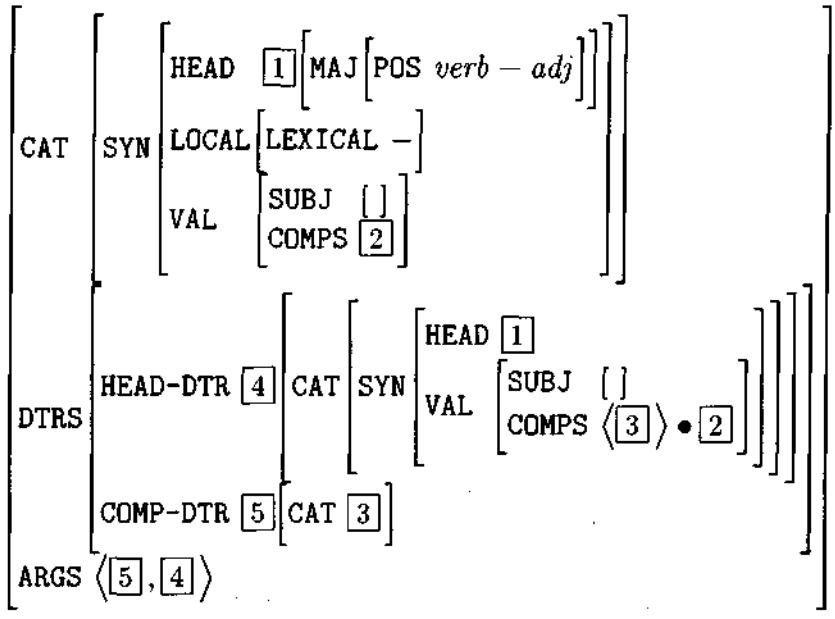


Abbildung 2.10: Binäre Kopf-finale Kopf-Komplement-Regel

von HPSG postulierten universalen Grundschemata für Regeln werden dadurch weiter aufgespalten, so daß von einem Schema letztendlich mehrere Regelinstanzen oder Untertypen gebildet werden. Diese Vorberechnung von Regelinstanzen ermöglicht es, die Regeln mit Standard-Parsing-Techniken zu verarbeiten, anstatt sie, wie in der HPSG teilweise vorgesehen, über eine aufwendigen Typinferenz zur Laufzeit abzuleiten. Da der obere Teil des Typverbandes nach wie vor die wesentlichen Grundannahmen der HPSG widerspiegelt, ergibt sich kein Verlust an Generalität, wohingegen der Effizienzgewinn bei der Verarbeitung beträchtlich sein dürfte.

Die Schnittstelle zum Parser ist über ein Merkmal ARGS definiert, das die Sequenz der Töchter in einer Regel widergibt. Diese lineare Ordnung wird zu den Regelschemata hinzuaddiert, wodurch die Trennung in lineare Präzedenz und unmittelbare Dominanz bei der Spezifikation der Regeln erhalten bleibt. Als Beispiel ist hier eine Kopf-Komplement-Regel aufgeführt (Abb.2.10), die verbale oder adjektivische Köpfe mit einem vorausgehenden Komplement verbindet, wobei die Subkategorisierungsliste COMPS auf der Basis des Valenzprinzips um ein Element reduziert wird.

Als eine Struktur mit einem speziellen Status ist es möglich, Beschränkungen über den Start- oder Root-Knoten einer Ableitung anzugeben. Diese Struktur wird am Ende mit der Wurzel eines vollständigen Parsingergebnisses unifziert und erlaubt es z.B. zu fordern, daß nur vollständig saturierte Phrasen akzeptiert werden. Gleichzeitig dient dieser Root-Knoten dazu, Informationen zu bündeln, die für die Sprechakterkennung relevant sind.

Regeln zur Sprechakterkennung

Die Regeln zur Sprechakterkennung sind wiederum als unäre Ableitungsregeln kodiert, die auf dem Wurzelknoten des Parsingergebnisses operieren. Diese Regeln berücksichtigen (bei sententialen Konstituenten) vor allem die relevanten Aspekte des Satzmodus eines Satzes, d.h. Verbstellung, Verbmodus, Form des Subjektes, Anwesenheit von Interrogativpronomina und die Besetzung des Vorfeldes. Darüberhinaus werden Teile des propositionalen Gehalts, wie z.B. Modalverben oder performative Verben, erkannt. Auf dieser Basis wird die Semantik des Satzes entsprechend um ein Prädikat erweitert, das die möglichen Sprechaktinterpretation(en) kodiert.

2.4.3 Linguistische Abdeckung der Grammatik

Neben der Entwicklung der Grammatikarchitektur wurden auch eine Reihe von Ergebnissen im Bereich der theoretischen Linguistik und der Formalisierung und Implementierung von grammatischen Beschreibungen des Deutschen erzielt. Die theoretische Grundlagenarbeit war hier vor allem auch dadurch erforderlich, daß eine Reihe von Annahmen der Standard-HPSG sich nicht problemlos auf das Deutsche übertragen liessen.

Das Basisfragment des Deutschen umfaßt im wesentlichen

- Nominalphrasenkonstruktionen mit Determination, attributiver Adjektivadjunktion, Possessivkonstruktionen, post-nominalen Adjunktionen von Präpositionalphrasen und Adverbien, und Ellipsen von nominalen Köpfen.

- Verbalkonstruktionen mit NP- und PP-Komplementation, adjektivischer, adverbialer und PP-Adjunktion, Modalverb- und Copula-Konstruktionen

- Präpositionalphrasenkonstruktionen mit einfachen und kontrahierten Präpositionen

- Adjektivkonstruktionen mit Komplementation und Adjunktion in prädikativer, attributiver und adverbialer Funktion

Darüberhinaus wurden speziell die im folgenden beschriebenen Bereiche untersucht, formalisiert und implementiert. Die semantischen Aspekte der Grammatik werden zusammen mit den weiteren Komponenten der semantischen Verarbeitung im Abschnitt 2.5 beschrieben.

Lexikon

Neben der bereits erwähnten architektonischen Strukturierung lag der linguistische Schwerpunkt im Bereich Lexikon vor allem in der Identifikation von relevanten lexikalischen Merkmalen und im Aufbau einer adäquaten Typhierarchie für das Deutsche. Dazu wurden für jede relevante Kategorie des Deutschen zunächst die verschiedenen morpho-syntaktischen

Merkmale erarbeitet, wobei vor allem die verschiedenen Formen von Kongruenz zu berücksichtigen waren. Um einen möglichst hohen Grad an Redundanzfreiheit zu erreichen, wurden dabei die Möglichkeiten der Unterspezifizierung, die sich mithilfe von binären Merkmalen ergeben, voll ausgeschöpft. Darüberhinaus wurden die verschiedenen Formen von Rektion formalisiert und in proto-typischen Einträgen implementiert.

Parametrisierung von grammatischen Regeln

Der Hauptunterschied zur StandardHPSG im Bereich der grammatischen Regeln ist vor allem, daß alle Schemata als binäre Regeln spezifiziert wurden, um eine adäquatere Behandlung von Wortstellungsphänomenen zu erreichen. Dies betrifft vor allem die Komplementationsregeln, deren Formulierung in der HPSG sich primär für konfigurationale Sprachtypen eignet. Mithilfe von binären Regelstrukturen und einer entsprechenden Parametrisierung von Selektionsbeschränkungen und linearen Präzedenz-Regeln können hingegen sowohl konfigurationale als auch nicht-konfigurationale Sprachen beschrieben werden.[1] Durch diesen Ansatz konnten vor allem Wortstellungsphänomene, wie die freie Abfolge von Adjunktion und Komplementen im Deutschen erfaßt werden.

Funktionale Köpfe

Um die komplexen Relationen zwischen Determinern, Adjektiven und Nomina im Deutschen adäquater zu erfassen, wurde die HPSG-Theorie von Spezifikatoren revidiert und ein Alternativvorschlag erarbeitet [123], der davon ausgeht, daß funktionale Kategorien (Determiner, Konjunktionen) als Köpfe zu interpretieren sind, die sich mit substantiellen Kategorien (Nomen, Adjektiv, Verb) in einer bestimmten Weise verbinden. Dieser Ansatz führte zu einer wesentlichen Vereinfachung der Behandlung von Kongruenz- und Deklinationsphänomenen im Bereich der Nominalphrasen. Darüberhinaus ergibt sich daraus eine Analyse von determinerlosen Nominalphrasen ("Mass-Nouns" und reine Plurale), die keine leeren Terminalknoten postulieren muß.

Verbstellung, Verbkomplexe und Satzmodus

Auf der Basis der der Theorie der funktionalen Köpfe wurde ein neuartiger Ansatz für die Behandlung der Stellung des finiten Verbums im Deutschen entwickelt [120]. Dieser Ansatz erklärt unter anderem, warum Konjunktionen und die initiale Stellung des Finitums eine komplementäre Distribution haben. Gleichzeitig erlaubt er, Sätzen mit initialer und finaler Stellung des Verbums eine homogene Struktur zuzuweisen und die für die Behandlung von Wortstellungs- und Skopusphänomenen wichtige binäre Rechtsverzweigung des sogenannten Mittelfeldes aufrechtzuerhalten.

[1] Unter nicht-konfigurationalen Sprachen versteht man grob gesprochen Sprachen mit einer freieren Wortstellung, wie z.B. das Deutsche, bei denen grammatische Funktionen, wie Subjekt, Objekt etc., nicht über eine spezifische Position in einer strukturellen Konfiguration definiert sind.

Im Bereich der Verbsyntax wurde darüberhinaus eine Analyse von Modal- und Kontrollverben implementiert, die sich an der deskriptiven Standardbehandlung für Verbkomplexe im Deutschen orientiert [119]. Auf der Basis dieser Analyse lassen sich Wortstellungsprobleme, wie sie durch sogenannte "Clause Union" und "Verb-Raising"-Phänomene auftreten, erfassen [119].

Auf der Basis der Verbstellungsanalyse konnte auch eine Behandlung von verschiedenen Satzmodi entwickelt werden, die die Haupttypen des Deutschen, d.h. Deklarativsätze, Ja/Nein-Interrogativsätze, W-Interrogativsätze und Imperativsätze, umfaßt. Diese Satzmodi werden eindeutig identifiziert und dienen u.a. als Eingabe für die Sprechaktanalyse.

Nicht-Lokale Abhängigkeiten

Für die Behandlung von nicht-lokalen Abhängigkeiten wurde ein von der Standard HPSG abweichende Analyse entwickelt, die wiederum die speziellen Gegebenheiten des Deutschen berücksichtigt. Der Ansatz unterscheidet sich von dem HSPG-Ansatz vor allem darin, daß er für die Terminierung von SLASH-Werten keine leeren Knoten benötigt, sondern diese mithilfe einer lexikalischen Regel erfaßt. Darüberhinaus wird die Überbrückung von Satzgrenzen lexikalisch gesteuert, so daß die wesentlich stärkeren Restriktionen, denen nicht-lokale Abängigkeiten im Deutschen unterliegen, leichter zu kontrollieren sind.

Datums- und Zeitausdrücke

Für die im COSMA-System relevante Anwendungsdomäne der Terminvereinbarung wurde eine Spezialgrammatik für Datums- und Zeitangaben implementiert. Die Formalisierung erfolgte im wesentlichen mithilfe von Mehrwortlexemen und speziellen Scanner-Mechanismen, die sich für diesen Bereich besonders eignen. Die Abdeckung der Grammatik basiert auf einer empirischen Untersuchung und Datensammlung, bei der verschiedene Konstruktionen und Formate zusammengetragen wurden.

2.5 Semantik

2.5.1 Konzeption

Die Semantikkonstruktion in der Syntax-Semantik-Schnittstelle ist zweistufig organisiert. Die erste Stufe bilden die unter SEM in die Grammatik integrierten semantischen Merkmalsstrukturen. Diese Merkmalsstrukturen stellen metalogische Beschreibungen von semantischen Repräsentationsstrukturen des NLL-Formalismus dar ([104, 107]). In einem zweiten Schritt werden diese unterspezifizierten semantischen Repräsentationen in die vollen semantischen Repräsentationen im NLL- Formalismus überführt. Dabei werden auch der Quantorenskopus bestimmt und anaphorische Bezüge aufgelöst. Dieser Aufteilung der Syntax-Semantik-Konstruktion liegt eine arbeitsteilige Konzeption zugrunde.

Die kompositionelle Seite der Semantikkonstruktion ist in den grammatischen Analyseprozess integriert. Damit werden der Analyse zugleich semantische Constraints auf

lexikalischer und satzsemantischer Ebene zur Verfügung gestellt, welche eine frühzeitige Desambiguierung auf lexikalischer und struktureller Ebene erlaubt. Dagegen werden auf der zweiten Ebene vor allem nicht-kompositionelle Aspekte der semantischen Interpretation behandelt, für die Wissen über den Diskurskontext, zusätzliches Weltwissen und Inferenzen erforderlich sind.

Die Repräsentation im NLL-Formalismus ist damit die Ebene für die semantische Auswertung und Verarbeitung. Sie stellt zugleich die Schnittstelle zur Anwendung dar. Durch die Verwendung eines eigenständigen Repräsentationsformalismus - *NLL* - wird die Anwendungsschnittstelle zugleich unabhängig von der verwendeten Grammatik bzw. dem Grammatikformalismus. So konnten die Schnittstellen auch im ASL- und VERBMOBIL-Projekt verwendet werden, obwohl dort gänzlich andere Grammatikformalismen und Grammatiken auf unterschiedlicher theoretischer Basis als in DISCO verwendet wurden.

2.5.2 Semantikkonstruktion in der Grammatik

Ein erheblicher Teil der Semantikkonstruktion wird bereits während der grammatischen Analyse geleistet. Dies wird ermöglicht durch den Ansatz der Kospezifikation von morphologischer, syntaktischer, semantischer … Information in der SIGN-Struktur ([107, 68]). Diese grammatikinterne Semantikkonstruktion deckt das Grammatikfragment vollständig in folgenden Aspekten ab:

- Spezifikation der Prädikat-Argument-Strukturen
- Spezifikation der Modifikationsrelation für Adjunkte
- die Abbildung von syntaktischen Konstituenten auf thematische Rollen in Prädikat-Argument-Strukturen
- Unterstützung der lexikalischen und strukturellen Desambiguierung durch sortale Constraints und Selektionsbeschränkungen

Unterspezifiziert bleiben die Merkmalsstrukturen vor allem im Hinblick auf Phänomene, welche nicht-lokale Operationen oder nicht-lokale Information erfordern, wie Quantorenskopus oder Antezedenten von Anaphern. Im Falle des Quantorenskopus wurde zwar theoretisch ein Lösungsweg entwickelt ([104]), aber aus Effizienzgründen nicht in die Grammatik integriert. Diese Phänomene werden statt auf der satzgrammatischen Ebene auf der logischen Repräsentationsebene *NLL* behandelt.

Lexikalische Semantik

Für Desambiguierungsaufgaben und zur Formulierung von Selektionsbeschränkungen wurde eine semantische Ontologie entwickelt und als Sortenverband in TDL implementiert. Die Lexeme werden entsprechend dieser Ontologie klassifiziert. Die Sorten erscheinen als Teil der Constraints über die Argumente in den Prädikat-Argument-Strukturen und werden damit als Selektionsbeschränkungen bei der Verarbeitung wirksam ([108]). Dies wird

dadurch begünstigt, daß die Fähigkeiten von TDL die Verwendung eines eigenen taxonomischen Wissensrepräsentationsformalismus für diesen Zweck entbehrlich machen. Zusätzlich kann das Lexikon durch semantische Typen über den Prädikat-Argument-Strukturen semantisch strukturiert werden ([85, 102, 106]).

Kompositionelle Semantik

Die semantische Komposition wird in der Grammatik durch wenige Kompositionsprinzipien beschrieben, welche an die Grammatikregeln vererbt werden. Dabei mußten einige Modifikationen gegenüber der StandardHPSG vorgenommen werden.

- Auf das Semantik-Prinzip für quantifizierte Strukturen, welches mittels der Funktion successively-combine-semantics den Quantorenskopus festlegt, wurde teils aus Effizienzgründen, teils wegen Inadäquatheit verzichtet.
- Das Semantik-Prinzip für Head-Komplement-Strukturen wurde aufgrund der neuartigen Aufteilung von SEM in CONTENT und QUANT aufgespalten in zwei Prinzipien, die das unterschiedliche Projektionsverhalten dieser Merkmale berücksichtigen. Während CONTENT sich wie in der StandardHPSG verhält, wird für QUANT jeweils ein append (mittels Differenzlisten) der QUANT-Listen der Töchter durchgeführt.
- das Semantik-Prinzip für Adjunkte wurde ebenfalls modifiziert und zusätzlich aufgespalten in verschiedene Prinzipien für die NP- und die VP-Adjunktion. Bei der VP-Adjunktion wurde für VP-Komplemente ein zusätzliches nicht-lokales Merkmal SLADJ eingeführt, welches die Adjunktsemantik Zwischenspeichern kann, bis ein zum Adjunkt sortal passender Kopf gefunden ist. Für die NP-Adjunktion mußte der Standard-Mechanismus für den QUANT-Aufbau außer Kraft gesetzt werden, um das Auftreten ungebundener (logischer) Variablen in den Adjunkten zu unterbinden.

2.5.3 *NLL*

Als semantische Repräsentationssprache wird die ursprünglich bei Hewlett-Packard entwickelte Logiksprache *NLL* verwendet ([92]).

NLL basiert auf der Prädikatenlogik erster Stufe. Dieser Kern wurde mit Rücksicht auf die Erfordernisse der semantischen Repräsentation natürlichsprachlicher Phänomene um zahlreiche Konstrukte erweitert:

- Generalisierte und polyadische Quantoren *
- λ-Abstraktion
- thematische Rollen
- nicht-adische Prädikation
- mehrsortige Variablen

- komplexe Prädikate
- Prädikatsoperatoren
- spezielle Konstruktionen für Maßangaben
- Pluralterme
- Modalitäten
- Sprechaktindikatoren
- Propositionen

Zusätzlich wurden Konzepte insbesondere aus der Situationssemantik einbezogen, wie

- *state of affairs* (soas)
- restringierte Parameter für präsupponierte Information

Damit steht mit *NLL* ein äußerst mächtiges und flexibles konzeptuelles Instrumentarium zur semantischen Repräsentation zur Verfügung, in das sich auch Ansätze anderer semantische Theorien integrieren lassen. Die Sprache ist modelltheoretisch interpretiert ([103]). Diese Interpretation wurde zu einer dynamischen Interpretation erweitert, um die Bindung von Anaphern über den syntaktischen Skopus von Quantoren hinaus zu ermöglichen ([67]). Dadurch ist es uns möglich, die Diskursrepräsentationstheorie (DRT) in *NLL* zu emulieren.

2.5.4 Syntax-Semantik-Schnittstelle

Da aufgrund der zweistufigen Konzeption die Semantik partiell im Grammatikformalismus repräsentiert ist, ist eine Schnittstelle zwischen den Repräsentationssprachen erforderlich. Diese besteht aus einem als Compiler konzipierten Translator, der für einen Ausdruck des Ausgangsformalismus (hier: getypte Merkmalsstrukturen) einen abstrakten Syntaxbaum erzeugt, der durch verschiedene Substitutions-, Transformations- und Filterregeln einen abstrakten Syntaxbaum des Zielformalismus (hier: *NLL)* erzeugt, aus dem ein Unparser (Printer) den zielsprachlichen Ausdruck erzeugt ([36, 112, 113]). Vorteile dieses Ansatzes sind:

- Erhöhung der Modularität
- effiziente und flexible Verarbeitung
- leichte Wartbarkeit der Schnittstelle
- Anwendbarkeit auf verschiedene Formalismen, auch in unterschiedlichen Programmiersprachen und -Umgebungen

Die Entwicklung dieser Syntax-Semantik-Schnittstelle motivierte die Entwicklung von Werkzeugen zur automatischen Generierung eines solchen Translators aus deklarativen Spezifikationen. Dazu wurden die Systeme ZEBRA und SLANT entwickelt ([19]), die von der Konzeption sehr ähnlich sind, sich aber in Umfang und Code-Erzeugung unterscheiden.

In beiden Systemen werden die Transformationsregeln als Termersetzungsregeln definiert, die zu Regelsystemen zusammengefaßt das Transformationsystem spezifizieren. Die Systeme stellen dafür eine merkmalsbasierte Spezifikationssprache zur Verfügung. SLANT verfügt darüber hinaus auch über einen graphischen Regeleditor. Aus diesen Spezifikationen wird durch die Compiler direkt ausführbarer Lisp-Code generiert. Beide Systeme werden nicht nur für die Syntax-Semantik-Schnittstelle, sondern allgemein für Schnittstellenspezifikationen eingesetzt.

Da die semantischen Merkmalsstrukturen eine unterspezifizierte metalogische Beschreibung der semantischen Repräsentation darstellen, entsprechen die Typen der Merkmalsstrukturen logischen Strukturtypen in *NLL*. Dadurch kann die Übersetzung weitgehend typgesteuert durchgeführt werden. Dies macht den Übersetzungsvorgang zugleich auf einfache Weise reversibel. Um allerdings auch mit ungetypten Merkmalsstrukturen umgehen zu können, existiert ergänzend ein eigenständiger Klassifikator der Merkmalsstrukturen. Dies erhöht die Flexibilität der Schnittstelle beträchtlich.

Der Übergang von den semantischen Merkmalsstrukturen zu den *NLL*-Repräsentationen ist aber nicht lediglich eine Übersetzung. Gleichzeitig mit der Übersetzung werden der Skopus der Quantoren festgelegt und die logischen Ausdrücke vereinfacht. Die in der QUANT-Liste vorgegebene Reihenfolge wird dabei als Default-Wert für die Skopusbeziehungen behandelt, während CONTENT immer den engsten Skopus hat.

Die gleichen Methoden wurden für die umgekehrte Übersetzungsrichtung verwendet, um aus *NLL*-Repräsentationen wohlgeformte semantische Merkmalsstrukturen zu erzeugen. Allerdings stellt sich hierbei das Problem der semantischen Äquivalenz: es kann nicht allgemein garantiert werden, daß die semantische Merkmalsstruktur, obwohl wohlgeformt, eine ist, die in dieser Form durch die Grammatik ableitbar wäre. Dies stellt vor allem für die Generierung ein Problem dar.

2.5.5 Referenzauflösung und Diskursgedächtnis

Es wurde eine Referenzauflösungskomponente entwickelt, die folgende Phänomene behandelt:

- Personalpronomina
- Reflexivpronomina
- Possessivpronomina
- Definite Beschreibungen (Kennzeichnungen)

Für jede dieser Klassen anaphorischer Ausdrücke gelten eigene Regeln im Hinblick auf mögliche Antezedenten. Dabei wurde angestrebt, die Constraints aus der syntaktischen

Bindungstheorie sowie die Zugänglichkeitsbeschränkungen, welche die DRT postuliert zu integrieren. Auch zu verwandten Problemen in situationstheoretischen Ansätzen wurden Lösungsvorschläge entwickelt ([47, 48]). Die bindungstheoretischen Constraints wurden dem HPSG-Ansatz folgend über der Obliqueness-Hierarchie grammatischer Funktionen definiert, während die DRT-Constraints vor allem durch die Quantoren- und Operatorenstrukturen gegeben sind. Die Einbeziehung dieser Klasse von Constraints war eine wesentliche Motivation für die Spezifikation des Quantorenskopus in der semantischen Repräsentation, statt ihn, wie in der *Quasi Logical Form* der *Core Language Engine*, unspezifiziert zu lassen. Zusätzlich zu diesen strukturellen Constraints werden im wesentlichen Kongruenzmerkmale und die Sortenontologie benutzt.

Mögliche Antezedenten aus vorhergehenden Äußerungen werden in einem Diskursgedächtnis verwaltet, welches aus drei Komponenten besteht:

- einem *Fokus-Stack* von Diskursreferenten-Mengen, dessen oberstes Element per Default den jeweils aktiven Fokus enthält.
- einer Menge von Ä*ußerungsparametern* (Sprecher, Adressat usw.)
- einer Diskurs-Datenbank, in der das im Diskurs vermittelte Wissen über die Diskursobjekte für die Resolution von Kennzeichnungen verwaltet wird.

In zwei Richtungen wurde über die Behandlung pronominaler Identitätsanaphern, die auf im Diskurs explizit eingeführte Referenten aufgelöst werden, hinausgegangen. Zum einen wurde ein Summationsverfahren für die Auflösung pluralischer Pronomina entwickelt, mit dem neue Mengenobjekte aus vorhandenen als Antezedenten konstruiert werden. Zum anderen wurde exemplarisch ein Verfahren zur Auflösung von N-bar-Ellipsen realisiert. Dabei wird keine Bindung zwischen zwei Diskursreferenten hergestellt, sondern es wird ein mit dem elliptischen Ausdruck maximal konsistenter Teil der Beschreibung, unter der ein passender Antezedent eingeführt wurde, rekonstruiert.

Darüber hinaus wurden theoretische Untersuchungen zum Projektionsverhalten von Präsuppositionen in modalen Kontexten durchgeführt ([65]).

2.6 Parsing

Das Parser-Modul von DISCO besteht aus einem in vielen Dimensionen parametrisierbaren Chart-Parser, der dem von Görz [49] ähnelt. Es implementiert sowohl einen CKY-Parser als auch einen echten Earley-Parser [41]. Obwohl keine objektorientierte Sprache zur Implementierung benutzt wurde, wurde es doch in einer objektorientierten Weise implementiert, um zu ermöglichen, daß verschiedene Parser, die verschiedene Aufgaben erledigen (u.U. auf unterschiedliche Weise und mit unterschiedlichen Constraintlöseverfahren) im gleichen System koexistieren können.

2.6.1 Möglichkeiten zur Parametrisierung

Prioritäten für Teilschritte

Der Parser arbeitet task-orientiert, d.h. er zerteilt den Parsingprozeß in kleine Arbeitsschritte, die Tasks. Indem man jedem Task eine Priorität gibt und immer den mit der jeweils höchsten Priorität ausführt, können leicht verschiedene Parsingstrategien eingestellt werden. Da die Funktion zur Berechnung der Prioritäten auch auf die internen Datenstrukturen des Parsers zugreifen kann, können auch sehr komplexe Strategien implementiert werden.

Ein Beispiel für eine solche Strategie stellt die bevorzugte Ausführung solcher Tasks dar, die Nominalphrasen aufbauen. Da diese einen (im Vergleich zu Verbalphrasen) relativ geringen Nichtdeterminismus beinhalten, können sie schnell und zielgerichtet aufgebaut werden und so auch die Komplexität der Berechnung der Verbalphrasen verringern. Die Parsingstrategie kann auch dazu benutzt werden, bevorzugte Lesarten zuerst zu liefern und so eine Best-First Strategie zu implementieren. Dazu ist aber eine enge Verzahnung der Prioritätsberechnung mit Semantik und Pragmatik nötig, da i.a. nur aufgrund dieser Verarbeitungskomponenten solche Entscheidungen gefällt werden können.

Um den Grammatikentwickler bei Auswahl und Entwurf der Parsingstrategie zu unterstützen enthält der Parser eine *Statistikkomponente*, die die Anzahl der erfolgreichen, fehlschlagenden und gefilterten Tasks (auch aufgeschlüsselt nach Regeln), sowie die jeweils dafür benötigte Rechenzeit protokolliert. Diese Daten können nach jedem Lauf ausgewertet werden und in Verbindung mit dem Parsen größerer Satzmengen auch zum Bewerten von Parsingstrategien verwandt werden. Abbildung 2.11 zeigt die Ausgabe der Statistikkomponente nach Verarbeitung eines Satzes.

Ausfiltern von Tasks

Der Parser stellt einen Mechanismus zur Verfügung, der die Erzeugung eines Tasks ganz verhindern kann. Dieser Mechanismus wird im folgenden als *Taskfilter* bezeichnet. Der Taskfilter implementiert folgende Idee: wenn es einfach zu überprüfende Tests gibt, die das Scheitern eines Tasks zweifelsfrei vorhersagen, muß dieser Task nicht auf die Liste der zu verarbeitenden Tasks gesetzt werden, was außer Einfügen und Löschen aus der Datenstruktur auch die Berechnung einer Priorität erfordert. Die mit Filtered beschrifteten Spalten der Statistiktabelle in Abb. 2.11 protokollieren die Wirksamkeit dieses Mechanismus.

Auswertungsreihenfolge der Regelargumente

Es ist möglich, die Auswertungsreihenfolge der Regelargumente zu verändern. Das erlaubt z.B. wahlweises Parsen von links nach rechts oder von rechts nach links. Man kann durch Bevorzugen der Argumentpositionen, die eine höhere Filterwirkung haben (d.h. ihr Argument stärker einschränken), die Effizienz des Parsens steigern, da dann insgesamt weniger Konstituentenhypothesen aufgebaut werden. Die Argumentreihenfolge ist allerdings nicht ganz frei; sie ist auf den Aufbau kontinuierlicher Konstituenten beschränkt.

Parameter settings: •

Task priority function: #<Function DETERMINE-TASK-PRIORITY-BETTER>

Rule	Time		Tasks		
	Success	Failing	Success	Failing	Filtered
0	0.0000	0.0000	4	0	2
ADJUNCT-TOPIC-RULE	0.0000	0.1000	0	1	5
FILLER-HEAD-RULE	0.8830	0.0170	1	1	6
HEAD-FINAL-RULE1	0.0000	0.0830	0	1	4
HEAD-FINAL-RULE2	0.3330	0.0000	2	0	5
HEAD-FINAL-RULE3	0.0000	0.0000	0	0	6
HEAD-FINAL-RULE4	0.0000	0.0670	0	1	4
HEAD-INITIAL-RULE	0.0000	0.1670	0	2	4
HEAD-POSS-RULE	0.0000	0.1830	0	2	4
MW-DATE-TIME	0.0000	0.1510	0	2	4
MW-DAY-DATE	0.0000	0.1510	0	2	4
MW-DAY-DATE	0.0000	0.0660	0	2	4
POSTMOD-LP-RULE	0.0000	0.0000	0	2	4
PREMOD-NP-RULE	0.0000	0.0000	0	0	6
PREMOD-VP-RULE	0.0000	0.0000	0	0	4
SLASH-TERM-RULE	0.0000	0.0000	0	0	5
SUBJECT-TOPIC-RULE	0.5000	0.0000	2	0	6
VP-V-RULE	0.0000	0.0000	0	0	6
—Sum———————1.71600.98509				16	83

Reading	Seconds	Success	Failing	Filtered
1.	2.77	9	16	83
Overall	2.77	9	16	83

Tasks contributing to any reading: 4 tasks
Tasks contributing to first reading: 4 tasks

Number of active items on the chart: 3 items
Active items contrib. to any reading: 2 items
Active items contrib. to first reading: 2 items

Number of passive items on the chart: 6 items
Passive items contrib. to any reading : 5 items
Passive items contrib. to first reading: 5 items

Epsilon items added at: 2

Abbildung 2.11: Beispiel einer Statistiktabelle

Bottom-Up- vs. **Top-Down- (Earley-) Verarbeitung**

Der Parser erlaubt sowohl Top-Down- als auch Bottom-Up-Verarbeitung. Zur Verarbeitung von "reinen" Merkmalsstruktur-Grammatiken bietet sich eine Bottom-Up Verarbeitung eher an, da Top-Down-Constraints eventuell in der Berechnung zu teuer sind. Bei Grammatiken mit ausgearbeitetem kontextfreiem Rückgrat kann Top-Down-Verarbeitung Effizienzvorteilebringen.

Die objektorientierte Implementierung dieses Moduls erlaubt es, verschiedene Parser, die die zuvor beschriebenen Fähigkeiten in unterschiedlicher Weise nutzen, in einem System nebeneinander zu halten. Diese Parser können auch völlig verschiedene Regelformate, Eingabe- und Ausgabestrukturen sowie verschiedene Constraintlöseverfahren benutzen. Ein Parser für reine kontextfreie Grammatiken und ein Parser für Unifikationsgrammatiken können problemlos gleichzeitig in einem System verwandt werden. Dies ermöglicht auch den direkten Vergleich von Grammatiktypen und die Verzahnung verschiedener Verarbeitungsformen, die je nach Grammatik eine Efnzienzsteigerung bewirken. Wie später noch genauer beschrieben wird, werden im DISCO-System mehrere verschiedene Instanzen des Parsermoduls benutzt, die morphologische Analyse, syntaktische Analyse und syntaktische Sprechakterkennung ausführen.

Diese streng modulare Implementierung des Parsers erlaubt außerdem einen direkten und einfachen Einsatz in anderen Systemen; das Modul kann aus dem DISCO-System herausgenommen und ohne Probleme in ein anderes System eingepaßt werden, da der Kern keine anderen Teile aus DISCO benutzt. Die Schnittstelle ist relativ einfach, wegen der starken Parametrisierbarkeit allerdings etwas größer. Wie das Parsermodul eingesetzt werden kann und im DlSCO-System eingesetzt wird beschreibt ein eigenes Handbuch [76].

2.6.2 Chart Display

Mit dem Parsermodul ist ein interaktives Display gekoppelt, daß die Chart während und nach einem Parse anzeigen kann. Es erleichtert das Debuggen sowohl des Parsers selbst wie auch das der benutzten Grammatik. Das Chartdisplay wird in Abschnitt 3.2 genauer beschrieben.

2.6.3 Anwendung im DISCO-System

Die erste und wichtigste Anwendung für das DISCO-Parser-Modul ist die syntaktische Analyse der eingegebenen Sätze. Dieser Parser benutzt direkt die von *TDL* kommende getypte Merkmalsstrukturgrammatik, was ein direktes Debugging der Grammatik erlaubt. Da weder Regeln noch Lexikoneinträge verändert werden, können gefundene Fehler direkt in der Ausgangsgrammatik lokalisiert werden.

Da die direkte Verarbeitung der Unifikationsgrammatik nicht besonders effizient ist, wurden mehrere Maßnahmen zur Effizienzsteigerung getroffen.

Zum einen wurden durch Verändern der Parsingstrategie erhebliche Effizienzgewinne erzielt. Eine starke Verbesserung ergab sich durch die Ausnutzung der Taskfilter: es wurde

eine zusätzliche Komponente implementiert, die nach der Grammatikdefinition die Regeln auf ihre paarweise Unifizierbarkeit prüft. Die Ergebnisse dieses Tests werden im Speicher abgelegt und zur Laufzeit wird anhand dieser Tabelle entschieden, ob der entsprechende Task ausgeführt wird oder nicht. Hierdurch können bis zu 70% der fehlschlagenden Tasks herausgefiltert werden. Diese Tabelle ist manuell erweiterbar, so daß der Grammatikschreiber unsinnige Kombinationen, die nicht automatisch erkannt werden, zusätzlich ausschliessen kann. Insgesamt werden in der momentanen Version der DISCO-Grammatik zwischen 70 und 80 Prozent der fehlschlagenden Tasks vorgefiltert.

Außer zur syntaktischen Analyse wird der Parser auch im Morphologiemodul X2MORF (vgl. Abschnitt 2.3) benutzt. Eine von der syntaktischen Verarbeitung verschiedene Version des Parsermoduls überprüft mögliche morphologische Segmentierungen. Die Sprechakterkennung aufgrund syntaktischer Merkmale benutzt einen ähnlichen Parser wie die syntaktische Analyse; in dieser Anwendung wird das Präferenzsystem vor allem zum Auffinden der bevorzugten Lesart benutzt.

2.7 Generierung

Dieser Abschnitt stellt die Arbeiten zu constraintbasierter Generierung in DISCO vor. Zunächst wird der verwendete Algorithmus skizziert und seine Implementation in dem System SEREAL beschrieben. Danach wird auf die notwendigen Kompilationsschritte für die Laufzeitgrammatik eingegangen. Abschließend werden Probleme dargestellt, die aus der direkten Abhängigkeit von Semantikmodellierung und Verarbeitungsstrategie resultieren.

2.7.1 Der Algorithmus

Die Generierung aus satzsemantischen TDL-Repräsentationen wurde auf der Grundlage des "semantic-head-driven" Verfahrens von Shieber, van Noord, Pereira und Moore [148] (im folgenden durch SNPM abgekürzt) entwickelt. Dieses Verfahren basiert auf der Annahme, daß Grammatikregeln in zwei Klassen unterteilt werden können: Kettenregeln und Nichtkettenregeln. Ein Kettenregel zeichnet sich dadurch aus, daß die Semantik der linken Seite identisch zur Semantik von mindestens einem Element der rechten Seite ist. Dieses Element heißt semantischer Kopf der Kettenregel. Regeln ohne diese Eigenschaft heißen Nichtkettenregeln. Trivialerweise gehören Lexikoneinträge zu dieser letzteren Klasse.

Ausgehend von einem Ausgangsknoten wird ein Pivotknoten definiert, der lediglich die Semantikinformation des Ausgangsknotens enthält. Von Pivotknoten aus generiert der Algorithmus zunächst top-down mithilfe einer Nichtkettenregel. Die Anwendung einer Nichtkettenregel bedeutet die Unifikation ihrer linken Seite mit dem Pivotknoten. Danach generiert das Verfahren rekursiv von einem Element der rechten Seite dieser Regel aus. Die Anwendung einer Kettenregel bedeutet Unifikation ihres semantischen Kopfes mit dem aktuellen Knoten. Danach generiert das Verfahren rekursiv von einem weiteren Element der rechten Seite dieser Regel aus.

Nachdem alle Elemente der rechten Seite einer Regel abgearbeitet (und terminiert) sind,

prüft das Verfahren, ob der Wurzelknoten des generierten Teilbaums mit dem Ausgangsknoten unifiziert. Dann ist die Berechnung beendet. Andernfalls leitet der Algorithmus einen Bottom-up-Schritt ein mithilfe einer Kettenregel.

Die Auswahl der Regeln erfolgt zufällig; schlägt die Anwendung einer Regel fehl, erfolgt chronologisches Backtracking. Auf diese Weise erzeugt der Algorithmus alle Ableitungen, die die Grammatik für die Eingabesemantik zuläßt.

SNPM zeigen anhand von Beispielen, wie gut das Verfahren für lexikonzentrierte Grammatiken geeignet ist. Wenn die Grammatik die Verwendung von Kettenregeln für die Komplementgenerierung ermöglicht, führt dies zu einem vorwiegend bottom-up verlaufenden Prozeß: zunächst wird das Verb im Lexikon gefunden, und anschließend werden die Komplemente aufgrund der Subkategorisierungsinformation des Verbs und der in der Eingabesemantik enthaltenen Information erzeugt.

2.7.2 Das System SEREAL

In DISCO wurde dieses Verfahren in Lisp implementiert und mit der Kernmaschine durch folgende Schnittstellen verbunden:

- Schnittstelle zu *UDiNe* für die Unifikation;
- Schnittstelle zu X2MORF bzw. zu einem Vollformenlexikon;
- Schnittstelle zur Grammatik.

In dem System SEREAL (sentence realizer) ist der entwickelte Generator über die DISCO- Shell (vgl. Abschnitt 6.2) mit dem Parser verbunden, so daß der Benutzer Eingabestrukturen für den Generator bequem von Parsingergebnissen ableiten kann. Da dieselbe Grammatik die Grundlage für Parsing und Generierung bildet, kann somit für jede Semantikdarstellung aufgrund des Parsings festgestellt werden, welche Oberflächensätze ihr entsprechen.

SEREAL ist nicht Teil der aktuellen Version des DISCO-Systems. Die Semantikdarstellung in der Grammatik bildet die Sprache für Eingabestrukturen des Generators. Die aktuelle Semantik wurde auf eine Weise formuliert, die zwar die notwendige Unterteilung in Ketten- und Nichtkettenregeln durch zusätzliche Kompilationsschritte erlaubt, die jedoch zu einer extrem ineffizienten Verarbeitung führt (siehe unten). Daher arbeitet der Generator lediglich mit einer älteren Version der Semantik, die diese Probleme nicht verursacht.

SEREAL erlaubt es dem Benutzer, über eine Menü-Schnittstelle verschiedene Trace-Modi des Generators anzuwählen. Grammatikregeln, Lexikoneinträge und Eingabestrukturen lassen sich über das Menu mithilfe des Feature-Editors visualisieren.

Der Benutzer kann im Parser-Modus jede vom Parser erzeugte Lesart als mögliche Eingabestruktur für den Generator definieren. Im Generator-Modus kann er zwischen verschiedenen Satzmodi wählen und somit zwischen Deklarativ-, Frage und Imperativsätzen unterscheiden.

Eine vollständige Beschreibung von SEREAL findet sich in [30].

2.7.3 Verbesserungen des Algorithmus

Terminierung. Martinovic und Strzalkowski [95] nennen Terminierungsprobleme bei der Top-Down-Expansion für das in SNPM beschriebene Verfahren. Diese Probleme können durch eine einfache Prüfung, ob die Eingabestruktur 'leer' ist, behoben werden. Diese Prüfung basiert auf der Annahme semantischer Monotonie, d.h. die Semantik eines Elements der rechten Seite einer Nichtkettenregel subsumiert die Semantik der linken Seite. Diese Annahme ist bei kompositionaler Semantik erfüllt.

SNPM schlagen eine einfache Abarbeitung der Elemente der rechten Seite einer Regel vor. Sie beachten nicht, daß die Elemente gemeinsame Information enthalten können, so daß die Reihenfolge, in der sie abgearbeitet werden, wichtig werden kann. Beispielsweise kann die Semantik des ersten Elements 'leer' sein, was normalerweise Backtracking zur Folge haben würde. Wäre stattdessen das zweite Element (mit nichtleerer Semantik) zuerst bearbeitet worden, könnte infolgedessen bei entsprechenden Koreferenzen die Semantik des ersten Elements nichtleer sein. Der Generator prüft die Elemente der rechten Regelseiten und legt eine optimale Abarbeitungsreihenfolge fest (vgl. [28]).

Lexikonzugriff. Die Auswahl von Lexikoneinträgen wurde in SNPM nicht behandelt. Bei einem größeren Lexikon ist ein effizienter Zugriff unbedingt erforderlich. Insbesondere ist die einfache Unifikation semantischer Strukturen weder korrekt noch effizient. Sie ist inkorrekt, da das Ergebnis Informationen enthalten kann, die in der Eingabestruktur nicht vorlagen. Sie ist ineffizient, da eine große Menge von Lexikoneinträgen erfolgreich unifizieren könnte. Stattdessen muß jeder geeignete Lexikoneintrag die Struktur, mit der auf das Lexikon zugegriffen wird, subsumieren. Da für *UDiNe* kein allgemeines Subsumtionsverfahren vorliegt, mußte ein Indexierungsansatz für Semantikdarstellungen entwickelt werden. Er beruht auf den folgenden Annahmen:

- Die Unifikation bestimmter Typen von komplexen Semantikdarstellungen, die als inkompatibel gelten sollen, ist nicht erfolgreich. Dies verhindert z.B., daß atomare und komplexe Formeln miteinander unifizieren.

- Die Merkmale atomarer Formeln sind stets vollständig spezifiziert, d.h. jedes Merkmal, das nicht auf der obersten Ebene einer atomaren Formel spezifiziert ist, darf nicht mit ihr unifizieren. Dies stellt die wesentliche Grundlage dar für einen partiellen Test, ob ein Lexikoneintrag die Eingabestruktur subsumiert, d.h. ob er keine zusätzliche Information enthält.

- Für die zu verwendende Grammatik kann eine Menge von Pfaden fester Länge angegeben werden, die in Semantikdarstellungen zu Merkmalwerten führen, mit denen das Lexikon indiziert ist. Gewöhnlich handelt es sich dabei um atomare Werte, die das semantische Prädikat eines Lexems denotieren.

Ein Prioritätensystem für die Pfade schränkt die Menge zugänglicher Indices für eine gegebene Semantikstruktur ein. Die Indices sind so gewählt, daß nur sehr wenige (bis zu

drei, in Ausnahmefällen zehn) Lexeme zugänglich sind. Diese werden der Reihe nach dem genannten Test unterworfen. Diejenigen, die ihn bestehen, werden als Nichtkettenregeln dem Generator zur Verfügung gestellt.

Weitere notwendige Annahmen über die Eigenschaft der Semantikdarstellung sind:

- Jeder Lexikoneintrag hat eine nicht-leere Semantik, die zumindest eine Indexierung erlaubt.[2]
- Bedeutungsunterschiede von Lexemen schlagen sich (unter anderem) in ihrer semantischen Repräsentation nieder.

Der Ansatz zum Lexikonzugriff erwies sich als effizient und weitgehend unabhängig von der Größe des Lexikons. Allerdings müssen bei jeder Änderung der Semantikrepräsentation die Zugriffspfade angepaßt und priorisiert werden. Darüber hinaus kann mit diesem Ansatz nicht die Vollständigkeit nachgewiesen werden.

2.7.4 Die Laufzeitgrammatik des Generators

Im Grundsatz sind die Grammatiken für Parser und Generator dieselben. Die Laufzeitgrammatik des Generators unterscheidet sich jedoch von der Parsinggrammatik durch Explizierung einiger generierungsrelevanter Information. Die Laufzeitgrammatik wird durch eine Reihe einfacher Kompilationsschritte aus der Parsinggrammatik abgeleitet.

Kettenregeln erfordern die Kennzeichnung des semantischen Kopfes durch ein Merkmal, das dem Generator zugänglich gemacht wird. Ferner werden die Liste der verbleibenden Elemente der rechten Regelseite sowie die linke Regelseite durch entsprechende Merkmale gekennzeichnet.

Disjunkte in der Semantikdarstellung innerhalb einer Regel können alternativ eine Ketten- oder eine Nichtkettenregel implizieren. In diesen Fällen wird die Disjunktion aufgelöst und für jedes Disjunkt eine eigene Regel definiert.

Grammatiken, die Verbletztstellung als Basiskonfiguration annehmen und bei alternativen Verbpositionen eine Spur für das finite Verb ansetzen, stellen ein spezielles Problem für semantikbasierte Generatoren dar. Von der Spur aus sollten die Verbkomplemente bottom-up generiert werden. Dies erfordert die Präsenz der lexikalischen Subkategorisierungsinformation des Verbs und somit den Zugriff auf den Lexikoneintrag, ohne daß das Verb an dieser Stelle morphologisch realisiert wird. Die Lösung des Problems besteht in der Kompilation der (lexikalischen) Spur in eine Kettenregel, deren Tochter das Finitum darstellen kann.

2.7.5 Implementation

Die Backtracking-Strategie der Prologimplementation von SNPM wurde in Lisp durch einen allgemeinen Mechanismus imitiert, der alternative Funktionsaufrufe als funktionale

[2]Indexierung anhand syntaktischer Eigenschaften ist zwecklos, da die Strukturen, aufgrund derer der Lexikonzugriff erfolgt, ausschließlich sernantische Information enthalten.

Objekte auf einen Stream schreibt und, sobald ein Backtrackpunkt erreicht ist, das nächste funktionale Objekt vom Stream liest und evaluiert. Die Verwaltung der Änderungen erfolgt mithilfe der Kontrollobjekte von *UDiNe*.

Unifikation erfolgt destruktiv mit *UDiNe*, wodurch teures Kopieren der Merkmalstrukturen weitgehend entfällt. Nur wenn eine Regel oder ein Lexikoneintrag benötigt werden, entsteht Kopieraufwand. Benutzte Kopien werden wiederverwendet, denn etwaige Änderungen durch den Unifikator werden durch Backtracking aufgehoben. Somit wird wiederholtes Kopieren desselben Objekts vermieden.

2.7.6 Semantikdarstellung und Generierung: Probleme

Trotz vieler erfolgreicher Anstrengungen, das Verfahren von SNPM zu verbessern und auf große Grammatiken erfolgreich anzuwenden, bleibt nach unseren Erfahrungen ein Nachteil, der in der direkten Abhängigkeit zwischen dem Algorithmus und dem Design der Semantikdarstellung liegt. Die Verarbeitungsstrategie hängt davon ab, ob der Grammatikschreiber eine Regel als Kettenregel oder als Nichtkettenregel formuliert. Effizienzüberlegungen im Hinblick auf einen bestimmten Algorithmus werden somit in der Praxis mit der Frage nach linguistisch adäquaten Repräsentationen verknüpft.

Im folgenden werden einige Probleme mit der Semantikdarstellung skizziert und der Versuch geschildert, sie durch geeignete Kompilationsstrategien zu lösen. Die Semantikdarstellung in der DISCO-Grammatik mußte einem dreifachen Zweck dienen:

1. sie soll eine linguistisch wohlmotivierte, oberflächennahe Semantik natürlichsprachlicher Ausdrücke definieren,
2. sie soll als Schnittstelle für nachfolgende Desambiguierungsprozesse dienen (Anaphern- und Skopusresolution) und
3. sie soll die Eingabesprache für Generierung bilden.

In DISCO wurden Grammatiken mit einer Semantikdarstellung versehen, die das erste und dritte Ziel erfüllte. Diese Semantikdarstellung bezeichnen wir als "frühe" Semantik. Grammatiken mit der aktuellen Semantik (vgl. Abschnitt 2.5.2) genügen dem ersten und dem zweiten Ziel, nicht jedoch dem dritten.

Abbildung 2.12 zeigt die einfache Semantikdarstellung in der frühen Version. Die Head-Tochter ist der semantische Kopf. Die Semantik der Komplement-Tochter wird durch den Lexikoneintrag des Verbs vermittels des SUBCAT-Merkmals spezifiziert. Abbildung 2.13 zeigt die aktuelle Semantikdarstellung (die Subkategorisierungsinformation blieb im großen und ganzen erhalten). Das QUANT-Merkmal dient zur Speicherung der Oberflächenreihenfolge der (meisten) Konstituenten, die zur späteren Anaphernresolution benötigt wird (Ziel 2). `QUANT` besteht aus Listen von Semantikdarstellungen von Konstituenten. Die konkatenierten Listen der QUANT-Merkmale von Head- und COMP-Tochter bilden das QUANT-Merkmal der Mutter.[3] Offensichtlich ist die aktuelle Version im Gegensatz zur frühen keine Ketten-

[3]Die Implementation benutzt Differenzlisten.

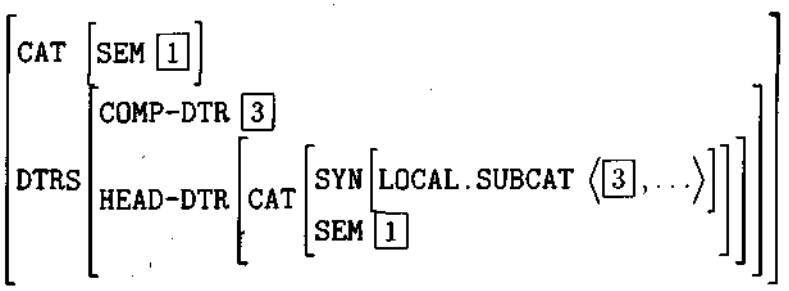


Abbildung 2.12: Head-Final-Rulel in der "frühen" Version.

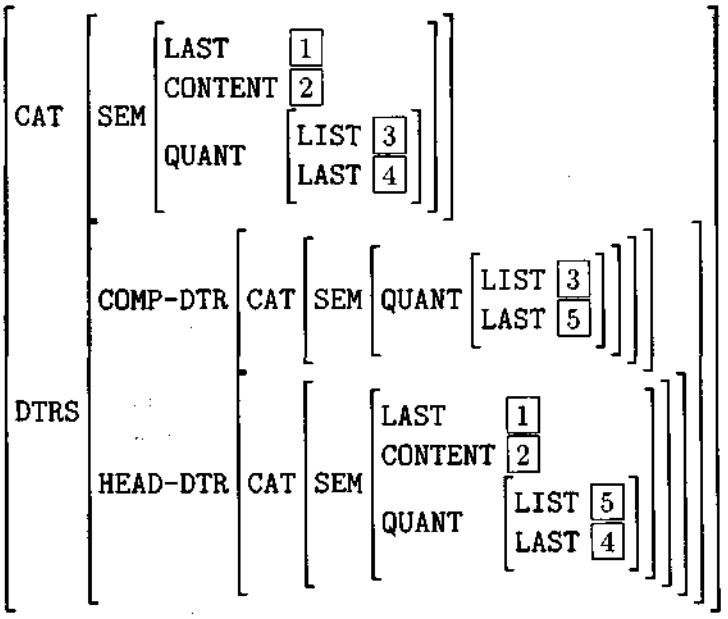


Abbildung 2.13: Head-Final-Rulel in der aktuellen Version.

regel. Eine Top-Down-Expansion ist jedoch nicht deterministisch möglich, da es mehrere Möglichkeiten gibt, die QUANT-Liste an der linken Seite in Sublisten zu unterteilen.

QUANT enthält (bei Adjunkten) Information, die für die Indexierung des Lexikons wesentlich ist. Daher scheidet die Möglichkeit aus, lediglich das CONTENT-Merkmal als Pivot zu benutzen.

Der Konkatenationsmechanismus wird in seiner Allgemeinheit nicht voll ausgenutzt. Die Grammatik ist so gestaltet, daß Komplementtöchter maximal ein Element in QUANT beitragen. Daher wurde versuchsweise mittels Änderung der Grammatik die Länge der Liste auf null bzw. eins festgelegt. Im ersten Fall ergeben sich Kettenregeln, da nun neben CONTENT auch QUANT an linker Seite und Head-Tochter gleich ist. Im zweiten Fall ergeben sich Nichtkettenregeln infolge der unterschiedlichen QUANT-Merkmale. Mit der solchermaßen geänderten Grammatik wurden einige Tests durchgeführt, die jedoch ein exponentielles Laufzeitverhalten zeitigten. Dies liegt daran, daß die Kompilation des QUANT-Merkmals in Ketten- und Nichtkettenregeln etwa ein Dutzend Nichtkettenregeln erzeugt, deren Anwendbarkeit infolge der stark unterspezifizierten Semantik nahezu unbeschränkt bleibt.

Als Konsequenz ist festzustellen, daß die Semantikdarstellung mit allen drei Zielen überfordert ist. Die Eingabesprache für den Generator sollte getrennt von der Semantikdarstellung formuliert werden. Dann können die generierungsspezifischen Anforderungen leichter erfüllt werden.

2.8 Lernen von Ableitungsmustern

Im DISCO-Projekt wurde eine Methode zum automatischen Lernen von generalisierten Ableitumgsmustern entwickelt. Zentrale Einsatzbereiche sind zum einen Effizienzsteigerungen der grammatikalischen Analyse und zum anderen die Verarbeitung von Subsprachen [129, 118].

Die entwickelte Methode folgt dem Modell des erklärungsbasierten Lernens (kurz EBL) (vgl. hierzu [100, 43, 150]). Die EBL-Methode wurde erstmals für die Verarbeitung von constraint-basierten Grammatiken von Rayner und Samuelsson [142, 144] eingesetzt. Dort wurde EBL zur korpusbasierten Kompilation der Kompetenzgrammatik verwendet. Unser Ansatz zeichnet sich dadurch aus, daß eine enge Verzahnung mit dem normalen Parsingprozeß ermöglich wird, was im Ansatz von Rayner et al. geringere Bedeutung besitzt. Die Vorteile einer engen Verzahnung sind jedoch:

- Die Verarbeitung häufig vorkommender grammatikalischer Konstruktionen kann durch Einsatz sehr effizienter Methoden erheblich beschleunigt werden.

- Eine enge Verzahnung mit dem normalen Parsingprozeß erlaubt eine höhere Flexibilität, die andernfalls verloren ginge.

- Die erworbenen Strukturen sind, bezogen auf die verwendete Kompetenzgrammatik, *gültige* grammatikalische Strukturen.

Bevor auf Einzelheiten der im DISCO-Projekt entwickelten Methode eingegangen wird, folgt im nächsten Abschnitt eine kurze Einführung in die Methode des erklärungsbasierten Lernens.

Erklärungsbasiertes Lernen. Erklärungsbasiertes Lernen (EBL) ist eine Lernmethode, in der angestrebt wird, auf der Basis von Beobachtung und anschließender Generalisierung von Beispielen die Verarbeitungsgeschwindigkeit ähnlicher Eingaben zu erhöhen. In diesem Sinne lernt ein intelligentes System aus der "Erfahrung" früherer Analysen. Wichtig ist, daß die Generalisierungen auf der Basis von Hintergrundwissen vorgenommen werden (damit unterscheidet sich EBL wesentlich von induktiven Lernmethoden). Als Methode wird EBL für folgende Lernaufgaben eingesetzt: Generalisierung, Chunking, Operationalisierung und Analogiebildung ("justified analogy").

Im Kontext natürlichsprachlicher Verarbeitung stellt die Kompetenzgrammatik das Hintergrundwissen dar. Der Parser übernimmt die Aufgabe des (grammatikalischen) Problemlösens. Der durch den Parsingprozeß berechnete Ableitungsbaum eines Beispielsatzes wird als gültige Erklärung interpretiert. Unter dem EBL-Gesichtpunkt bedeutet dies, daß der Ableitungsbaum eine Erklärung dafür liefert, warum der analysierte Satz ein gültiges Element der durch die Kompetenzgrammatik definierten Sprache ist.

Die berechneten Ableitungsbäume einer Trainingsinstanz werden dann benutzt, um gültige Generaliserungen zu erhalten. Dies wird im wesentlichen durch Entfernen ("uninstantiating") von bestimmten lexikalischen Merkmalswerten erreicht. Diese generalisierten Ableitungsbäume (sie werden auch als Templates bezeichnet) werden für neue Eingaben so eingesetzt, daß die in einem Template aufgeführten Regeln in der vorgegebenen Sequenz deterministisch angewendet werden. Ist dies möglich, führt es zu einer analogen grammatikalischen Struktur.

Die EBL-Methode im DISCO-System. Abbildung 2.14 zeigt die Architektur der EBL-Methode für das DISCO-System.

Der rechte Teil der Graphik zeigt die linguistische Kompetenzbasis (LCB, Linguistics Core Base) bestehend aus morphologischer Komponente, Parser, HPSG-basierter Grammatik und Lexikon. Der EBL-Modul (linker Teil) besteht aus einer Trainings- und Anwendungskomponente und einer Menge von Templates, die wir als Ableitungsgeschichte bezeichnen.

Der zugrundeliegende Kontrollfluß kann kurz wie folgt beschrieben werden: Während der Trainingsphase des Systems wird eine Trainingsinstanz a (also ein Beispielsatz) LCB als Eingabe überreicht. Nach einer morphologischen und grammatikalischen Analyse resultiert der Ableitungsbaum $d(\alpha)$ (genauer: eine Merkmalsstruktur, die den Ableitungsbaum enthält). Die Trainingskomponente des EBL-Modules TM generalisiert diesen Baum zu $d'(\alpha)$ und speichert das neue Template in einem Diskriminationsnetz ab. Ein entsprechender Pfadausdruck wird auf der Grundlage der morphologischen Kette, die Terminalkette des Parsingergebnisses ist, konstruiert.

Für eine neue Eingabe β wird zuerst eine morphologische Analyse durchgeführt. Die

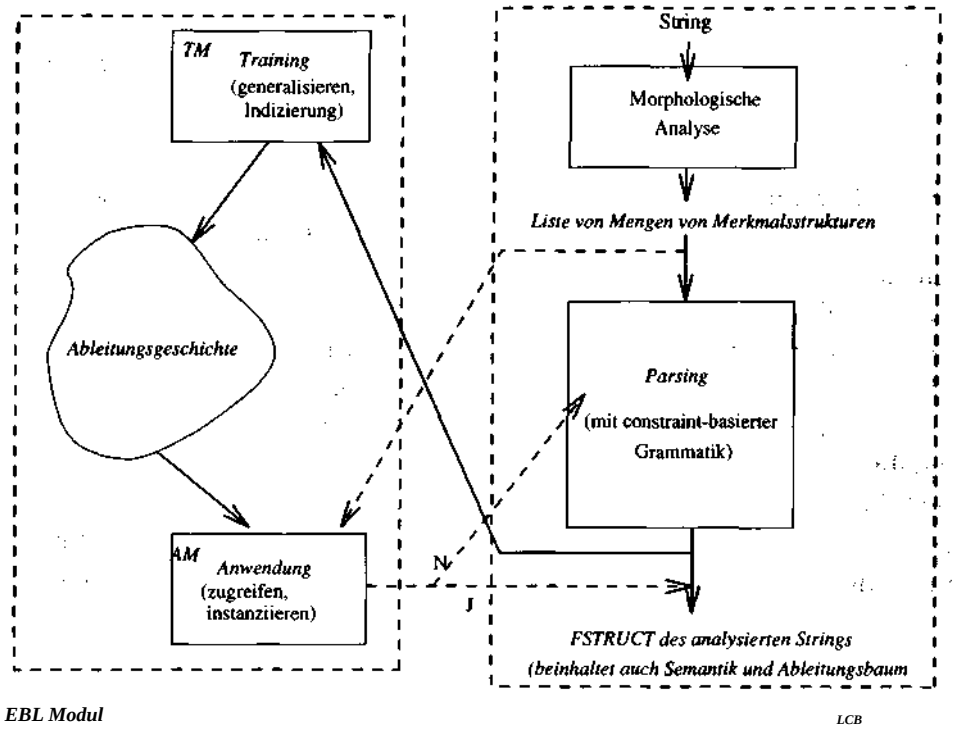


Abbildung 2.14: Die Architektur der EBL-Methode des DISCO Systems

resultierende morphologische Kette wird nun jedoch nicht dem Parser übergeben, sondern zum Traversieren des Diskriminationsnetzes benutzt. Falls der Zugriff erfolgreich ist, was bedeutet, daß ein Template gefunden wurde, wird dieses Template mit der morphologischen Information instanziiert. Der auf diese Weise bestimmte Ableitungsbaum stellt eine grammatikalische Analyse für β dar. Wenn dagegen kein Template extrahiert werden kann, oder wenn die Instanziierung fehlschlägt wird der Parser mit der morphologischen Kette aufgerufen.

Ein Template zusammen mit dem korrespondierenden Index des Diskriminationsnetzes charakterisiert die Ableitung aller Sätze einer Sprache gleicher Länge (basierend auf der Länge des Indexes), konsistenter morphologischer Information und gleicher Ableitung.

Damit hat aber die eben beschriebene Methode folgende Nachteile:

- Die Länge eines Strings ist notwendiges Kriterium für einen erfolgreichen Zugriff auf mögliche Templates.

- Da identische Teilbäume von Templates nicht geteilt ("geshared") werden können, liegt eine relevante Redundanz vor.

Um beide Probleme in den Griff zu bekommen, werden wir nun kurz beschreiben, wie die EBL-Methode auf beliebige Phrasen anwendbar ist.

Verarbeitung beliebiger phrasaler Strukturen durch EBL. Die Trainingsphase verläuft im wesentlichen wie bereits oben beschrieben. Wir benutzen jedoch einen komplizierteren Generalisierungsschritt.

Die zentrale Idee ist, von dem durch den Parser bestimmten Ableitungsbaum ein *Skelett* unter Verwendung vordefinierter phrasaler Typen zu extrahieren. Phrasale Typen sind als Constraints formuliert und definieren strukturelle Eigenschaften phrasaler Strukturen (z.B. maximale Projektionen, Art und Grad der Rekursivität, Beziehung zum Mutterknoten etc.). Der Wurzelknoten eines Ableitungsbaumes wird stets als phrasaler Typ interpretiert.

Ein Skelett kann als *kompakte strukturelle Beschreibung* des originalen Ableitungsbaumes interpretiert werden. Die phrasalen Typen definieren hierbei nichtterminale Elemente im Skelett. Terminale Knoten werden zur Bestimmung von Pfadausdrücken im Diskriminationsnetz verwendet.

Ein Skelett wird in Form von kontextfreien Regeln notiert. Die Regeln eines Skelettes werden strikt bottom-up links-nach-rechts abgearbeitet. Der Zugriff in das Diskriminationsnetzes entspricht daher einem Scanning. Das Instanziieren einer Regel teilt eine Regel in einen aktiven und einen inaktiven Teil. Es ist nun möglich, die Instanzierung durch eine deterministische Earley-Strategie vorzunehmen.

Verzahnung mit der normalen Verarbeitung. Die Tatsache, daß Templates Generalisierungen von durch den normalen Parsingprozeß berechneten Ableitungsbäumen sind, garantiert, daß sie auch wohlgeformt sind. Wenn die Auswahl des Trainingsmaterials realen Anwendungssituationen entspricht, beschreiben diese tatsächlich vorkommende Strukturen.

Die Integration der Anwendungsphase von EBL mit der normalen Analyse stellt sicher, daß der Parser auch für nur partiell instanziierte Templates diese Information nutzen kann und zumindest für diese eine erneute Berechnung vermeidet.

Wenn, wie im DlSCO-System, ein chartbasierter Parser eingesetzt wird, können die durch EBL bestimmten phrasalen Strukturen zur Initialisierung der Chart verwendet werden. Der Parser kann nun diesen Strukturen höchste Prioriät zuweisen. Wenn weiterhin verlangt wird, daß in die Ableitungen von Templates nicht "hineingesprungen" werden darf, dann kann effektiv nur die durch EBL definierte Subsprache verarbeitet werden. Damit wird jedoch der Suchaufwand für den Parser ergeblich verringert.

Implementation. Die einfache Variante ist vollständig im DISCO-System implementiert. Erste Experimente zeigen eine Verringerung der Laufzeit durch EBL von Faktor 20 und mehr. Die algorithmische Spezifikation der phrasalen Methode ist abgeschlossen und wird in einem der Nachfolgeprojekte realisiert werden.

n

Kapitel 3

Werkzeuge

Komfortable Software-Werkzeuge sind unabdingbar für die Entwicklung, die Erprobung und die Erweiterung natürlichsprachlicher Systeme. Die übersichtliche Präsentation der Verarbeitungsergebnisse stellt bei größeren constraint-basierten Grammatiken ein beträchtliches Problem dar. Die gewaltige Anzahl von Merkmalspezifikationen, die einen einzelnen Satz beschreiben, muß durch eine spezielle, hocheffiziente Komponente im Griff gehalten werden. Abschnitt 3.1 beschreibt den in DISCO entwickelten Editor für Merkmalstrukturen.

Insbesondere bei der Grammatikentwicklung ist es wichtig, Informationen über einzelne Analysezustände zu bekommen. Hierfür wurde in DISCO ein Chart-Display entwickelt, das die einzelnen Parsingschritte anzeigt und den mausgestützten Zugang zu den zugrundeliegenden linguistischen Objekten erlaubt (Abschnitt 3.2).

Die Erprobung von Grammatiken muß durch eine möglichst vollständige Beschreibung der sprachlichen Phänomene unterstützt werden. In DISCO wurde eine Datenbank für deutsche Testsätze entwickelt und eingesetzt (Abschnitt 3.3).

Unter Verwendung umfangreicher Lexika des Sonderforschungsbereichs 100 wurden wesentliche Grundlagen geschaffen, um ein großes Fragment des Deutschen abzudecken. Diese Arbeiten beschreibt Abschnitt 3.4.

3.1 Ein Editor für Merkmalsstrukturen: Fegramed

Fegramed ist ein voll interaktiver Editor zum Entwickeln, Überprüfen und Anzeigen von Merkmalsstrukturen. Es wurde als Werkzeug zur Grammatikentwicklung erstellt, um die Komplexität von Unifikationsgrammatiken in den Griff bekommen zu können.

Da Merkmalsstrukturen im wesentlichen als gerichtete Graphen betrachtet werden können und Fegramed nichts von der Semantik von Merkmalsstrukturen weiß, kann man ihn auch als Editor für Graphen betrachten. Hierbei sind Konjunktionen, Disjunktionen, Implikationen usw. nichts als spezielle Knoten in einem Graphen, die eine jeweils andere Darstellungsart erhalten. Einen Eindruck von den verschiedenen Darstellungsmöglichkeiten von Fegramed vermittelt Abbildung 3.1.

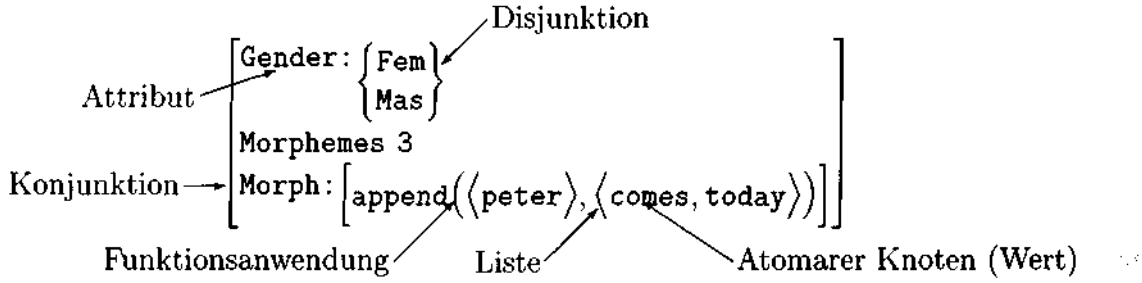


Abbildung 3.1: Verschiedene Darstellungsmöglichkeiten in Fegramed

In Merkmalsstrukturen können zwei Attribute den gleichen Wert teilen; das entspricht zwei Kanten, die auf den gleichen Knoten zeigen. Solche Situationen werden durch *Koreferenzen* dargestellt (siehe Abbildung 3.2).

Da nur das erste Vorkommen der Koreferenz den Knoten (den Wert) an dieser Stelle anzeigt, gibt es die Möglichkeit, durch Anklicken einer leeren Koreferenz den Wert jeweils an dieser Stelle anzuzeigen. Er verschwindet dann an der Stelle, an der er zuvor gezeigt wurde.

Da die Merkmalsstrukturen in Unifikationsgrammatiken oft sehr groß werden können (man denke z.B. an die Analysen längerer Sätze), kann man in Fegramed bestimmte Attribute, die für die aktuelle Ansicht unwichtig sind, ausblenden. Die Merkmalsstruktur läßt sich so in den meisten Fällen schon auf eine handhabbare Größe bringen. Außerdem lassen sich komplexe Knoten zu kleinen, anklickbaren Flächen *implodieren*, um schon betrachtete Teile selektiv auszublenden.

Eine weitere Möglichkeit zur Verkleinerung der Strukturen ist die Beschränkung der Anzeigetiefe. Bei Erreichen der maximalen Tiefe werden komplexe Strukturen durch anklickbare Felder ersetzt (ähnlich wie bei implodierten Knoten), in die *hineingezoomt* werden kann. Dabei werden dann tieferliegende Teile der Struktur gezeigt (siehe Abbildung 3.3).

Wenn man sich in einer tieferen Ebene der Struktur befindet, kann man auch ebensoleicht durch Anklicken des äußeren Randes oder eine Tastenkombination wieder herauszoomen, d.h. eine höhere Ebene anzeigen lassen.

Die Ansicht kann auch durch Angabe der Sortierreihenfolge der Attribute verändert werden. Alle nicht angegebenen Attribute werden nach den angegebenen lexikographisch sortiert (der Normalfall). Diese Reihenfolgen können ebenso wie die Liste der verdeckten Merkmale abgespeichert und in einer anderen Sitzung wiederverwendet werden.

Außer der komfortablen Ansichtskomponente ist Fegramed ein vollständiger Editor.

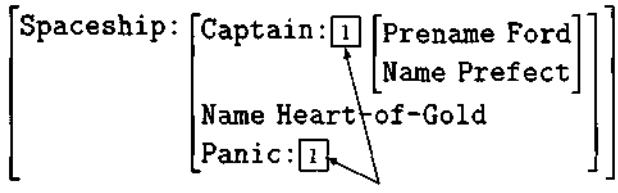


Abbildung 3.2: Koreferenzen in einer Merkmalsstruktur

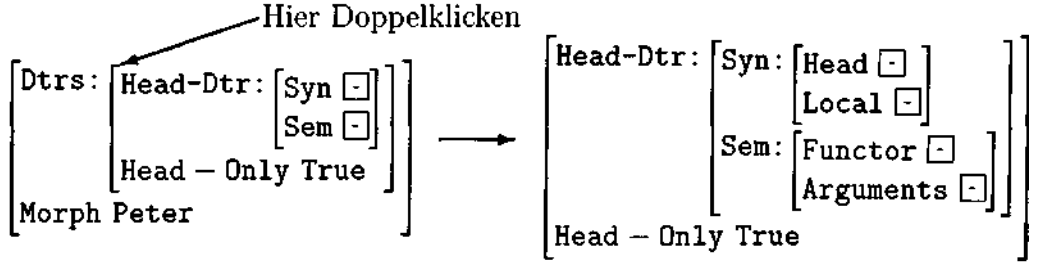


Abbildung 3.3: In den Wert von Dtrs zoomen

Grundlegende Editierkommandos sind z.B. das Einfügen von Knoten und Kanten jeder Art und das Setzen verschiedener Schalter für spezielle Zeichensätze oder die Negation eines Knoten.

Die folgenden mächtigeren Editieroperationen können ebenfalls ausgeführt werden.

- Löschen von Knoten und Kanten und den darunterliegenden Strukturen

  Es wird darauf geachtet, daß Strukturen, die in nicht gelöschten Teilen der Merkmalsstruktur noch gebraucht werden, nicht zerstört werden

- Kopieren und Einkleben von Knoten

  Es ist möglich, ganze Subgraphen zu kopieren und an einer anderen Stelle im Graphen wieder einzusetzen. Dabei werden die inneren Koreferenzen des Subgraphen erhalten. Diese Operation funktioniert nicht nur innerhalb einer Struktur, sondern es können auch Teilstrukturen von einem Fenster in ein anderes und somit in eine völlig andere Merkmalsstruktur kopiert werden.

- Erzeugen und Löschen von Koreferenzen

  Die Endknoten von Kanten können umgesetzt werden, was das Erzeugen und Löschen von Koreferenzen erlaubt. Setzt man allerdings den Endknoten einer Kante um, auf die nur diese Kante zeigt, kommt das dem Löschen der Substruktur gleich.

- Bewegen von Subgraphen

  Subgraphen können mithilfe der Operationen zum Erzeugen und Löschen von Koreferenzen innerhalb einer Struktur bewegt werden. Alle Koreferenzen bleiben hierbei erhalten.

- Exportieren als Graphik

  In der Macintosh Version von Fegramed ist es möglich, die Merkmalsstruktur ganz oder teilweise als Graphik über das Clipboard in Zeichenprogramme oder Wortprozessoren zu exportieren. Dadurch kann der Merkmalseditor auch beim Erstellen von Dokumenten als WYSIWYG-Editor[1] eingesetzt werden.

'WYSIWYG = What You See Is What You Get.

Fegramed stellt eine komfortable Suchfunktion zur Verfügung. Hierbei hat man eine größere Auswahl an Suchmodi:

- Suche nach atomaren Knoten oder Kanten oder beidem
- Teilstring- oder Ganzwortsuche
- Beachten oder Ignorieren von Groß-/Kleinschreibung
- Suche nach den Vorkommen einer Koreferenz

Dabei kann der Startpunkt der Suche durch Mausklick bestimmt werden oder auch wahlweise die ganze Struktur durchsucht werden. Falls eine Suche erfolgreich war, kann sie durch Tastendruck wiederholt werden, um das nächste Vorkommen zu finden.

In der Macintosh-Version von Fegramed kann man ganze Merkmalsstrukturen (oder ausgewählte Teile) an den Drucker schicken, um bei besonders schwierigen Fällen oder als Dokumentation eine Papierkopie der Struktur in der Hand zu haben.

Fegramed wurde in ANSI-C implementiert und ist dadurch im Kern hochportabel. Neben einer Macintosh-Version gibt es zwei X-Windows Version; eine, die auf dem X-Toolkit direkt aufbaut und nicht mehr gewartet wird und eine, die auf MOTIF-Bibliotheken aufbaut. Diese Version wurde schon erfolgreich auf mehreren Hardwareplattformen wie Sun Spare, HP PA und IBM RS 6000 übersetzt und zum Laufen gebracht.

Die sorgfältige Implementierung von Fegramed sorgt dafür, daß das Programm bei sehr hoher Performanz nur sehr wenig Systemressourcen verbraucht und daher auch auf kleinen Maschinen (wie z.B. Macintosh) problemlos auch mit großen anderen Programmen zusammen benutzt werden kann. Als Stand-alone-Anwendung mit einem einfachen und klaren Interface kann der Merkmalseditor leicht an andere Applikationen angeschlossen werden.

Wegen der einfachen Portabilität und der hohen Leistungsfähigkeit wird Fegramed schon in vielen anderen Institutionen benutzt, so z.B. die Firma Cray Research, die im Auftrag der Europäischen Gemeinschaft ALEP entwickelt, die TFS Gruppe an der Uni Stuttgart, die VerbMobil Gruppe der Uni Hamburg und das BiLD Projekt der Uni des Saarlandes.

3.2 Chart-Display

Zum Parsermodul gehört auch ein interaktives graphisches Chartdisplay, das ein überaus wichtiges Hilfsmittel zum Debuggen von Grammatiken als auch für die Entwicklung von Parsingstrategien darstellt. Die graphische Darstellung der Chart zeigt die Abarbeitungs-reihenfolge des Parsingprozesses

Die Chart einer Instanz des Parsermoduls kann sowohl während als auch nach dem Parsingvorgang angezeigt werden. Die Anzeige des Verlaufs des Parsens kann wichtige Hinweise auf die zeitkritischen Punkte einer Analyse geben, da man die Reihenfolge des

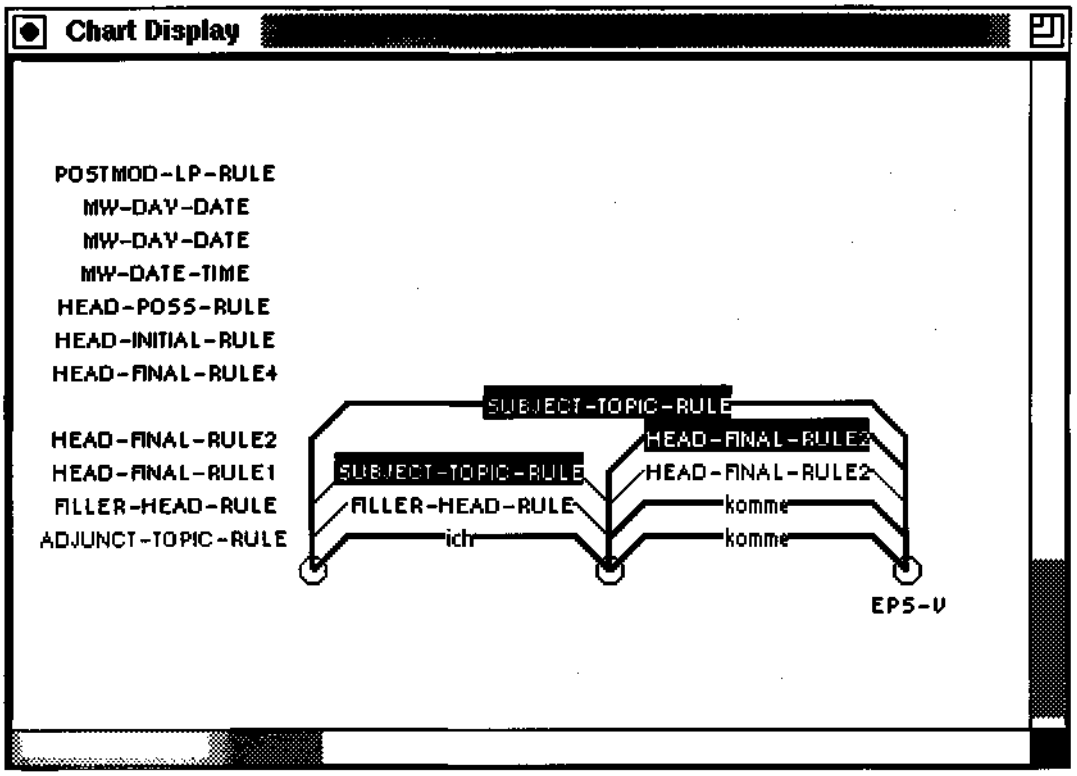


Abbildung 3.4: Das Chartdisplay während des Parsens

Konstituentenaufbaus im laufenden Prozeß direkt beobachten kann. Anschließend kann die Chart auf ungewünschte oder fehlende (Teil-)Analysen untersucht werden oder einfach die Verarbeitungsstrategie des Parsers oder der Grammatik betrachtet werden.

Mit der Maus können Kanten zur weiteren Verarbeitung durch andere Werkzeuge ausgewählt werden, was die Suche nach Fehlern in der Grammatik immens erleichtert. Man kann so direkt Verarbeitungsschritte nachvollziehen, die unkorrekterweise entweder fehlgeschlagen sind und keine neue Kante erzeugt haben oder im Gegenteil eine falsche neue Kante in die Chart eingefügt haben.

Es können auch verschiedene Kantenmengen interaktiv ein- oder ausgeblendet werden, um die Übersichtlichkeit zu erhöhen. So können aktive oder passive Kanten, Kanten mit bestimmten Start- oder Endpunkten, einzelne Teilbäume, die Parsingresultate und aktive Kanten mit bestimmter Wachstumsrichtung ausgewählt werden. Besonders die Auswahl einer Konstituente mit dem zugehörigen Teilbaum, der ja die Ableitungsgeschichte darstellt, wird häufig gebraucht.

Da das Parsermodul mehrere funktional verschiedene Parserinstanzen erlaubt, können auch die per Maus anzuwählenden Menüs dem jeweiligen Parser angepaßt werden. Die Menüpunkte erhalten dabei spezielle, durch den Entwickler des Parsers implementierte Funktionalität; diese ist nicht auf Grundfunktionen eingeschränkt. Die Parser des Disco-Systems, die ja mit Unifikation und Featurestrukturen arbeiten, bieten im Auswahlmenü für Konstituenten z.B. die Möglichkeit, die Featurestruktur mithilfe des Featureeditors Fegramed anzuzeigen, um die Mechanismen der internen Verarbeitung nachvollziehen zu können.

3.3 Datenbank für Testsätze des Deutschen

3.3.1 Motivation

Die Entwicklung des Diagnostik-Werkzeugs DiTo hat sich aus der Notwendigkeit ergeben, den Entwicklern von natürlichsprachlichen Systemen Testmengen zur Verfügung zu stellen, anhand derer sie die Performanz der Systeme kontrollieren können. Durch die modulare Architektur natürlichsprachlicher Systeme besteht die Möglichkeit, einzelne Komponenten und Teilkomponenten separat zu testen und entsprechend zu modifizieren. Bisher existierten leider oftmals entweder nur unzureichende Testmengen, oder ausführlich erarbeitete Testdaten wurden nur für ein spezielles System angelegt, so daß sie nicht auf andere Systeme übertragbar waren. Aus dieser Motivation entstand die Idee, eine Testmenge zu entwickeln, die im Bereich der *Syntaxkomponente* als Kontrollmenge eingesetzt werden kann. Mit dem Diagnostik-Werkzeug DiTo wurde nun der Versuch unternommen, einen linguistischen Datenkatalog für das Deutsche zu erstellen, mit dem Ziel, möglichst alle wesentlichen Bereiche der deutschen Syntax anhand von Beispieldaten *systematisch* abzudecken. Der Katalog sollte folgende Aufgaben erfüllen:

- **Debugging** : Fehler der syntaktischen Verarbeitung können leichter lokalisiert und benannt werden, wenn eine empirische Grundlage für die Fehlerdiagnose zur Verfügung

steht.

- **Konsistenzhaltung:** Anhand der Daten kann z.B. überprüft werden, ob sich die Abdeckung eines syntaktischen Bereiches verändert hat, nachdem die Grammatik des Systems an anderer Stelle modifiziert wurde.
- **Monitoring der Systemperformanz:** Die Verarbeitungsperformanz kann durch die regelmäßige Anwendung der Testdaten gezielt kontrolliert werden.

Von anderen Arbeiten in diesem Bereich unterscheidet sich DiTo in zwei Punkten:

- Die Beispieldaten sind systematisch erstellt und nicht aus Texten extrahiert, um eine möglichst genaue Kontrolle über die Testdaten zu haben.
- Die Beispieldaten sind mit phänomen-spezifischen Annotationen versehen, die den jeweiligen Syntaxbereich weitgehend charakterisieren und mit allgemein-syntaktischen Informationen, die die Oberflächenstruktur der Sätze beschreiben.

Bei der Wahl der Annotierungen wurde davon ausgegangen, daß der Datenkatalog nützlicher sein wird, wenn Beispielsätze zu einem speziellen syntaktische Phänomen herausgezogen werden können und gleichzeitig mit Hilfe der allgemein-syntaktischen Informationen die Systeme anhand detaillierter und leicht kontrollierbarer Daten auf Genauigkeit innerhalb der syntaktischen Analyse hin überprüft werden können [114] [115].

3.3.2 Realisierung

Bei der Entwicklung des Diagnostik-Werkzeugs ging es zum einen darum, eine Methode zu entwerfen, um die notwendigen linguistischen Daten zu sichten, auszuwählen und zu klassifizieren. Auf der anderen Seite standen Überlegungen, wie die Daten effizient abgelegt und benutzbar gemacht werden können.

Zuerst war es notwendig, eine Liste der syntaktischen Phänomene des Deutschen zusammenzustellen und die Merkmale ausgewählter Syntaxbereiche anhand einschlägiger linguistischer Literatur zusammenzutragen. Mit Hilfe der syntaktischen Merkmale der einzelnen Bereiche wurden dann systematisch Beispielsätze erarbeitet. Um bei dem angestrebten Einsatz der Testmenge kontrollieren zu können, ob alle und nur die richtigen Sätze verarbeitet werden, umfaßt die Datensammlung auch eine möglichst systematische Zusammenstellung ungrammatischer Sätze. In Zusammenarbeit mit anderen Forschungsgruppen[2] ergaben sich für die Bereiche "Verbrektion", "Koordination", "Funktionsverbgefüge" und "Relativsätze" insgesamt 600 grammatische und 800 ungrammatische Sätze [77] [82].

Um eine effiziente Speicherung und Nutzung der Daten zu ermöglichen, wurde das linguistische Material in einer relationalen Datenbank organisiert. Dies hat mehrere Vorteile: Die Testmenge ist klar strukturiert und erleichtert so den Zugriff auf die Daten.

[2]Institut für Computerlinguistik an der Universität Koblenz/Landau, Institut für Computerlinguistik an der Universität des Saarlandes, Institut für angewandte Informationsforschung in Saarbrücken

Außerdem bietet der relationale Aufbau gute Voraussetzungen, die Datenbank um neue Syntaxbereiche zu erweitern und die Daten konsistent zu halten. Und schließlich kann auf die Bedürfnisse der Benutzergruppen stärker eingegangen werden, da den interessierten Gruppen verschiedene logische Sichten auf die Daten zur Verfügung stehen.

Bei der Suche nach einer geeigneten Datenbanksoftware, die als *public domain* auf Unix-, Macintosh- und DOS-Rechnern laufen sollte, damit das Diagnostik-Werkzeug einer möglichst breiten Forschungsgemeinschaft zur Verfügung gestellt werden kann, fand sich zum damaligen Zeitpunkt kein kommerzielles System, das die gestellten Anforderungen erfüllte. Aus diesem Grund wurde eine prototypische Datenbank in *awk* implementiert, die die Basisfunktionen "speichern", "editieren", "abrufen" bereitstellt. Damit waren die für die Benutzung und Erweiterung des Diagnosewerkzeugs notwendigen Funktionen vorhanden, wenn auch die Benutzeroberfläche unbefriedigend blieb. Als prototypische Datenbank wurde diese awk-Version an den linguistischen Forschungsinstituten in Koblenz und Berlin zu Testzwecken installiert. Als Resultat dieser Testphase ergab sich die Notwendigkeit einer kompletten Reimplementierung in C, um die Portabilität, Effizienz und Robustheit des DiTo-Systems zu verbessern, eine wohldefinierte Schnittstelle zum DiTo-Kern in Form einer C-Funktionsbibliothek zu schaffen und die Anfragesprache und Funktionalität der Benutzerschnittstelle zu erweitern.

Eine erste C-Version, die über die Funktionalität der awk-Version hinausgeht, ist fertiggestellt. Zudem existiert ein Entwurf für die Benutzeroberfläche, der den einzelnen mitarbeitenden Gruppen zur Begutachtung vorgelegt werden kann und als Ausgangspunkt für die Realisierung benutzer-unterstützender Werkzeuge für die Erweiterung des DiTo-Sytems eingesetzt werden kann.

3.3.3 Ergebnisse

Die Entwicklung des Diagnostik-Werkzeugs hat in den Bereichen Datenerarbeitung, Datenorganisation und Datennutzung Erfolge erzielt.

Wie die erfolgreiche Kooperationen mit dem Institut für Computerlinguistik in Saarbrücken und Koblenz sowie dem Institut für angewandte Informationsforschung (IAI) in Saarbrücken zeigen, findet die Idee, systematisch Beispieldaten für linguistische Phänomene des Deutschen zu erarbeiten, die Zustimmung anderer Forschungsgruppen. Zudem haben Wissenschaftlerinnen der LDV/CL Trier, des Forschungsschwerpunkt Allgemeine Sprachwissenschaft in Berlin, der GMD/IPSI-Gruppe in Darmstadt und Sharp Laboratories in Oxford auf dem DiTo-Workshop im März 1993 in Saarbrücken ihr Interesse an der DiTo-Arbeit bekundet und stehen weiterhin mit der DiTo-Gruppe in Saarbrücken in Kontakt, um Verbesserungsvorschläge auszutauschen und die Erweiterung des Datenkataloges zu unterstützen.

Mit den DiTo-Daten wurde außerdem ein Beitrag zur Korpus-Forschung geleistet, indem die Daten der vier Synatxbereiche auf der CD-ROM "ECI / MCI" (Multilingual Corpus 1) der European Corpus Initiative verfügbar gemacht wurden.

Schließlich werden z.Z. Ergebnisse der DiTo -Testdaten-Systematik sowie der syntaktischen Annotationen im LRE-Projekt TSNLP ("Test Suites for Natural Language Proces-

sing"), das zum Ziel hat, Testsuites für Deutsch, Englisch und Französisch zu erarbeiten, mit einbezogen. Sowohl die Daten selbst als auch Teile der Annotationsparameter werden direkt in die TSNLP-Arbeit einfließen.

Bei der Organisation der Daten in der relationalen Datenbank wurde das konzeptuelle Schema mehrfach modifiziert, bis letztendlich eine stabile modulare Klassifizierung der linguistischen Daten erreicht wurde. Die relationale Struktur der Testdaten unterstützt die Erweiterung der Testdaten um neues Datenmaterial und erleichtert selbst nachträgliche Modifikationen des Entity-Relationship Modells. Die Grundlagen, die wir mit dem Entwurf des konzeptuellen Schemas geschaffen haben, werden die Basis zum weiteren Ausbau der Testsuites im TSNLP-Projekt bilden. Außerdem wird die Datenbanksoftware, die für DiTo entwickelt wurde, zum Aufbau der linguistischen Datenbank für die drei Sprachen Englisch, Deutsch und Französisch verwendet werden.

Was die Nutzung der linguistischen Testdaten betrifft, so ist die Anbindung des Testwerkzeugs an natürlichsprachliche Systeme unproblematisch, da das DiTo-System über eine systemunabhängige Schnittstelle verfügt. Für Testläufe können über einfache Datenbankanfragen Testdaten so zusammengestellt werden, daß gezielt einzelne Aspekte der Syntaxkomponente überprüft werden können.

3.4 Arbeiten an SADAW

Um mittel- und langfristig zu einer breiten Datenbasis im Bereich der Morphologie und des Lexikons zu gelangen wurde im Rahmen des DISCO-Projektes damit begonnen, die SADAW Datensammlung für modernere Verarbeitungsmethoden und -Umgebungen verfügbar zu machen. Diese Arbeiten wurden in Zusammenarbeit mit der Gruppe von Dr. Block bei der Siemens AG, München, und dem Institut für Maschinelle Sprachverarbeitung an der Universität Stuttgart durchgeführt und von diesen Institutionen teilweise mitfinanziert.

SADAW, "Saarbrücker deutsches Analyse-Wörterbuch", ist ein im Sonderforschungsbereich 100 'elektronische Sprachforschung' (1972-1986) erarbeitetes Lexikon mit syntaktischer und morphologischer Information. Es wurde auf der Basis des Wahrig Deutschen Wörterbuchs erstellt und umfaßt ca. 140.000 Lemmata.

Das Hauptinteresse an SADAW bestand und besteht darin, auf seiner Basis ein weitgehend theorieunabhängiges (und damit wiederverwendbares) Lexikon zur maschinellen Sprachverarbeitung zu gewinnen.

Der Zugang zu der Datensammlung war zunächst durch das veraltete Datenformat stark erschwert. Das Lexikon wurde deshalb in ein moderneres, leichter lesbares Format überführt. Es wurde eine Untersuchung durchgeführt, deren Ziel es war den Inhalt der Datenbank bezogen auf einzelne Datenfelder und Attribute festzustellen und die Zuverlässigkeit und Korrektheit der Daten stichpunktmässig zu überprüfen, um dadurch eine Bewertung der (Wieder-)Verwendbarkeit der Daten zu erhalten [37].

Das wichtigste Ergebnis dieser Analyse war, daß SADAW für die maschinelle Sprachverarbeitung grundsätzlich sehr brauchbar sein könnte und als umfangreiche Grundlage für ein lexikalische morphologisch-syntaktische Datenbank dienen kann. Es zeigte sich jedoch,

daß in manchen Bereichen ein relativ hoher Grad an Fehlerhaftigkeit zu finden war, der vermutlich auf fehlende Werkzeuge zur Konsistenzprüfung bei der Erstellung der Datenbank zurückzuführen ist.

In einem weiteren Schritt wurde deshalb damit begonnen, SADAW von korrupten Einträgen zu reinigen, Redundanzen zu beseitigen und eine konsistente Notierung einzuführen, um damit zu einer kontinuierlichen Verbesserung zu gelangen.

Dazu wurden zunächst die Kodierung der offenen Wortklassen (Nomen, Verb, Adjektiv) aufgespalten in morphologische und syntaktische Information, wobei die syntaktische Information im wesentlichen lemma-basiert, die morphologische stamm-basiert kodiert ist.

Zur Fehlerbeseitigung wurde speziell eine kleine C++-Bibliothek entwickelt, mit deren Hilfe die Daten (relativ) benutzerfreundlich und sicher sequentiell durchgesehen und verändert werden können. Mithilfe dieser selbsterstellten Werkzeuge wurden umfangreiche Berichtigungsarbeiten an SADAW durchgeführt.

Der Schwerpunkt der Arbeit lag dabei bei den Verben, bei denen die meiste syntaktische Information kodiert ist und wo die Datenanalyse auch die meisten Inkonsistenzen festgestellt hatte. In diesem Teilbereich des Lexikons wurde u.a. die (morphologisch wichtige) Länge abtrennbarer Präfixe berichtigt, mehrfach kodierte Lemmata wurden gesichtet und entfernt, und die Perfektbildung und die Reflexivkodierung wurden auf der Basis von Heuristiken halbautomatisch überarbeitet. Bei Adjektiven wurde die Kodierung der 'attributiven vs. prädikativen' Verwendbarkeit überprüft und bei den Nomina vor allem korrupte Einträge entfernt.

Um über diese Heuristiken hinaus zu weiteren Verbesserungen zu gelangen, wurden die Daten punktuell für einige Attribute und Datenfelder (präpositionale Valenz, Kasusvalenzen, adverbiale Ergänzung) manuell mit der kommerziell vertriebenen lexikalischen Datenbank CELEX abgeglichen. Dabei zeigte sich zum einen, daß die in CELEX und SADAW enthaltenen Lemmata sich nur zu ca. 60 % überschneiden, zum anderen daß die in CELEX enthaltenen Annotationen, in einigen wichtigen Punkten ärmer sind, als die SADAW-Kodierungen. Des weiteren wurde beim Abgleich festgestellt, daß die CELEX-Annotationen keineswegs immer zuverlässiger sind, als die in SADAW. Die Synthese der Informationen führte entsprechend zu einem im Vergleich zu den Ausgangsdaten wesentlich verbesserten Kodierung.

Die Arbeiten an SADAW bieten damit die Grundlage für ein großes, theorieübergreifendes und wiederverwendbares Lexikon des Deutschen, das als vielseitige Wissenquelle in der maschinellen Sprachverarbeitung eingesetzt werden kann. Weitere Arbeiten in diesem Bereich (Einbindung in eine Datenbank, weitere Korrekturen der Annotationen, Spezifikation von Schnittstellen zu theoriespezifischen Lexika) sind allerdings unbedingt erforderlich und sollen im Rahmen des Anschlußprojektes PARADICE durchgeführt werden.

Kapitel 4

Diskurs und Dialog

4.1 Sprechakttheorie

Die beste Art, Dialogsysteme für autonome kooperierende Agenten (z.B. Roboter oder Software-Agenten) zu testen, ist die Messung ihrer Fähigkeit, natürlichsprachliche Äußerungen mit entsprechenden Handlungen zu verknüpfen. Demgegenüber entstand die Sprechakttheorie aus ganz anderen Gründen, d.h., um vom philosophischen Standpunkt aus der Problematik der angeblich modalen oder völlig unlogischen Aspekte beobachteter natürlicher Sprache [6, 147] Rechnung zu tragen. Ihr eigentlicher Beitrag besteht darin, daß sie natürlichsprachliche Äußerungen als Handlungen im Kontext ansieht anstatt als kontextfreie Darstellung eines Zustandes. Zu einer Zeit, zu der die Künstliche Intelligenz sehr mit den Themen Inferenz, Planung und Aktion beschäftigt war, verbuchte die Charakterisierung von Sprachgebrauch als geplante Aktion große Akzeptanz und half, weitere Probleme bei der Erforschung von Sprache wie z.B. kommunikative Implikatur [138] zu lösen.

Natürlichsprachliche Äußerungen als Kommunikationshandlungen einzustufen bringt große Vorteile: Kommunikationshandlungen dienen zur langfristigen Speicherung von Konversationen [146], als Generierungsbasis [101], zur Top-Down-Vorhersage von Funktion und Struktur von Äußerungen [2] und als Basis für allgemeines Planen und Handeln [4].

Dieses auf eine gewisse Weise von linguistischen Themen abstrahierte und zuerst in [157] vertretene Sprechaktkonzept ist seitdem mehr und mehr in dem Bereich der Künstlichen Intelligenz verbreitet, der sich mit autonomen, kooperativen Softwaresystemen befaßt. DaherwaresauchdieTheoriederWahlfürDISCO.

Im Laufe des Projekts wurden Lösungen für eine Reihe wesentlicher Forschungsprobleme erarbeitet:

- die Anpassung von Methoden zur Erkennung 'oberflächennaher' Sprechakte innerhalb typisierter Merkmalformalismen für eine umfassende, HPSG-gestützte Grammatik des Deutschen;

- die Erstellung eines für Dialoge mit mehreren Agenten geeigneten Modells der Überzeugungen der Agenten;

- zusätzliche Unterstützung für Terminplanungsanwendungen: Planung und raumzeitliches Schließen.

Die zuerst genannten Ergebnisse wurden in beiden Versionen des COSMA-Systems eingesetzt (vgl. Kapitel 5). Die anderen Ergebnisse wurden in der zweiten COSMA-Version benutzt, die in diesem Kapitel beschrieben wird.

4.2 Integration mit der Grammatik

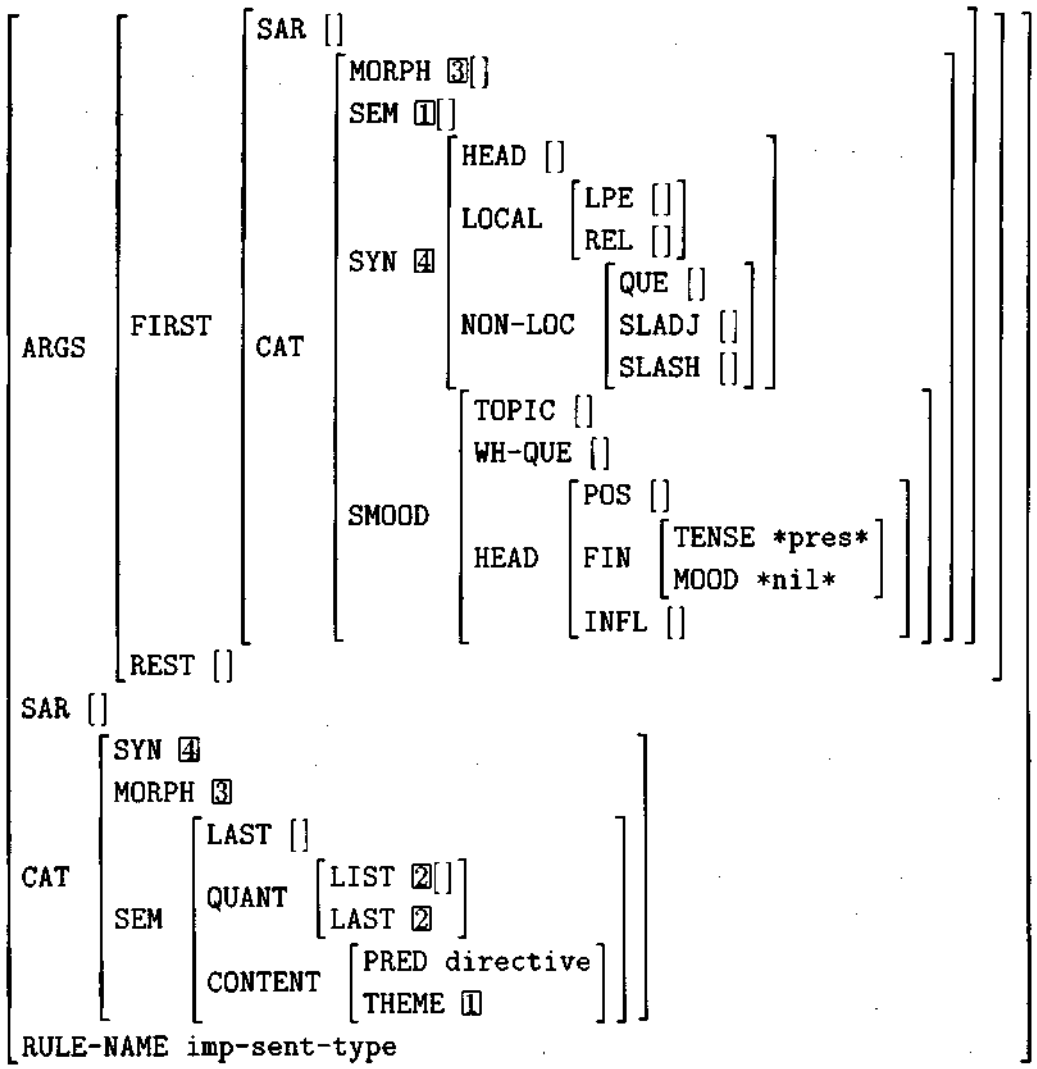
In [52] war bereits ausgearbeitet, wie im Englischen Sprechakte durch Musterabgleich *(pattern matching)* erkannt werden können. Entsprechende Regeln stellen einerseits die linguistischen Merkmale dar, die konventionell mit einem bestimmten Sprechakt assoziert sind und andererseits diesen Sprechakt selbst. Es blieb dann zu zeigen,

- daß diese Methoden für die deutsche Sprache auch anwendbar sind,
- wie sie im Rahmen von *TDL* zu formulieren sind,
- daß sie mit einer HPSG-Grammatik zusammen funktionieren und
- daß sie mit einer umfangreichen Grammatik funktionieren.

Die Sprechaktregeln wurden in *TDL* entwickelt. Sie stellen eine optionale Erweiterung zu der in DISCO benutzten Grammatik dar. Die Grammatik selbst ist von den Grammatikentwicklern so erweitert, daß sie bei der Verarbeitung alle für die Sprechakterkennung wichtigen syntaktischen Merkmale unter bestimmten Schnittstellenmerkmalen sammelt. Die Sprechaktregeln benutzen diese Informationen, die sich unter einem festen Pfad befinden. Dies hat den Vorteil, daß die Sprechaktregeln nicht das komplette Resultat der Syntax-Analyse betrachten müssen, sondern alle wichtige Information, die sie benötigen, an einer festen Stelle vorfinden. Neben dem Aspekt der modularen Entwicklung von Grammatik und Sprechaktregeln ermöglicht diese Lösung eine größere Wiederverwendbarkeit der Regeln. Denn es genügt bei Einsatz derselben Regeln in einer anderen HPSG-Grammatik, dieselben Merkmale bereitzustellen.

Das Ableiten eines möglichen Sprechaktes, also die Inferenzschritte, erfolgen durch die Unifikation der Regeln mit dem Resultat des Syntaxparsers. Da mehrere Regeln bei gleicher Eingabe unterschiedliche Ergebnisse liefern, ist ihre Anwendung nur mit einer Präferenzverteilung möglich, d.h. jede Regel erhält ein relatives Gewicht, das die Priorität der Anwendung einer Regeln bestimmt.

Folgende Regel identifiziert imperative Sätze und leitet *directive,* d.h. Befehle und Bitte her.



In der ersten COSMA-Version wurde gezeigt, daß diese Methode für das Deutsche innerhalb des getypten Merkmalsformalismus und der für DISCO maßgeblichen HPSG-Grammatik funktioniert. Das Fehlen der Inferenzkomponente gemäß der Methode von [52] erwies sich jedoch als nachteilig, weshalb die zweite (auf RHET gestützte) Terminplananwendung erstellt wurde.

Bei der Dialogverarbeitung sind Folgerungen über die Intentionen von Sprecher und Hörer sehr wichtig. Die Übertragung der Dialogverarbeitung auf mehrere Agenten macht Sprechaktdarstellungen erforderlich, die die Überzeugungen und die Intentionen meherer Dialogteilnehmer beschreiben können.

4.3 Gruppen als Agenten

Eines der Ziele der Dialogmodellierung ist die Darstellung von Planen und Handeln während eines Dialogs aus der Sicht eines Gesprächsteilnehmers - und zwar auch für Situationen, in denen die anderen Teilnehmer nicht gänzlich erfaßt werden können bzw. in denen die entsprechende Personengruppe so groß ist, daß nicht alle Anwesenden modelliert werden können. Man betrachte beispielsweise das folgende Gespräch aus der Perspektive eines Teilnehmers, dem gerade einfällt, daß das nächste Arbeitstreffen am Dienstag stattfinden soll:

Jan : also, sollen wir das Treffen am
Donnerstag dann ausfallen lassen.

Max : am Dienstag
Ich : das am Dienstag
Eva : ja, sollten wir.
Jan : gut, dann fällt's aus.

Hier schlagen der erste Sprecher und ein weiterer Teilnehmer eine Terminänderung vor, die von Eva und Jan bestätigt wird, wobei hier auch die Teilnahme weiterer Personen möglich ist.

Der Schwerpunkt unserer Arbeit über Gruppenagenten lag auf den Auswirkungen von Gesprächsverhalten auf die Vorstellung des Gesprächsteilnehmers von der Situation. Im Gegensatz zur früheren Dialogforschung interessieren hier jedoch diejenigen Situationen mit mehr als drei Teilnehmern, denn es ist äußerst unwahrscheinlich, daß jemand in einer Gesprächsrunde von zehn und mehr Personen die Äußerungen und Ansichten eines jeden anderen Teilnehmers dem jeweiligen Sprecher zuordnen kann oder womöglich permanent im Kopf hat, wer genau anwesend ist.

Traum [151] beispielsweise faßt Sprechakte als gänzlich gemeinschaftliche Handlungen der Gesprächsteilnehmer auf. Demnach werden gemeinschaftliche Sprechakte nicht nur gemeinsam produziert, sondern sie haben bezeichnenderweise auch zumindest dem Namen nach gemeinschaftliche Auswirkungen: wenn sie vollständig sind und im Ergebnis dennoch keine von allen geteilte Absicht oder Ansicht zur Folge haben, dann ist dies möglicherweise Höflichkeit, Unaufrichtigkeit (vgl. [137]) oder anderen sozialen Funktionen zuzuschreiben.

Für diese Auffassung der Sprechaktverarbeitung ist es nötig, die Grundzüge der Thematik der gemeinsamen Ab- und Ansichten zu betrachten. Obgleich diese Thematik mittlerweile recht gut erforscht wurde, sind die von den logischen Grundprinzipien gelieferten Analyseergebnisse jedoch tendenziell relativ komplex und liefern weder leicht verständliche Notationen noch plausible computerlinguistische oder kognitive Modelle. Kurz, normative Analysemethoden sind nicht unbedingt auch deskriptiv.

Hiervon abgesehen stößt man noch auf eine Reihe anderer Schwierigkeiten:

- bei mehreren Teilnehmern steigt die Anzahl der 'Glaubenswelten' (modale Einordnungsmuster) sprunghaft an;

- ist der tatsächliche Kenntnisstand bzw. Grad an Wissen eines anderen Teilnehmers unbekannt (wie z.B. bei einem Online-Gesprächspartner der Fall), dann kommt eine Vielzahl von Gesprächssituationen in Betracht;
- bei der Behandlung von großen Gruppen oder Organisationen ist eine Massen- bzw. Sammelmodellierung unbedingt erforderlich.

Nehmen wir als Beispiel an, Sie sind der Ansicht, daß der Staat über Ihr Einkommen von letztem Jahr Bescheid weiß. Sie glauben nun nicht, daß jeder einzelne Staatsbeamte dies weiß, sondern daß derjenige Finanzbeamte, der hierfür zuständig ist und dies wissen will, diese Kenntnis besitzt. Demnach würden wir typischerweise annehmen, daß jemand, der in seiner Funktion als Finanzbeamter Kontakt mit Ihrem Buchhalter aufnimmt, dieses Wissen besitzt. Wir wollen uns nun von den dies ermöglichenden Kommunikationskanälen abwenden und nur die Verfügbarkeit der Schlußfolgerung - idealerweise im Hinblick auf die Erfordernisse der Gesprächsmodellierung - weiter im Auge behalten.

Im nächsten Abschnitt wird eine für die Verarbeitung von teilnehmerintensiven Sprechakten nötige Methode der Informationsrepräsentation beschrieben, die Gruppen als eigenständige Agenten ansieht und die Sprechakte selbstverständlich und direkt auf diese wirken läßt.

4.3.1 Gruppenagenten und Propagierung von propositionalen Einstellungen

Grundlage unseres Modells ist der Gruppenagent. Diese 'Agenten' stellen Gruppen dar, können jedoch, wie Personenagenten auch, Ansichten ($B_{agent}P$ genannt) und Absichten ($G_{agent}P$) haben, die ihnen direkt zugeordnet sind. Demnach kann man sie sich als eine Art Zwischentyp zwischen Gruppen von Einzelagenten und prototypischen Gruppenmitgliedern vorstellen. Sie unterscheiden sich von einfachen Gruppen auf drei grundlegende Arten: Erstens sind sie Intensionsstrukturen, die auch bei gleicher Ausdehnung unterschiedlich sein können (wie z.B. wenn die Mitglieder einer Marketingabteilung Dienstags abends ein Volleyballteam bilden); zweitens sind ihnen propositionale Einstellungen direkt und möglicherweise abweichend von irgendwelchen ihren Mitgliedern oder Subagenten zugeordneten Einstellungen zugeordnet (vgl. 4.3.4); drittens ist die Existenz von Untergruppen keine hinreichende Bedingung für die Existenz von Subagenten - einige betriebseigene Volleyballteams sind als soziale Einheiten eindeutig Agenten des Betriebs, während es andere nicht sind.

Obwohl wir nicht in der Lage sind, detaillierte psychologische Thesen aufzustellen, glauben wir, daß Strukturen dieser Art mit dem kompatibel sind, was wir über die linguistische und kognitive Handhabung von Aggregaten in anderen Zusammenhängen wissen.

Die Gruppenagenten dienen zur Darstellung von dauerhaften sozialen Gruppen und vorübergehenden Gruppen von Gesprächsteilnehmern.

Im Folgenden werden die Beziehungen zwischen Agenten in Diagrammen dargestellt (vgl. Abb. 4.1). Hier stehen die abgerundete Rechtecke für Agenten und die sie verbindenden Linien für die Subagent-Relation (E): Die Agenten umfassen das System selbst

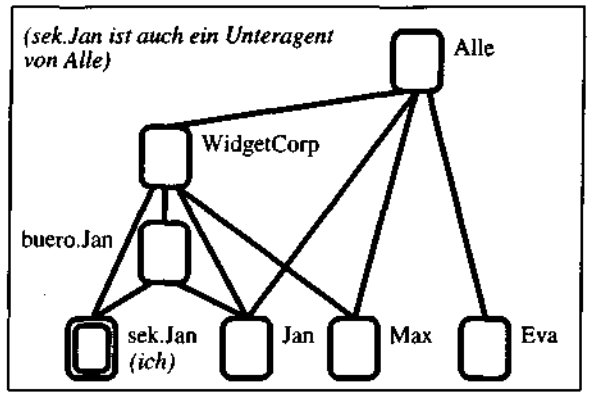


Abbildung 4.1: Agenten

(Sek.Jan), den Chef des Systems (Jan), Jans Büro, eine mitarbeitende Person Max, die sie beschäftigende Firma (WidgetCorp) und eine weitere 'Randperson' , Eva, die Mitglied der WidgetCorp ist.[1]

Da die systemeigene Darstellung der Welt behandelt werden soll, ist zu beachten, daß das doppelt umrandete Rechteck das Ich-Modell des Systems darstellt und die ganze Abbildung das System selbst - dieser Punkt bedarf hier jedoch keiner ausführlicheren Behandlung, solange der Informationsfluß zwischen den beiden frei ist.

Die Errechnung des Transfers von propositionale Einstellungen zwischen den Agenten basiert auf einem Näherungsmodell - teils, zur Vereinfachung des Rechenprozesses, teils weil die Verfügbarkeit einer ausreichenden, eine exakte Lösung ermöglichenden Datenmenge nicht der Normalfall ist. Dasselbe Modell (mit Parametervariation im Wissensbereich) wird auf alle vom System den Agenten zugeordneten propositionale Einstellungen angewandt; in der obigen Implementierung können dies sowohl Ansichten als auch Absichten sein.[2] Anders als auf herkömmliche Glaubenslogiken gestützte Darstellungen führt unser Modell keine verschachtelten Kontexte ein, solange sie nicht explizit in Äußerungen enthalten sind (wie dies bei einem Satz wie 'Aber Sigi ist der Ansicht, daß das Treffen auf jeden Fall stattfinden soll' der Fall wäre), auch wenn erweiterte Argumentationsprozesse dies möglicherweise widerspruchsfrei tun.

In diesem vereinfachten Modell ist die Propagierung von propositionalen Einstellungen schwach, wie auch durch die semantische Beziehung zu den jeweiligen Agenten festgelegt. Im Idealfall würde diese Einschätzung dem sozialen Weltwissen und der Information über die Aufgabe der Gruppen entspringen. In der derzeitigen Implementierung wird sie näherungsweise erbracht durch eine einfache Einordnung der Gruppenagenten, ihrer Rolle und

[1]Im weiteren Verlauf werden die den Agenten zugeordneten Einstellungen mit kleinen Rändern innerhalb der Rechtecke dargestellt und deren Propagierung mit dünnen gebogenen Pfeilen.

[2]Da unser Modell keinen analytischen Charakter hat, wollen und brauchen wir keinen 'Wissensbegriff': dem System fehlt der direkte Zugang zu empirischer Wahrheit und sozialem Einvernehmen. Das System macht daher auch keine kognitive Differenzierung zur Bewertung von Argumenten vor dem Hintergrund eines unabhängigen Rationalitätsbegriffs.

der Gesprächsthemen in den ontologischen Bereich. Verzögerungen und Zufälle werden nicht modelliert. Ausgegangen wird von einer den folgenden Einschränkungen unterliegenden Propagierung aller relevanten propositionalen Einstellungen zwischen den Agenten:

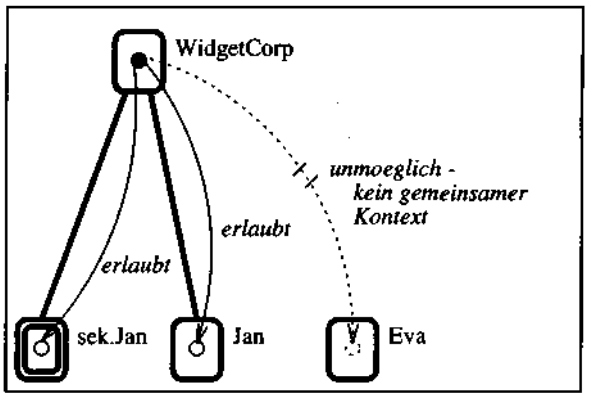


Abbildung 4.2: Gemeinsame Kontexteinschränkungen

1. Propositionale Einstellungen werden nur zwischen Superagent und Subagent propagiert (und umgekehrt). D.h. es findet nur dann eine Kommunikation zwischen Agenten statt, wenn ein gemeinsamer sozialer Kontext existiert. Bei Bildung neuer sozialer Gruppen können natürlich neue Gruppierungen eingeführt werden.

   Somit können in Abb. 4.2 die der WidgetCorp zugeordneten Ansichten auf die Subagenten Jan und seine Sekretärin propagiert werden, nicht jedoch auf Eva als Nichtmitglied (und damit Nicht-Subagent) der WidgetCorp[3]. Wenn für propositionale Einstellungen ein direkter Beweis vorliegt, werden sie als gefüllter Kreis dargestellt, andernfalls nur durch eine Kreislinie. Gestrichelte Linien sind 'negiert' - sie werden nicht durch eine durchgezogene Linie dargestellt, weil sie eine der genannten Einschränkungen verletzen.

2. Die Propagierung von propositionalen Einstellungen ist nur ein 'Default' (die spezielle Wortwahl der Defaultlogik braucht hier nicht zu interessieren). Ein direkter Beweis für die Zuordnung einer gegenteiligen propositionalen Einstellung zu einem Agenten hindert die Propagierung externen Ursprungs. Diese Eigenschaft ist für die Modellierung von Unehrlichkeit, Verhandeln, Einlenken und von atypischen Gruppenmitgliedern im allgemeinen ausgesprochen wichtig.

   Abb. 4.3 zeigt eine solche Blockade. In diesem Fall kann unser Modell von Jan keine Absicht (Kreis) von Jans Büro bekommen, weil diese einer anderen Absicht (Quadrat) entgegenläuft, für deren Zuordnung zu Jan es einen vorrangigen unabhängigen Beweis gibt.

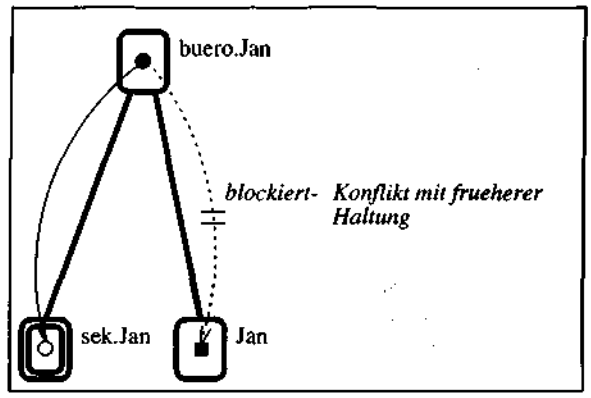


Abbildung 4.3: Propagierung als Default

3. Das System kann nie annehmen, daß seine eigenen propositionalen Einstellungen automatisch zu einem Superagenten aufwärtspropagiert werden. Das System muß seine eigenen Aktionen immer noch selbst planen - und ausführen -, auch wenn der Pfad für die Aufwärtspropagierung von propositionalen Einstellungen die Wirkung von externen Agenten gemäß ihrer verschiedenen Kommunikationsziele unabhängig modelliert. Demnach ist das System - Sek.Jan - in Abb. 4.4 daran gehindert, einfach anzunehmen, daß seine eigenen Ansichten von seinem Arbeitgeber geteilt werden - obwohl die Ansichten von Mitarbeitern propagiert würden, sofern sie anderweitig konsistent wären. (Einige Menschen handhaben diese Einschränkung weniger restriktiv.)

4. Sogenannte nonce-Gruppen werden dynamisch eingeführt zur Darstellung von vorüber-

[3]Um die zufällige propositionale Einstellungspropagierung einzuschränken, erscheint diese Forderung angemessen, da indirekter Transfer nur über einen einzelnen Agenten stattfinden kann, der zugleich transitiver Super- oder Subagent beider Endagenten ist. Damit wäre eine Propagierung auf dem sich ergebenden N-Pfad auch dann nicht möglich, wenn Jan und Eva einer der WidgetCorp ähnlichen Gruppe mit ähnlicher semantischer Domäne angehörten. Genausowenig ermöglicht die gemeinsame Mitgliedschaft in allen Gruppen den Transfer von Ansichten, da deren Relevanzfilter maximal restriktiv ist.

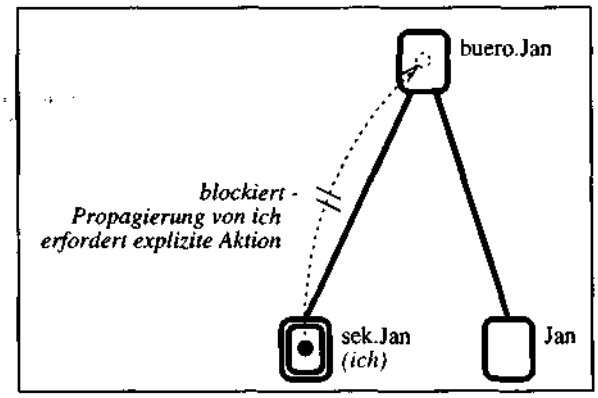


Abbildung 4.4: Die Sonderstellung des 'Ich' , Eigenpropagierung

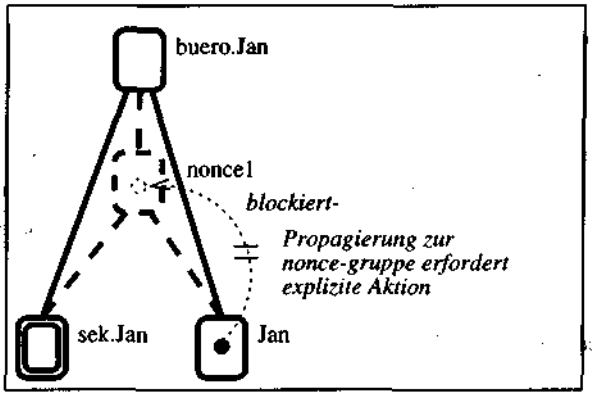


Abbildung 4.5: Redenotwendigkeit

gehenden Zusammenschlüssen von Teilnehmern in einer aktiven Gesprächssituation, übernehmen jedoch nie propositionale Einstellungen von ihren Subagenten, sondern müssen sie als Resultat beobachtbarer Aktionen erhalten. Dem liegt der Gedanke zugrunde, daß das System während der (aktiven oder passiven) Teilnahme am Gespräch in der Lage ist, die Bildung der 'gemeinschaftlichen Aufzeichnung' dieses Gesprächs [93] direkt mitzuverfolgen; diese Aufzeichnung besteht exakt aus den propositionalen Einstellungen, die der Gesprächsgruppe selbst zugeordnet werden sollen. Bei Konversationen sollte sogar ein neues 'unausgesprochenes Einvernehmen' aus der beobachteten Konversation hergeleitet werden und nicht nur die persönlichen Ansichten des Systems über die Ansichten anderer Teilnehmer.

Die Tatsache, daß Konversationsgruppen immer noch von Superagenten propositionale Einstellungen bekommen können, ermöglicht es, die Diskussion in einen sozialen Kontext einzuordnen, der gemeinsame Hintergrundvoraussetzungen liefert. Die Tatsache, daß deren Subagenten wiederum von ihnen diese propositionalen Einstellungen bekommen können, ermöglicht eine Modellierung der Informationsbeschaffung einzelner Teilnehmer, einschließlich des Systems, über die gemeinschaftliche Aufzeichnung ohne zusätzliche Vorrichtungen.

Abb. 4.5 zeigt diese Situation: Nonce1, die Gesprächsrunde, stellt ein gemeinsames soziales Konstrukt dar, das sich von unserer Vorstellung von Jans eigenen Ansichten unterscheidet. Dies erlaubt einen angemessenen Umgang mit der Situation, in der wir, Sek.Jan, Jan beim Lügen ertappen (oder sogar unterstützen).

Wichtigste Eigenschaft dieses Modells der propositionalen Einstellungszuordnung ist die Einführung nur von denjenigen Glaubenswelten, die unabhängig motiviert wurden durch identifizierte soziale Gruppierungen oder Aufzeichnungen von tatsächlichen Gesprächen, an denen das System teilnimmt. Dies verringert die Wahrscheinlichkeit, daß das System sich in irrelevante strukturelle Details verliert und verhindert besonders die für die klassischen normativen Modelle typisch komplexe Art der Verschachtelung von Überzeugungsmodellen. Die Zuordnung durch Annahme von Defaults ermöglicht auch die Darstellung von Indivi-

duen als Mitglieder mehrerer verschiedener Gruppen mit gegensätzlichen Ansichten, wobei diese Individuen nur diejenigen Ansichten übernehmen, die den als privat dargestellten gleich sind.

Daher unterscheiden sich auch die hier erzielten Resultate wesentlich von denen der klassischen Logik [4, 81, 5, 34]. Sie unterscheiden sich auch von anderen pfadbasierenden Algorithmen [18] durch die Bereitstellung von semantischen Relevanzbedingungen und dadurch, daß diese der Notwendigkeit gemeinsamer propositionaler Einstellungen Rechnung tragen durch direkte Zuordnung der propositionalen Einstellungen zu Gruppen anstatt durch Speicherung komplexer Datensätze des Inhalts 'Wer ist welcher Ansicht ? ' . Dies ermöglicht die Beschreibung und Verarbeitung von Konversationsmechanismen ohne Rückgriff auf verschachtelte Überzeugungsmodelle *(x* glaubt, daß *y* glaubt, daß...), gleichwohl können solche Strukturen für andere weniger gebräuchliche Kognitionsprozesse weiterhin erforderlich sein.

Der nächste Abschnitt behandelt die Anwendbarkeit dieses Einstellungszuordnungsmodells für die Implementierung der Verarbeitung von teilnehmerintensiven Sprechakten.

4.3.2 Teilnehmerintensive Sprechakte

Wie [151] gehen wir davon aus, daß ein Basis-Sprechakt im Grunde genommen durch eine Reihe von Äußerungen verwirklicht wird, die den Basisprozeß verkörpern. Dieses Modell erfordert, daß die Definitionen der Sprechakte selbst gesondert betrachtet werden und hochrangige Aktionen hervorbringen, die in Handlungen der nichtlinguistischen Domäne integriert werden können. Die Anwendung unseres Mechanismus zur Zuordnung von propositionalen Einstellungen zu mehreren Agenten ermöglicht eine weitere Vereinfachung durch die Definition von Sprechakten als gemeinschaftliche Aktionen, die direkt auf die zu modellierende Konversationsgruppe wirken.

Der folgende verallgemeinerte Handlungsoperator stellt einen einfachen Basis-Sprechakt dar:

$$\begin{aligned} & \mathsf{Inform}_a p \\ \textit{conditions}: \quad & \mathbf{B}_b p \wedge b \sqsubseteq a \wedge \mathsf{live}\, a \\ \textit{effects}: \quad & \mathbf{B}_a p \end{aligned}$$

Dieses 'Informieren' ist eine echte gemeinschaftliche Aktion. Agent *a* ist die Hapaxgruppierung, die alle Teilnehmer gemeinsam darstellt (das Prädikat 'live' verlangt, daß diese Gruppierung einer derzeit stattfindenden[4] Konversation entspricht). Obwohl der Individualagent *b* die Informationsquelle ist, wirkt sich die Handlung direkt auf unser Gruppenmodell aus. Von da aus ist die Abwärtspropagierung auf die einzelnen Teilnehmer eine Funktion des propositionalen Einstellungszuordnungsmodells und unterliegt den o.g. Restriktionen. (Das System nimmt tatsächlich an, daß im Kopf der einzelnen Gesprächsteilnehmer entsprechende Informationsaktualisierungen stattfinden.)

Die Richtigkeit dieser Formulierung stützt sich auf zwei Tatsachen: Erstens sorgt die den Basis-Sprechakt-Operator verwirklichende Basisstruktur dafür, daß der Inhalt erfolg-

[4] Unsere derzeitige Implementierung arbeitet allerdings mit *e-mail* statt mit gesprochener Sprache und muß mehrere aktive Dialoge zugleich verarbeiten.

reich befördert wird. Zweitens gilt: Wenn ein Sprechakt, der auf die Konversationsgruppe wirkt, voll verwirklicht und sauber begründet wurde, dann muß jeder Zuhörer, der eine dieser Wirkung gegenläufige Meinung hat, explizit handeln, um diese zu stoppen. D.h., wird die Annahme einer Nachricht angezeigt, bewirkt dies die Vermutung, daß der Empfänger deren Inhalt zustimmt. Es ist jedoch zu beachten, daß, wenn der Sprachakt unwidersprochen bleibt, dies lediglich heißt, daß die Gesprächspartner ihn als Teil der gemeinschaftlichen Aufzeichnung stehen lassen. Es heißt nicht, daß sie auch wirklich von dessen Inhalt überzeugt sind; die hier aufgestellten Regeln ermöglichen lediglich die Voraussage, daß die Teilnehmer den Inhalt des Sprechakts annehmen, wenn es keinen Beweis für das Gegenteil gibt.

Erfolgreiche Nachfragen wirken sich eher auf die Absichten als auf die Ansichten der Gruppe aus. Es ist wichtig, sowohl kommunikative als auch nichtkommunikative Gruppen einzubeziehen. Die im folgenden zuerst dargestellte Absicht ist nichtkomunikativ und besagt lediglich, daß die gewünschte Aktion durchgeführt werden soll. Es ist zu beachten, daß es, obwohl der (Gruppen-)Agent der gewünschten Aktion am Gespräch teilnimmt, keine dahingehende Einschränkung gibt, daß dieser Dialog den Fragenden nicht mitumfassen darf. Wenn wir $\mathbf{Bif}_{ap}$ für $B_a p \vee B_a \neg p$ und O für den Futuroperator schreiben, erhalten wir:

$$\begin{aligned} &\mathsf{Request}_a e \\ \textit{conditions}: \quad &\mathsf{agent}\, e \sqsubseteq a \wedge \mathsf{live}\, a \\ \textit{effects}: \quad &\mathbf{G}_a \vec{\Diamond}\, e \wedge \mathbf{G}_a \mathbf{Bif}_a \vec{\Diamond}\, e \end{aligned}$$

Die zweite Absicht bei den Wirkungen ist kommunikativer Art: Die Gruppe erhält die Absicht, herauszufinden, *ob* die gewünschte Aktion durchgeführt wird. Die Konsequenz hieraus ist, daß der Fragende einen Hinweis darauf erhält, ob der Wunsch Erfolg hatte; denn selbst wenn man von Kooperation ausgeht, führen Absichtskonflikte und Planungsbeschränkungen manchmal zur Abweisung einer erfolgreich kommunizierten Anfrage.

4.3.3 Verarbeitung eines N-Pfad-Sprechakts

Jetzt können Sprechakte wie die obigen in den auf Planung und Folgerungen basierenden Algorithmen kommunizierender intelligenter Agenten dargestellt werden. Da ein Gespräch unvorhergesehene Ereignisse umfassen kann, müssen solche Agenten in der Lage sein, auf veränderte Umstände auch zu reagieren und dürfen nicht nur völlig von Vorausplanung abhängen. Bei der Ankunft eines jeden Sprechaktes aktualisiert der Agent seine Ansichten und Absichten; diese bilden die Basis für weitere Aktionen. Dies ist nicht nur der Schnittstelle zwischen Aufgabe und Dialog angemessen, sondern auch absolut erforderlich für den Basisprozeß.

Eine Anwendung für N-Pfad-Sprechakte in einem teilnehmerintensiven Umfeld ist die Terminplanung, bei der Dialogsysteme als künstliche persönliche Terminkoordinatoren für menschliche Agenten dienen. Das von uns implementierte System COSMA arbeitet auf diesem Gebiet. Es modelliert ein Paar bestehend aus menschlichem Benutzer und machinellem Sekretariatsassistenten als Gruppenagent, bei dem Ansichten über Termine von beiden Individuen aus auf- und abwärtspropagieren und Absichten für Termine vom Mensch hinauf

zum Paar und vom Paar hinab zum künstlichen Terminkoordinator propagieren.

Am Anfang dieses Beispiels hat das Dialogsystem (Sek.Jan) die Rolle eines künstlichen Terminkoordinators für einen menschlichen Agenten (Jan). Diese beiden bilden die Gruppe 'Jans Büro'. Jan schickt Sek.Jan über Email eine Nachricht bezüglich eines bereits existierenden Termins:

Jan: Treffen mit der Hardwaregruppe absagen. [—>sec.Jan]

COSMA interpretiert diese Eingabe indem es zuerst eine Hapaxgruppe, Noncel, für den neuen Dialog bildet mit:

$$Jan,\ \text{sec.Jan} \sqsubseteq nonce1 \sqsubseteq office.Jan$$

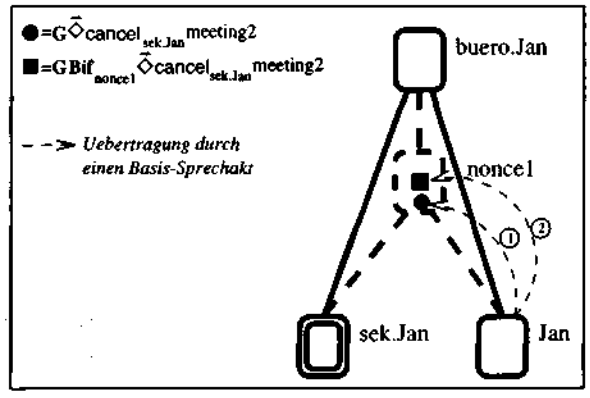


Abbildung 4.6: Anfrage

Alle Mitglieder einer Hapaxgruppe übernehmen von dieser Ansichten und kommunikative Absichten. Die Interpretation der ersten Äußerung als Sprechakt ist:

$$\text{Request}_{\{Jan,\ sec.Jan\}}\,\text{cancel}_{sec.Jan}\,meeting2$$

Diese Interpretation wird nach der Methode von Hinkelman auf Verträglichkeit mit dem Kontext untersucht und ergibt eine akzeptierbare Lesart. Ihre Auswirkungen auf die Gruppe stellen sich wie folgt (und 1, 2 in Abb.4.6) dar:

$$\mathbf{G}_{nonce1} \vec{\Diamond}\ \text{cancel}_{sec.Jan}\,meeting2$$
$$\mathbf{G}_{nonce1}\mathbf{Bif}_{nonce1} \vec{\Diamond}\ \text{cancel}_{sec.Jan}\,meeting2$$

Wenn alle Eingaben verarbeitet sind, kontrolliert das System seine eigenen Absichten, um herauszufinden, welche Aktionen es womöglich durchführen wird. Es wird keine unmittelbaren eigenen Absichten finden, dafür aber zwei, die es übernommen hat. Da das System an dem noch andauernden Gespräch mit Jan teilnimmt, erhält es die kommunikative Absicht der Hapaxgruppe Bif... (3 in Abb. 4.7). Es erhält weiterhin die Absicht,

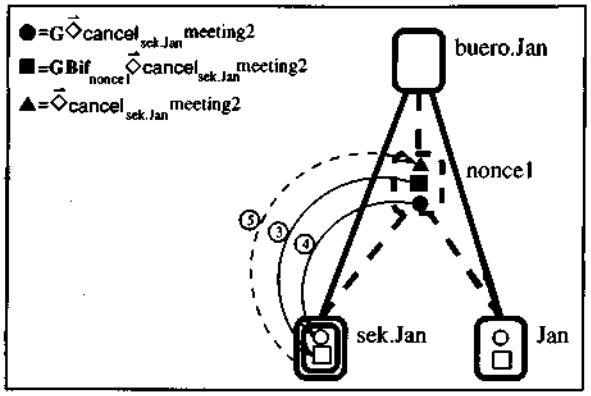


Abbildung 4.7: Antwort

sicherzugehen, daß die Terminstornierung auch tatsächlich stattfindet (©) [5]. (Ein weniger entgegenkommender Agent als COSMA würde nichtkommunikative Absichten nicht direkt von der Hapaxgruppe erhalten, sondern die Stornierungsabsicht indirekt über Jans Büro bekommen. Die Implementierung könnte jedoch sehr ähnlich aussehen, da der Pfad für indirekte Weitergabe herauskompiliert werden kann, wenn die Hapaxgruppe gebildet wird.)

Das Dialogsystem erhält somit die folgenden Absichten:

$$\mathbf{G}_{sec.Jan}\vec{\Diamond}\ \mathsf{cancel}_{sec.Jan}meeting2$$
$$\mathbf{G}_{sec.Jan}\mathbf{Bif}_{nonce1}\vec{\Diamond}\ \mathsf{cancel}_{sec.Jan}meeting2$$

Diese Absichten bilden die Eingabe für den Planungsprozeß. Die erste Absicht kann im gegenwärtigen Kontext erreicht werden, indem der Terminkalender geöffnet und dann ein gespeicherter Unterplan für die Absage von Terminen ausgeführt wird, der die Modifizierung des Datenbankeintrags und Benachrichtigung der Teilnehmer umfaßt. Unser reaktiver Algorithmus erlaubt die freie Einbindung von kommunikativen Plänen in Domänenaktionen und umgekehrt.

Nachdem diese Handlungsabfolge gefunden ist, weiß das System, daß *O* ... gilt. Es kann somit Inform_{nonce1} ($\vec{O}$ ...) planen, womit es die zweite Absicht erfüllt (©). Der Output für das zweite Ziel lautet:

$$\mathsf{Inform}_{nonce1}(\vec{\Diamond}\ \mathsf{cancel}_{sec.Jan}meeting2)$$

sek.Jan: Ok, ich sage ihn ab. [—>Jan]

Schließlich muß das System die Ausführung seines Plans vollenden, um die erste Absicht durch Aktualisierung des Terminkalenders und Benachrichtigung der Teilnehmer zu erfüllen. Der Benachrichtigungsschritt erfordert die Konstruktion einer passenden Konversationshapax, diesmal ein Abkömmling von WidgetCorp selbst (in gesprochenen Dialogen erfordert dies neben der Aufstellung der erforderlichen internen Datenstrukturen, die

[5]Beachte, daß das System bei einer entsprechenden Nachfrage nun darauf zurückgreifen könnte, daß Jan dieses Absichten auch hat; dies ist jedoch kein Teil des Algoritmus zur Sprechaktinterpretation selbst.

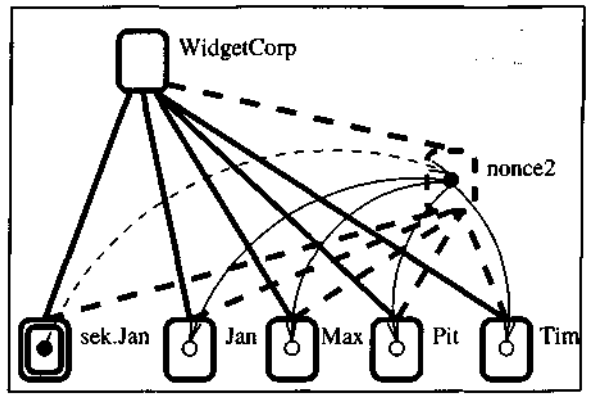


Abbildung 4.8: Informieren der Gruppe

Adressaten zu treffen und zu grüßen; bei der Kommunikation via Email ist analog nur die Zusammenstellung eines passenden Kopfs für die Emailnachricht erforderlich). Wie in Abb. 4.8 gezeigt initiiert das System dann eine weitere eigene Benachrichtigungsaktion:

$$\mathsf{Inform}_{nonce4}(\neg\vec{\Diamond}\ meeting2)$$

Diese kann wie folgt verbalisiert werden:

sec. Jan:

Das Treffen am Montag, 13.02., um 15.00 Uhr fällt aus.

[—>sec.Jan, Jan, Max, Eva, Tim]

4.3.4 Diskussion

Eine wichtige Eigenschaft des im DISCO-Projekt entwickelten Gruppenagentenmodells ist sein Skalierbarkeit. Obwohl die Zahl der Agenten in einem nicht alltäglichen Weltmodell groß sein kann, führen wir nur Kontexte ein, die tatsächlichen Objekten entsprechen, von denen das System weiß. Im Einzelnen entsprechen die verwendeten Gruppenagenten entweder sozialen Gruppierungen oder Aufzeichnungen von aktuellen Gesprächen. Individuelle Sprechaktdefinitionen müssen sich, wenngleich sie für alle Agenten des Dialogs gelten, auf mindestens zwei Agenten beziehen.

Im Gegensatz zu normativen Modellen verlangt unser Sprechaktmodell nie die Modellierung der einzelnen Adressaten. Natürlich ist ein Dialog typischerweise von dem Wunsch bestimmt, die Ansichten des Adressaten zu ändern, unser System kann diese Updates jedoch jederzeit bei Bedarf durchführen. So lange die Erstellung detaillierter Partnermodelle an sich nicht nötig wird, bleibt somit der Aufwand für die Planung und Beantwortung von Sprechakten bei steigender Zahl der Gesprächsteilnehmer nahezu gleich.

Damit ist die Ausweitung der Sprechaktmodellierung auf teilnehmerintensive Umfelder gelungen, was einen Schritt über andere sprechaktbasierte Modelle [38, 98, 21, 151] hinaus bedeutet. Dabei wurde gleichzeitig die Komplexität der Schnittstelle zwischen Aufgabe und Dialog reduziert.

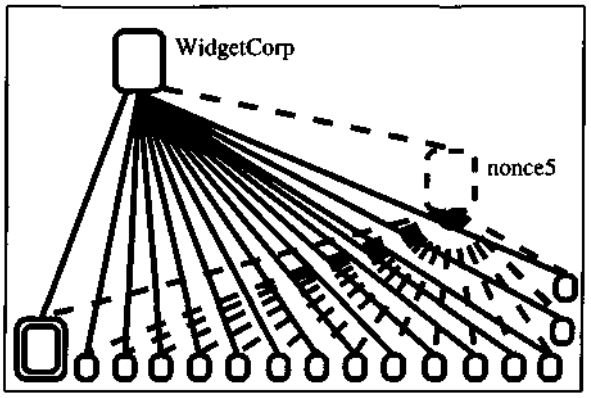


Abbildung 4.9: WidgetCorp dehnt sich aus

Man kann dem immer noch entgegenhalten, daß unser Modell nicht in der Lage ist, die detaillierte 'spy novel' Argumentation konventioneller Glaubenslogik zu unterstützen, die zurückverfolgt ob *F* glaubt, daß *C* glaubt, daß *p* gilt, und ob gemeinsame Ansichten wirklich gegenseitig sind oder nicht. Darauf kann dreierlei geantwortet werden:

- Problemlösung durch Nachdenken ist immer eine Option.
- Es sind Konversationsmechanismen entstanden für die Bewältigung der kognitiven Unzulänglichkeiten des Menschen.
- Auch wenn ein vollständiges System Zugang zu einem idealen normativen Dialogmodell hätte, würde es von der parallelen Benutzung eines weniger präzisen und deskriptiveren Modells profitieren.

Gruppenagenten sind eine Alternative zu normativen Glaubenslogiken, die eine Reihe interessanter sozialer und kommunikativer Phänomene direkt erfassen. Mit ihrer Hilfe können wir Basis-Sprechakt-Definitionen für den Fall von vielen Agenten sauber und präzise neu formulieren. Das Maß an Planungsabstraktion, das hieraus folgt, scheint gut geeignet für die Anforderungen an intelligente kommunikative Agenten, die in einem Umfeld mit vielen menschlichen Agenten arbeiten.

4.4 Weitere dialog-unterstützende KI-Aktivitäten

Als Implementierungsbasis fuer die oben genannte Arbeiten dient RHET, ein Wissensrepräsentationssystem auf Horn-Klausel-Basis, das an der Universität von Rochester entwickelt worden ist [99]. Durch eine Schnittstellenspezifikation zwischen Horn-Klausel-Prädikaten und Lisp-Funktionen können die Vorteile sowohl logischer als auch funktionaler Programmierung benutzt werden. Dazu sind auch Planerkennung, Plan- und Aktionsdarstellungen und temporales Schliessen vorhanden. Um aber ein Gesamtsystem für Dialog im Rahmen verteilter Terminplanung herzustellen, waren weitere KI-Aktivitäten nötig.

4.4.1 PISA

Der Planer PI SA [59] ist als Fortgeschrittenenpraktikum im Projekt DISCO entstanden. Ziel ist es gewesen, ein Werkzeug zu entwickeln, das eine domänenunabhängigere Diskursbehandlung ermöglicht. Nicht nur die ehemals festen Dialogscripts, sondern auch die gesamte Verarbeitung profitieren von der durch Planungsmethoden verbesserten Flexibilität und Unabhängigkeit vom Einsatzbereich.

Als Grundlage des Umgangs des Planers mit der Wissensbasis dient der *Situationskalkül* [97]. Repräsentiert werden dabei globale Situationen, die bei Updates nicht-monoton, also auch destruktiv, verändert werden. Die Verwendung des von RHET angebotenen Kontextmechanismus, der Wissen hierarchisch und unter Vererbungsaspekten anordnet, läßt die Darstellung mehrerer getrennter Situationen zu und bietet außerdem effizienten Zugriff auf unterschiedliches Wissen verschiedener Agenten.

Zum Generieren und Ausführen von Plänen sind Formulierungen von Aktionen, Aktionsbedingungen und Plänen nötig. Mit Hilfe des Typsystems von RHET wird deren allgemeine Spezifikation sowohl für den Planer als auch für die Diskursverarbeitung direkt zugänglich gemacht. Instanzen dieser allgemeinen Typen bilden dann die eigentlichen Aktionen und Pläne.

PI SA ist unabhängig von COSMA implementiert und läßt sich daher beliebig im Rahmen seiner Möglichkeiten einsetzen. Allein die Definition von Aktionen und Prädikaten innerhalb der Wissensbasis machen den Planer zum Einsatz in einem Anwendungsgebiet fähig. Besonders interessant ist natürlich das beliebige Vermischen von Aktionen verschiedener Domänen, wie dies z.B. mit Sprechakten [138] und Blocksworld-Aktionen [133] innerhalb der Testszenarien möglich gewesen ist.

Bei **PI** SA minimiert sich die Schnittstelle mit COSMA auf einen Horn-Klausel-Aufruf mit Angabe des Planziels und einem Verweis auf die Ausgangssituation. Beide Module laufen im gleichen Prozeß und greifen über die gleichen Funktionen auf das Wissen zu. So kann auch der Planer die Funktionalität des Kalküls der Diskursverarbeitung teilen, wie z.B. zeitliche und lokale Schlußfolgerungen (Abschnitt 4.4.2) und Modaloperatoren, die als vordefinierte Prädikate zugänglich sind. Die Ergebnisse des Planungsprozesses werden der Dialogverarbeitung auch wieder über die Wissensbasis vermittelt. Diese Vereinfachungen ermöglichen eine variable Aufrufstruktur innerhalb der Verarbeitung, was verschiedene Anwendungen des Planers denkbar macht.

In Bereichen, die sich mit Glauben und Wünschen beschäftigen, ist die Eigenschaft, komplexe Ziele ausdrücken zu können, wünschenswert. Neben Zielbeschreibungen mit der Standard *AND*-Verknüpfung aus STRIPS [44] erlaubt PI SA auch *OR* und *NOT*. Die Negation wird durch einen Normalisierungsprozeß sogar auf allen Ebenen erkannt und auf die Literalebene reduziert, wo sie dann schließlich als *negation as failure* bei Anfrage der Datenbasis oder als explizite Negation bei der Planung behandelt wird. Zusätzlich existieren Operatoren in der Zielspezifikation von PI SA, die das Backtrackingverhalten des Planers steuern können.

Die COSMA-Dialogverarbeitung modelliert System- und Agentenwissen innerhalb der Kontextstrukturen von RHET. Auch PI SA macht Gebrauch von diesen und kann sich so

auf verschiedene Glaubensräume einstellen. Je nach Bedarf kann das System so für sich selbst oder für andere Agenten, beispielsweise zur Hilfestellung oder zur Vorhersehung von Reaktionen, planen. Dank der Implementierung, insbesondere der Fähigkeit zur Teilinstanziierung (constraint-basierter Ansatz) von Aktionen, erstellt PI SA auch *joint plans*, d.h. Pläne, in denen veschiedenste Agenten gemeinsam eine Rolle spielen. Zusammen mit dem in RHET vorhandenen Planerkennungsmodul entsteht so eine flexible Reaktion auf die sich rasch ändernden Situationen im Diskurs.

4.4.2 Temporale und lokale Schlußfolgerungen

Terminplanung erfordert von Natur aus das Argumentieren über Ort und Zeit des Treffens - ein Punkt, der oftmals der wichtigste beim Versuch einer Terminfestlegung ist, wobei die Beschreibung dieser beiden Parameter in der Diskussion oftmals recht ambig ist. Auch wenn die planerische Komponente ausgearbeitet wurde, ist somit immer noch eine gewisse Fähigkeit zur Diskussion über Orts- und Zeitangaben erforderlich.

Temporale Schlußfolgerungen

Einer der Gründe für den Einsatz des RHET-System ist die Tatsache, daß es zusammen mit Tempos/Timelogic angewandt wird, d.h. einem System zur Argumentation über Zeiten wie auch schon in der bekannten Allen-Intervall-Logik dargestellt [3]. Es steht jedoch fest, daß dieses System zwei Unzulänglichkeiten im Vergleich zu der COSMA-Domäne aufweist, nämlich daß die verfügbaren Zeitangaben erstens relativ sind und sich zweitens nicht wiederholen. Die meisten Zeitangaben bei der Terminplanung sind demgegenüber jedoch entweder absolut festgelegt oder durch absolute (feste) Zeitvorgaben begrenzt, obgleich sie tendenziell sprachlich durch häufig wiederkehrende Ausdrücke wie 'Dienstag' ausgedrückt werden.

Der erste Mangel, nämlich das Fehlen von festen Zeitangaben, wurde direkt durch Timelogics Fähigkeit zum Argumentieren über Zeitdauer [timelogic-durations] behoben. Diese Seite von Timelogic selbst ist eine Weiterführung der Allen-Logik, knüpft mit einer großen Zahl von cross-inference rules an die Basisintervall-Logik an und drückt absolute Zeiten auf einen festen Standard-Referenzintervall bezogen aus. Das daraus resultierende System bekommt mit den von Timelogic gelieferten großen Datensätzen selbst Effizienzprobleme, ermöglicht jedoch eine relativ detaillierte Mischargumentation mit Mischeinschränkungen, die sowohl Intervallbeziehungen als auch zeitlich und örtlich absolute Einschränkungen umfassen. Dies hat weiterhin den Vorteil, daß die Grundplanrepräsentation Timelogic in RHET Zeitangaben verwendet; eine Codierung, die auch die transparente Anwendung von absoluten Einschränkungen auf Planungsstufen zuläßt.

Das Problem der Verarbeitung möglicherweise wiederkehrender sprachlicher Zeitausdrücke wurde separat bei der Übersetzung zwischen der semantischen Darstellung *(NLL)* und der pragmatischen Ebene (RHET) gelöst indem eine zweckgebundene objektorientierte Repräsentation der zugeordneten Zeitsequenzen benutzt wurde. Dieses System liefert direkte Implementierungen für einfache Intervalle, regelmäßig wiederkehrende Sequenzen

und als arbiträr errechnete Sequenzen und verwendet ein als 'CLOS multimethods' kodiertes Regelsystem zur gewünschten Normalisierung der Resultate von Zeitrechenoperationen. Anders als die absolute Zeitdarstellung in RHET wird dieses System nicht während einer online-Diskussion eingesetzt, sondern erst während der modularen Schnittstellenaufgabe. Bei einem schneller arbeitenden Argumentationssystem (natürlich schneller hinsichtlich LISP, nicht hinsichtlich der Darstellungsfähigkeit) wäre eine Integrierung dieser beiden Ansätze wünschenswert.

Lokale Schlußfolgerungen

Es ist ziemlich offensichtlich, daß die für Zeitargumente gedachte Allen-Intervall-Logik auf den Bereich der Ortsargumente ausgedehnt werden kann (vgl. z.B. [141]). In [46] wird z.B. eine Integrierung der Intervallogiken des Ortes und der Zeit in eine (topologische) Basislogik der Übertragung vorgeschlagen. In der aus dem COSMA-Bereich gewonnenen Bereichsanalyse wurden allerdings zwei vorherrschende Phänomene entdeckt, die der Repräsentation in dieser Art Modell nicht zugänglich sind. Das erste ist die Tatsache, daß Ortsangaben nicht nur in Terminplandiskussionen eingebracht werden, um mögliche Termine zu eliminieren (z.B. weil die Teilnehmer dann an unterschiedlichen Orten sein werden), sondern auch, um sie unter (typischerweise implizitem) Verweis auf die Anreisezeiten einzuschränken. Das zweite Phänomen ist die Tatsache, daß zwei (oder gelegentlich mehrere) Orte umfassende Regionen angegeben werden, z.B. DFKI mit zwei Standorten.

Um die Daten erfassen zu können, haben wir daher ein kleines System für die Ortsargumentation entwickelt, das die Intervallogik an möglicherweise mehrere Orte umfassende Regionen anpaßt und in Form von (Obergrenzen für) Fahrtzeiten innerhalb der Regionen eine Verbindung zur Zeitargumentation herstellt.

Die Kombination von zwei Orten umfassenden Regionen mit Fahrtzeitangaben erwies sich als überaus günstig, da sie typische Langstreckenfahrten mit öffentlichen Verkehrsmitteln gut modelliert. Definiert man z.B. dem Namen nach nicht zusammenhängende Regionen wie Frankfurt/Flughafen und Chicago O'Hare, so ermöglicht dies, die Tatsache auszudrücken, daß es (aufgrund der Flughafengebundenheit von Flugreisen) trotz der kürzeren Strecke länger dauert, von Saarbrücken nach Chicago zu reisen als von Frankfurt aus. Und das System könnte sogar unter Zuhilfenahme der Information, daß Saarbrücken und Frankfurt beide in Deutschland liegen, folgern, daß die Reise über Frankfurt eine plausible Strategie wäre. Obwohl für diesen topologischen Ansatz wesentlich weniger empirische Daten erforderlich sind, ist er damit auf vielfache Weise nützlicher als ein streng geometrisches Modell.

4.5 Arbeitspakete und Meilensteine

Nach der oben erfolgten Darstellung der sprechakttheoretischen und pragmatischen Seite des Disco-Projekts soll nun eine mit spezifischen Arbeitspaketen und Forschungsetappen verknüpfte Darstellung erfolgen. Die Arbeit auf diesem Gebiet begann im März 1992.

- Formalismus zur Wissensrepräsentation - Adaptierung und Schnittstellen. Das September 1992 erstellte COSMA-SYSTEM benutzte kein System zur Wissensrepräsentation. Die Schnittstelle zwischen dem natürlichsprachlichen System und dem anwendungsspezifischen Planer bestand aus Sprechakten und kompositioneller Semantik auf der inhaltlichen Ebene. Die Übersetzung erfolgte unter Verwendung der Sprache *NLL* auf der Basis von Kompilertechnologie. Im November 1992 wurde das System zur Darstellung rhetorischen Wissens RHET [99] wegen seiner Möglichkeiten zur Theorembestätigung, Zeitargumentation und Planerkennung adaptiert und mittels der Zebra-Kompilertechnologie mit einer Schnittstelle versehen.

- Sprechakt- und Pragmatiktheorie. Die wichtigsten Beiträge stammten aus dem Bereich der Erkennung oberflächennaher Sprechakte und der zuvor beschriebenen Gruppenagenten.

- Formalismus zur Wissensrepräsentation - Gebrauch für Diskurs und Benutzermodell. Zur Darstellung von Sprechakten und als Inferenzbasis für die oben beschriebene Sprechakterkennung und Planung wurde die von RHET gelieferte Planrepräsentation eingesetzt. Das Gruppenagentenmodell ist die Basis, auf der Modelle für Benutzer der Terminplandomäne und Benutzergruppen erstellt wurden.

Kapitel 5

Verteilte Terminplanung: Eine Anwendung der Kernmaschine

5.1 Grundlagen

Auf der Grundlage des linguistischen Kernsystems wurde ein System konzipiert, mit dem sowohl wissenschaftliche Fragestellungen als auch praktische Anwendungen bearbeitet werden können. Das System ist der kooperative Terminplaner COSMA (cooperative schedule management agent), der mit anderen COSMA-Systemen oder menschlichen Benutzern elektronisch vernetzt ist (vgl. Abbildung 5.1).

Terminvereinbarung ist ein Problem, dem viele Menschen und Organisationen täglich gegenüberstehen. Zu seiner Lösung bedarf es kooperativer Interaktion zwischen mehreren Teilnehmern. In DISCO wurden Termine betrachtet, an denen alle Adressaten teilnehmen sollten (z.B. Arbeitstreffen). Andere Einladungen und soziale Ereignisse, deren Termine in der Regel nicht verhandelt werden, waren somit ausgeschlossen.

Da Termine oft nach einer Reihe von Zwiegesprächen mit mehreren Dialogpartnern vereinbart werden (etwa telefonisch), sind gewöhnlich mehrere Kommunikationsrunden notwendig, bis alle Teilnehmer sich auf ein Datum und einen Ort einigen. Dies ist eine sehr zeitraubende Aufgabe, die nach Möglichkeit automatisiert werden sollte.

Theoretisch läßt sich diese Aufgabe vereinfachen, indem man eine zentrale Planung vorsieht, die Zugang zu den persönlichen Kalenderdaten hat. In der Praxis ist ein solcher Ansatz jedoch nicht sehr sinnvoll, da er den Teilnehmerkreis auf die Mitglieder bzw. Angestellten einer Organisation beschränken würde. Darüberhinaus sollte auch für die Angestellten in derselben Firma die vollständige Offenlegung personenegebundener Kalenderdaten aus Datenschutzgründen ausscheiden. Daher wurde in DISCO eine verteilte Lösung auf der Basis kooperierender Agenten bevorzugt. Ein solcher Ansatz ist besonders flexibel, da er es menschlichen Teilnehmern ermöglicht, in den Verhandlungsprozeß einzugreifen.

Die Interaktion von Menschen und Maschinen im selben Dialog erfordert die Verwendung natürlicher Sprache. Es kann nicht vorausgesetzt werden, daß alle Partner dieselbe Software benutzen; insbesondere werden viele Menschen noch einige Zeit lang gar keine

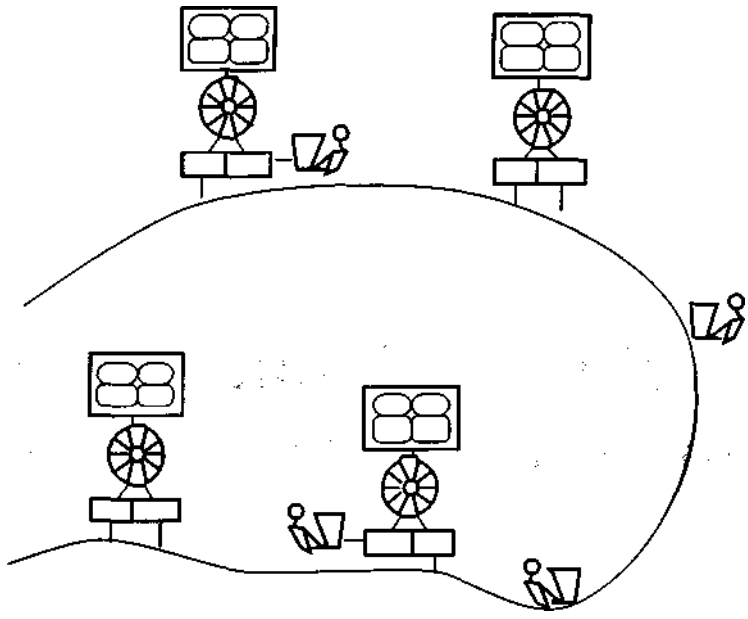


Abbildung 5.1: Eine dezentrale COSMA-Installation.

Software für Terminvereinbarung einsetzen. Ferner wird menschliche Sprache benötigt, um den Verhandlungsprozeß des Systems überwachen und gegebenenfalls eingreifen zu können.

Die Domäne kooperativen Planens hat viele attraktive Eigenschaften, sowohl für realistische Anwendungen von natürlichsprachlichen Systemnen, als auch für die Forschung:

- Es gibt Nachfrage für diese Anwendung.
- Die Anwendungsdomäne ist vergleichsweise einfach und kann durch bestehende KI-Techniken erfaßt werden.
- Die Anwendung ist "machbar".
- Die Anwendungsdomäne weist eine interessante Subsprache auf.
- Die natürlichsprachliche Funktionalität ist nicht kritisch; Nachrichten, die das System nicht verarbeiten kann, werden dem menschlichen Benutzer zugestellt. Der Benutzer kann eine Konfiguration wählen, in der keine Entscheidung ohne seine Zustimmung getroffen wird.
- Die Funktionalität kann in verschiedener Hinsicht erweitert werden. Das System kann z.B. darauf abgestimmt werden, den sozialen Status eines Agenten oder Reisezeiten in seine Inferenzprozesse einzubeziehen.
- Die Anwendungsdomäne kann in verschiedener Hinsicht erweitert werden. Viele logistische Probleme, wie etwa Flottenverwaltung oder Raumverteilung, erfordern eine sehr ähnliche Funktionalität.

- Korpora von Terminvereinbarungs-Nachrichten, mit denen das System verbessert und seine Leistung gemessen werden kann, sind leicht erhältlich.

- Die Anwendung läßt sich leicht vor Ort testen.

5.2 Kooperation zwischen DISCO und AKA-MOD

Die Konzeption von COSMA schließt Kompetenzen sowohl auf dem Gebiet natürlichsprachlicher Systeme als auch auf dem Gebiet verteilter kooperierender Agenten ein. Daher war eine Kooperation mit dem BMFT-geförderten Projekt AKA-MOD naheliegend und erwies sich als fruchtbar. Die in AKA-MOD entwickelten Verhandlungsstrategien für autonome kooperierende Agenten [143] bilden die Grundlage für ein System, das mit DISCO gekoppelt wurde. Auf diese Weise erhielt jeder Agent die Fähigkeit, seine Ziele mithilfe natürlicher Sprache zu kommunizieren.

Der von AKA-MOD entwickelte Planer folgt dem Ansatz von [94], indem er ein Protokoll auf der Grundlage endlicher Automaten implementiert, das auf einer festen Menge von Verhandlungsprimitiven beruht (z.B. arrange(), accept(), reject(), refine() etc.), die vollständige oder partielle Terminbeschreibungen als Argumente haben. Gewöhnlich wird ein Terminvereinbarungsdialog durch eine Nachricht an alle Teilnehmer eingeleitet, in der ein Terminwunsch mitgeteilt wird (arrangeO). Die Adressaten beantworten diese Nachricht (accept(), refine(), modify() oder reject()). Der Initiator entscheidet, ob alle Agenten einer Menge kompatibler Terminparameter (vor allem Zeit und Ort) zugestimmt haben, oder ob weitere Interaktionen mit einzelnen Teilnehmern erforderlich sind. Wenn Übereinkunft erreicht ist, versendet der Initiator die endgültige Terminbeschreibung an alle Teilnehmer als Bestätigung (confirmO), und die Agenten tragen sie in die jeweiligen persönlichen Kalender ein.

5.3 Übersicht über das COSMA-System

Jedes COSMA System fungiert als Sekretariatsassistent für einen menschlichen Benutzer, indem es Termine mit mehreren Teilnehmern vereinbart und bei Bedarf aushandelt.

Charakteristisch für den speziellen Anwendungsbereich von COSMA ist, daß auch menschliche Benutzer an einer Terminvereinbarung teilnehmen können, die über kein eigenes COSMA-System verfügen. Ihnen ist es möglich, in natürlicher Sprache an den Terminverhandlungen teilzunehmen. Damit stellt das COSMA-Szenario einen exzellenten Anwendungsbereich für das im Projekt DISCO entwickelte linguistische Kernsystem dar.

Jedes COSMA System besteht aus den folgenden zentralen Komponenten:

- Einem Terminplaner, der die Verwaltung der lokalen Kalenderdatenbasis übernimmt. Diese Komponente wurde von AKA-MOD bereitgestellt.

- Einem graphischen Zugangssystem zur Kalenderdatenbasis (vgl. Abschnitt 5.3.4).

- Dem linguistischen Kernsystem DISCO.

Zentrales Kommunikationsmedium für die verschiedenen Teilnehmer einer Terminverhandlung ist elektronische Post (e-mail). Daher wurde das DISCO-System um eine Schnittstelle zu normalen e-mail-Betriebssystemdiensten erweitert.

Zur Gewährleistung der adäquaten Integration in die COSMA-Anwendung wurde das linguistische Kernsystem um die folgenden Komponenten erweitert:

- Scanner: Analyse elementarer Textstruktur, Normalisierung anwendungsspezifischer Ausdrücke (z.B. Datums- und Zeitangaben wie "22.10.1993", "10:30h" etc.);
- *NLL* (in Zusammenarbeit mit dem Projekt ASL)
- Oberflächennahe Sprechakterkennung (siehe Abschnitt 4.2)
- Semantische Sprechaktverarbeitung und einfache Resolution: Transformation der *NLL*-Bedeutungsrepräsentationen durch (teilweise stark) anwendungsspezifische Inferenzregeln (siehe Abschnitt 5.3.2).
- Anwendungsschnittstelle (siehe Abschnitt 5.3.2)
- Template-basierte Generierung

Abbildung 5.2 zeigt im Detail die Architektur des COSMA Systems.

Die Kopplung des linguistischen Kernsystems an den Terminplaner wurde durch ein eigenes Modul realisiert, das DISCO-interne Bedeutungsrepräsentationen in Strukturen übersetzt, die von der Planungskomponente weiterverarbeitet werden können. Die Schnittstellensprache wurde in Zusammenarbeit mit AKA-MOD spezifiziert und entspricht in weiten Teilen der internen Repräsentationssprache des Terminplaners.

Die Übersetzungskomponente wurde gemäß der *NLL*-Grundidee realisiert, Transformation als Kompilationsschritt aufzufassen und durch geeignete Werkzeuge (Parser- und Printer-Generatoren) zu unterstützen, um die Erweiterung oder Anpassung an Veränderungen der Zielsprache zu vereinfachen.

Zusätzlich zu den neu integrierten Modulen wurden asynchrone Kommunikationskänale zur Planungskomponente und zum graphischen Zugangssytem geschaffen.

Die Grundlage für die Entwicklung der konkreten Systemarchitektur eines COSMA Systems stellte die objektorientierte DISCO Entwicklungsumgebung dar, die im Rahmen des Kernsystems entwickelt und bereits erfolgreich für die Entwicklung des linguistischen Kernsystems eingesetzt wurde (vgl. Abbschnitt 6.2).

Die Funktionalität des aktuellen Prototypen unterstützt u.a. die folgenden Verhandlungssituationen, die bereits in Form von Multi-Agenten-Dialogen zwischen mehreren Installationen des CosMA-Systems getestet wurden.

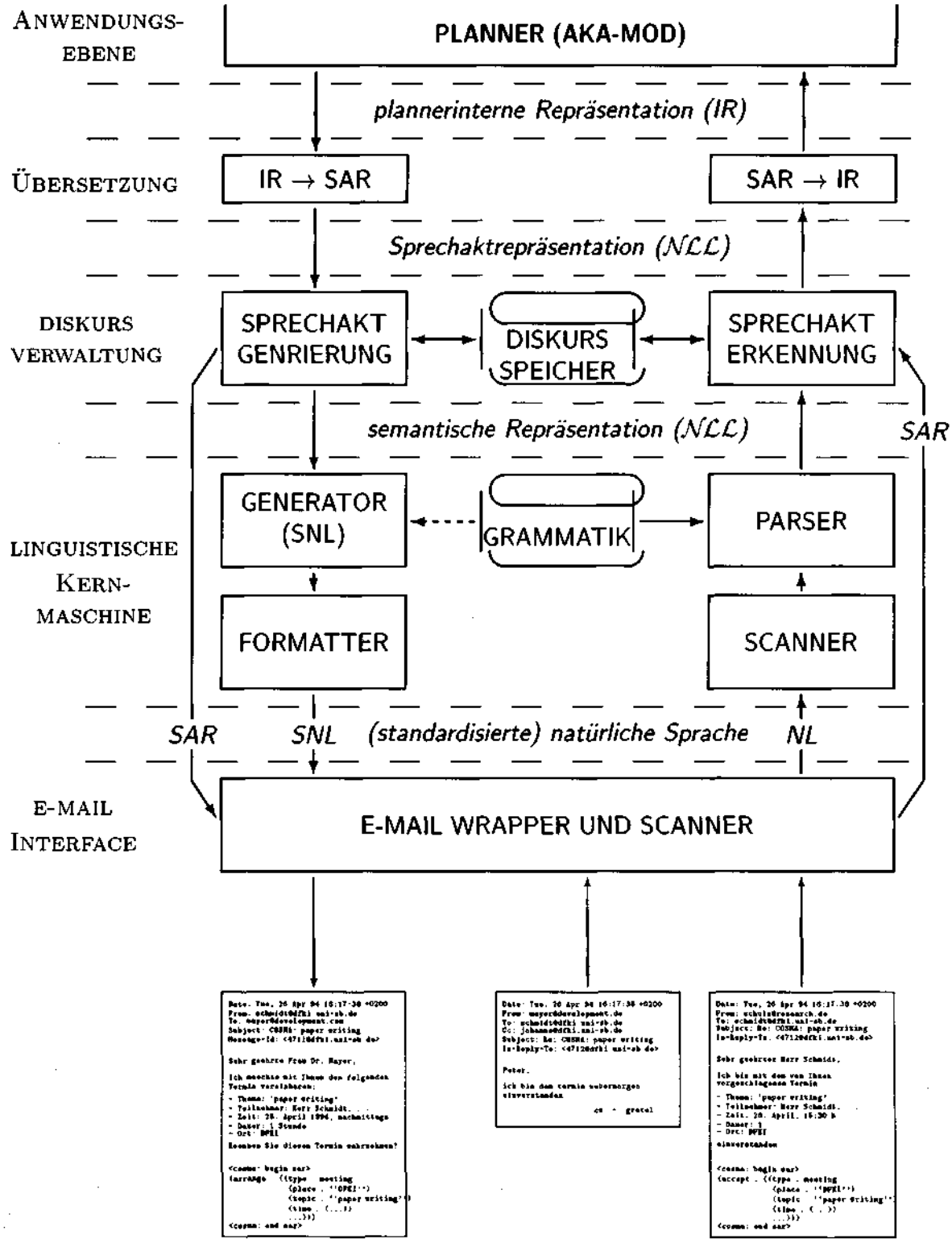


Abbildung 5.2: Die interne Architektur des COSMA Systems.

```
((from . "schmidt@dfki.uni-sb.de")
 (to . ("mayer@development.de" "schulz@research.de"))
 (message-id . 4711)
 (actions . ((arrange ((type . :meeting)
                       (topic . "Abschluszbericht")
                       (participants . ("schmidt@dfki.uni-sb.de"
                                        "mayer@development.de"
                                        "schulz@research.de"))
                       (time . ((year . 1994)
                                (month . :april) (day . 28)
                                (time-of-day . :afternoon)))
                       (place . "DFKI"))))))
```

Abbildung 5.3: Interne Repräsentation des COSMA-Terminplaners.

- Terminvereinbarung zwischen mehreren COSMA-Benutzern und Agenten, die über kein COSMA-SYSTEM verfügen. Die Verhandlungsinitiative kann sowohl durch die graphische Benutzerschnittstelle, als auch durch natürlichsprachliche elektronische Post ausgeführt werden. Im Zuge einer Terminverhandlung sind die beteiligten COSMA Systeme in der Lage, bestimmte Arten unter- und vage spezifizierter Anfragen zu verarbeiten.
- Hinzufügung und Änderung von Parametern (insbesondere Zeit, Dauer und Ort) von in der Verhandlung stehenden und bereits erfolgreich vereinbarten Terminen.
- Verschieben und Absagen von Terminen.

5.3.1 Verteilte Terminplanung: Eine Beispielinteraktion

Obwohl die eingeschränkte Terminvereinbarungssprache des COSMA-Planers nicht ausdrücklich für natürlichsprachliche Kommunikation konzipiert ist, erlaubt sie eine realistische Funktionalität in der Terminvereinbarung mit mehreren Teilnehmern. Die folgende Interaktion ist ein Beispieldialog mit drei Teilnehmern: Dr. Mayer, Herr Schulz und das COSMA-SYSTEM von Prof. Dr. Schmidt, das in einem authorisierten, automatischen Modus operiert:

(1) **Schmidts** COSMA **an Mayer und Schulz** (NL **und IR):**
"Ich möchte mit Ihnen den folgenden Termin vereinbaren: ..."
(vgl. Abbildungen 5.3 and 5.4)

(2) **Schulz an Schmidt** (NL):
"Ich kann an dem Termin übermorgen teilnehmen."

(3) **Mayer an Schmidt** (NL):
"Können wir uns am Donnerstag um 15:30 Uhr im DFKI treffen?"

(4) **Schmidts** COSMA **an Mayer und Schulz** (NL **und IR):**
"Alle Teilnehmer haben den folgenden Termin zugesagt: ..."

Schmidt benutzt die graphische Benutzerschnittstelle seines CoSMA-Systems, um ein einstündiges Treffen mit Mayer und Schulz für den Nachmittag des 28. April zu arrangieren. Sobald er die Eingabe abgeschlossen hat, reserviert der Terminplaner den gewünschten Zeitraum in seinem Kalender und benutzt das DISCO-System, um den Terminvorschlag als e-mail an Mayer und Schulz zu verbalisieren und zu verschicken. Interessant ist, daß dieser initiale Vorschlag noch vage ist bezüglich des Zeitpunktes des Treffens und daß (noch) kein Ort vereinbart ist (Nachricht (1)). Schulz empfängt den natürlichsprachlichen Teil der e-mail und antwortet zustimmend in Freitext (Nachricht (2)). Frau Mayer entnimmt ihrem altmodischen (ledergebundenen) Kalender, daß sie am gleichen Nachmittag bereits einen Termin hat, und schlägt 15:30 h als Zeit des Treffens vor (Nachricht (3)).

Schmidts COSMA-SYSTEM empfängt die ankommende e-mail und läßt den natürlichsprachlichen Rumpf durch das DISCO-Kernsystem analysieren. Die Analyse ergibt, daß es sich bei Nachricht (3) entweder um einen Vorschlag für einen neuen Termin handelt oder um eine Verfeinerung *(refinement)* eines Termins, über den bereits verhandelt wurde; die Disambiguierung erfolgt durch den Terminplaner aufgrund des aktuellen Zustandes der Wissensbasis. Da 15:30 h in den ursprünglich von Schmidt vorgeschlagenen Zeitrahmen (viz. den ganzen Nachmittag) fällt, akzeptiert sein COSMA-SYSTEM den Termin und benachrichtigt alle Teilnehmer per e-mail über die erfolgreiche Terminplanung und die endgültigen Daten (Zeit und Ort) des Treffens (Nachricht (4)). Schlußendlich wird Schmidts Terminkalender aktualisiert.

```
Date: Tue, 26 Apr 94 16:17:14 +0200
Message-Id: <4711@cl.dfki.uni-sb.de>
Received: by cl.dfki.uni-sb.de; Tue, 26 Apr 94 16:17:14 +0200
From: schmidt@dfki.uni-sb.de (Peter Schmidt)
To: mayer@development.de, schulz@research.de
Subject: Abschluszbericht

Sehr geehrte Frau Dr. Mayer, sehr geehrter Herr Schulz!

Ich moechte mit Ihnen den folgenden Termin vereinbaren:

  Zeit:         Donnerstag, der 28. April 1994 nachmittags
  Ort:          DFKI
  Teilnehmer:   Dr. Mayer, Herr Schulz, Prof. Dr. Schmidt
  Dauer:        1 Stunde
  Thema:        Abschluszbericht

Mit freundlichen Gruessen,
Peter Schmidt

--- Dieser Text wurde automatisch durch COSMA generiert ---
```

Abbildung 5.4: COSMA Terminvorschlag als e-mail.

5.3.2 Kopplung mit dem Anwendungssystem

Die Koppelung zwischen der DISCO Kernmaschine und der internen Repräsentationssprache (IR) des COSMA-Terminplaners ist eine Erweiterung der Methoden, die bereits erfolgreich in der semantischen Komponente (*NLL*) des DISCO-Systems eingesetzt wurden (siehe 2.5.3).

Basierend auf der Annahme, daß es etliche Gemeinsamkeiten der Zielsetzung und Anwendung zwischen Bedeutungsrepräsentationssprachen und Programmiersprachen gibt, orientieren sich die Techniken der semantischen Verarbeitung in *NLL* wie auch die *NLL*-Schnittstellen stark an den Methoden des traditionelllen Übersetzerbaus (Kompilation). Die Manipulation von NLL-Ausdrücken wird insofern ebenso wie die Übersetzung von Formeln in oder aus anderen Formalismen als Baumtransformation modelliert, die auf der abstrakten .*NLL*-Syntax operiert (Nerbonne/Laubsch/Diagne [113] führen die Parallelen zwischen *NLL* und Techniken des Übersetzerbaus näher aus).

Die im COSMA-Demonstrator realisierten Transformationen lassen sich gemäß ihrer Allgemeinheit (bezüglich der spezifischen Semantiktheorie beziehungsweise der Anwendungsdomäne) und ihrer Zielsprache (*NLL* beziehungsweise IR) in vier Hauptgruppen unterteilen ([134]):

(l) **reine** *NLL* **Inferenz:** theorieunabhängige, rein logische Äquivalenztranformationen, e.g. Auflösung verschachtelter Konjunktionen, Unterdrückung von unbeschränkten Variablen;

(n) **Simplifikation:** bedeutungserhaltende Transformationen, die eine 'kanonische' Form herstellen, e.g. Unifikation mehrerer Beschränkungen der gleichen Variable;

(m) Disambiguierung und domänenspezifische Inferenz: Transformationen, denen spezifisches Wissens über die Anwendungsdomäne zugrundeliegt, e.g. Skopusresolution, Verankerung unterspezifizierter Zeitangaben (teilweise implementiert);

(iv) **Übersetzung:** Abbildung in eine andere abstrakte Syntax; Moderation unterschiedlicher formaler Mächtigkeiten, e.g. Übersetzung in SQL oder COSMA ir.

Während eine Reihe rein logischer Inferenzregeln (Klasse (I)) im Kern des *NLL*-Semantikmodulsenthaltenist,hängtdiezweitederangegebenenKlassenvonderderDISCO-Grammatikzugrundeliege

tionsmethode realisiert. Die Hauptaufgabe der Simplifikationstranformationen ist es, von (überwiegend syntaktisch bestimmten) Oberflächeneigenschaften der initialen Semantik (als Merkmalsstruktur), die durch die DISCO-Grammatik abgeleitet wird, zu abstrahieren, um so eine quasi-normalisierte semantische Repräsentation zu erzeugen.

Analog wurde der Kompilationsansatz erfolgreich für die Schnittstelle zwischen *NLL* und der COSMA IR (siehe oben) adaptiert. Die Übersetzung wird durch mehrere Schichten

von Regelmengen realisiert, die in einer *post-order (bottom-up)* Baumtraversierung angewendet werden.[1]

Der *NLL*-Baumtransformationsansatz erwies sich insbesondere bei der Realisierung des COSMA-Prototypen als geeignet, größere und komplexe Strukturen zu verarbeiten und gleichzeitig die Modulation der unterschiedlichen Ausdrucksmächtigkeit zwischen *NLL* und der COSMA IR flexibel zu unterstützen.

5.3.3 Template--Generierung

Die Generierung von Systemäußerungen erfolgt in COSMA durch eine template-basierte Systemkomponente, die auf die vorliegende Kommunikationssituation abgestimmt ist. Vom Planungssystem werden stets komplette Terminspezifikationen übergeben, die mit einer der Domänenaktionen (arrange(), accept(), modify(), usw.) assoziiert sind. Das Anwendungssystem macht keinen Unterschied zwischen dem, was es "weiß" und dem, was es kommunizieren will. Folglich können diese Eingabestrukturen angemessen durch die Kombination vorgefertigter Textteile bzw. Templates verbalisiert werden. Dabei wird von folgender Textstruktur eines Briefs ausgegangen:

1. Anrede
2. Hauptteil des Briefs
   - Sprechakt
   - Tabelle mit Termindaten
3. Grußformel, Unterschrift
4. COSMA-Disclaimer

Für jeden Adressaten verwendet das System ein Modell, das Namen, e-mail-Adressen, Titel und Geschlecht enthält und angibt, auf welche Weise (formal, informell) er bzw. sie anzureden ist. Hierdurch bestimmen sich Anrede und Grußformeln. Der Sprechakt leitet sich aus den grundlegenden Domänenaktionen ab und wird als Satz formuliert. Die Termindetails werden der Übersichtlichkeit halber in fester Anordnung als Tabelle dargestellt. Dabei werden die kanonisch repräsentierten Zeitdaten aus der Eingabestruktur in Kalender- und Zeitangaben umgerechnet. Ein Beispiel zeigt Abbildung 5.4.

5.3.4 Graphische Benutzerschnittstelle

Auf der Basis des Garnet Toolkit der CMU wurde eine grafische Benutzerschnittstelle konstruiert, um die Vorteile des CoSMA-Systems für den Arbeitsplatz zu veranschaulichen.

[1] Obwohl alle vier Klassen von *NLL*-Transformationen im aktuellen Prototypen noch als handkodierte Regelmengen, die durch einen optimierten Baumtraversierungsalgorithmus verarbeitet werden, realisiert sind, wurde in der Zwischenzeit ein Regelkompiler entwickelt, der aus deklarativen meta-syntaktischen Regelspezifikationen die entsprechenden Transformationsfunktionen generiert.

Die Schnittstelle ermöglicht den direkten Zugriff auf den Inhalt des Terminkalenders und die Eingabe und Löschung von Einträgen. Das System ist in der Lage, selbständig mit anderen Benutzern über Email zu kommunizieren, sofern Änderungen im Terminkalender vorgenommen werden, die auch andere Teilnehmer betreffen. Obwohl diese Fähigkeit während der Laufzeit des Projektes nicht ausgeschöpft wurde, ermöglicht die Auslegung der Schnittstelle dem Benutzer auch die Eingabe von frei wählbaren oder strukturierten Textsequenzen in das Formular.

Abbildung 5.5 zeigt das System in Aktion. Die Benutzeroberfläche ist derzeit in Englisch - obwohl COSMA kein Übersetzungssystem ist, schaut die Grafik-Schnittstelle in der internen semantischen Repräsentation direkt nach und kann daher so gestaltet werden, daß sie alle direkt interpretierten Felder in der Sprache des Benutzers zeigt.

Das Kalenderfenster ganz links zeigt vergangene Zeit dunkel und die zukünftigen Tage hellgrau eingefärbt. Die Ansicht des Kalenders kann mit dem Rollbalken nach oben und unten in Wochenabständen verschoben werden.

Das danebenliegende Terminkalenderfenster zeigt die Zeit als unendliches Band, von dem aus "Terminvordrucke" durch Anklicken von existierenden Einträgen oder Löschen freier Felder geöffnet werden können. Der Status dieser Termine ist durch Grautöne unterschieden: Bestätigte Termine sind weiß, während für den Eintrag im grauen Feld noch die Zustimmung der anderen Teilnehmer eingeholt werden muß. Diese und andere Informationen des Displays werden ständig aktualisiert.

Das Formular für den Terminkalender hat auch einige nur teilweise strukturierte Felder: in sie kann man Daten eingeben mit der Hilfe von Pulldown-Menüs - wie z.B. im unteren Fenster sichtbar -, die rechts von den Symbolen aus geöffnet werden können. Numerische Werte sowie Kalender- und Uhrzeitwerte können in diesen Feldern direkt mit der Maus erhöht oder verringert werden, und erfahrene Benutzer können strukturierte Werte auch direkt eintippen. Ebenso kann ein freier deutscher Text in diese Felder eingegeben werden.

In der Endversion dieser Anwendung würde dies dem natürlichsprachlichen System zur Auswertung zugeleitet, oder es wäre möglich, Fragen oder Anweisungen statt über die Benutzung einer strukturierten Schnittstelle direkt über das Notizfeld einzugeben, etwa um eine Sekretärin zu benachrichtigen.

5.4 Erfahrungen mit COSMA

Mit der Entwicklung des COSMA-Systems haben wir nicht nur gezeigt, daß das existierende NL Kernsystem als Schnittstelle zu existierenden Anwendungssystemen eingesetzt werden kann, sondern darüber hinaus, daß das kommunikative Leistungsspektrum solcher Anwendungssysteme signifikant gesteigert werden kann. Da weiterhin angestrebt wurde, die existierende Funktionalität des Kernsystems kompromißlos einzusetzen, ist die Existenz von COSMA auch eine Bestätigung für den im gesamten Projekt verfolgten Ansatz, moderne computerlinguistische Methoden in relevanter Weise für Anwendungen nutzbringend einzusetzen.

Im besonderen können mit der Entwicklung von COSMA folgende Erfolge verbucht

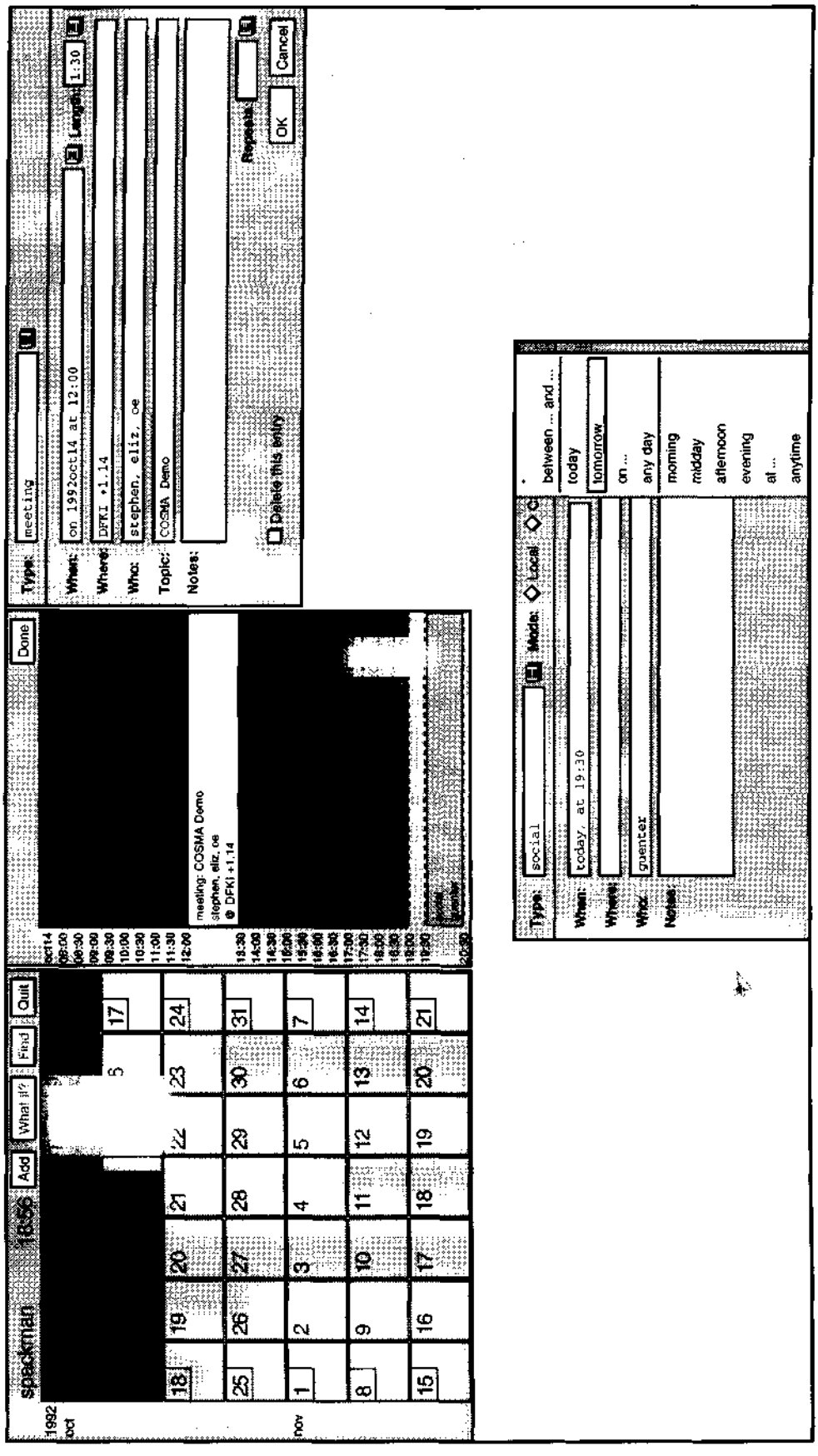


Abbildung 5.5: Fenster der graphischen Schnittstelle

werden:

- Die objektorientierte Architektur zeigte sich flexibel genug, um auch als Basis für die Systemintegration in COSMA eingesetzt zu werden. So konnten über zehn verschiedene neue Module innerhalb kurzer Zeit (ca. eine Woche) integriert werden.
- Die Verwendung der constraint-basierten Grammatik des Kernsystems vereinfachte die Gesamtarchitektur insgesamt, da durch die homogene und kompakte Repräsentation des grammatikalischen Wissens redundante Übersetzungen vermieden werden. Inbesondere der "sign"-basierte Ansatz vereinfachte die Realisierung der Sprechakterkennungskomponente erheblich.
- Die Notwendigkeit einer systematischen Koppelung des Kernsystems mit dem Anwendungssystem stimulierten neue Forschungsaufgaben in Bereichen wie anwendungsorientierter Erwerb und Einsatz von Subsprachen oder konfigurierbarer Generierungssysteme.
- Die Entscheidung, die Auswahl alternativer Sprechakte durch das Anwendungssystem selbst treffen zu lassen, wurde in relevanter Weise bestätigt. Denn nur die Anwendung verfügt über das hierfür notwendige Wissen ihres globalen und aktuellen Zustands.

Neben diesen positiven und auch stimmulierenden Erfahrungen konnte herausgearbeitet werden, welche Erweiterungen und Verbesserungen nötig sind, um das volle Leistungsspektrum der natürlicher Sprache zukünftig für COSMA und andere Anwendungen bereitzustellen.

Insbesondere zeigte sich, daß die auf endlichen Zuständen basierenden Protokolle des Planners unzureichend für flexible und robuste natürlichsprachliche Dialoge sind, was durch die Auswertung von e-mail Korpora in DiTo gezeigt wurde. Zusätzlich basieren die gegenwärtigen Protokolle auf der Annahme einer fehlerfreien Kommunikation, eine Anahme, die in Bezug auf natürlichsprachliche Äußerungen nicht gültig ist.

Um die angestrebte zusätzliche Flexibilität und Robustheit zu erreichen, gehen wir davon aus, daß ein stark kompositionelles Dialogmodell entwickelt werden muß, in dem komplexe Aktionen auf der Basis einer formalen und expliziten Beschreibung von Sprechakten definiert werden, die es auch erlaubt, Reparaturstrategien methodisch zu formulieren. Diese Techniken erfordern aber einen sehr komplexen Informationsaustausch zwischen den Kommunikationspartnerinnen. Wir gehen daher davon aus, daß die gegenwärtige Koppelung zum Anwendungssystem durch anwendungsunabhängiges Reasoning und Wissensrepräsentation im Kernsystem zu erweitern ist.

Kapitel 6

Implementation

Die Software-Entwicklung in DISCO nahm einen breiten Raum ein. Dieses Kapitel geht auf verschiedene Aspekte der Verwaltung, der Integration und der Portierung der im Projekt entwickelten Modulversionen ein.

6.1 Softwareverwaltung

Das Projekt DISCO hatte als großes NL Projekt mit dem Anspruch, auch lauffähige Demonstratoren zu liefern, eine große Anzahl von Teilgruppen, die mit Implementierung beschäftigt waren und eine noch größere Anzahl von Teilmodulen geliefert haben.

Um diese in viele verschiedene Module aufgeteilte Menge an Software mit all den wechselseitigen Abhängigkeiten verwalten zu können, war es nötig, ein Schema zur Verwaltung der Software zu entwickeln das mehreren Anforderungen genügt:

- Zentrale Versionsverwaltung der Module

  Eine zentrale Versionsverwaltung bringt zwar einen erhöhten Verwaltungsaufwand mit sich, hat aber den Vorteil, daß es für die Nutzer eines Moduls immer offizielle Versionen gibt. Diese Maßnahme verhindert weitestgehend die Existenz von lokalen Versionen und darauf aufbauenden Code, der dann nur mit erhöhtem Aufwand zu einem größeren System integriert werden kann. Außerdem kann so (fast) immer eine

•offiziellelauffähigeVersiongarantiertwerden.•LokaleEntwicklungsumgebungen

  Das bedeutet, daß es für jeden Entwickler möglich sein muß, Veränderungen an seinem Code lokal vorzunehmen, ohne daß andere davon betroffen sind. Diese Veränderungen können dann nach einer Testphase veröffentlicht werden.

- Lokale Testmöglichkeiten

  Es muß möglich sein, neue lokale Versionen mehrerer Module zusammen zu testen, bevor sie veröffentlicht werden. Das bedeutet zwar eine Verletzung der Vorgabe, daß

nur öffentliche Versionen benutzt werden dürfen, ist aber in Fällen, in denen mehrere Module verschiedener Entwickler gleichzeitig verändert werden müssen, um das Funktionieren des Gesamtsystems zu gewährleisten, unabdingbar. Trotzdem darf diese Möglichkeit nur sporadisch genutzt werden.

- Mehrere Versionen unter zentraler Kontrolle

  Um nach den Tests durch den Entwickler eine Phase öffentlicher Tests durchführen zu können, müssen mehrere Versionen gleichzeitig in der zentralen Verwaltung gehalten werden können. Falls sich schwerwiegende Fehler zeigen, kann man schnell auf die vorherige (stabile) Version zurückstellen. Außerdem sollten Entwickler, die von sehr vielen Modulen abhängen, nicht gezwungen sein, diese Testphase mitzumachen, da ihnen in Zeiten hoher Entwicklungsaktivität und Versionswechelraten sonst fast nie ein komplettes stabiles System zur Verfügung steht und sie in ihrer Entwicklung stark behindert werden.

Die für DISCO erstellte Softwareverwaltung benutzt das von Mark Kantrowitz implementierte portable defsystem-Paket. Es erlaubt eine dem UNIX make Kommando ähnliche Spezifikation von Modulabhängigkeiten, die für unsere Zwecke ausreichend war. Zusammen mit von uns implementierten Bibliotheken konnte die Bereitstellung von mehreren Versionen eines Moduls bewerkstelligt werden.

Die beiden im Softwareänderungszyklus wesentlichen Versionen *licensed* und *test,* entsprechend einer stabilen und der jeweils neuesten Version, konnten um andere, wie z.B. zur Auslagerung einer bestimmten Gesamtsystemversion für eine Demonstration erweitert werden. Module, die nicht von DISCO selbst implementiert waren, wurden dabei vollständig als "externe" Versionen aus den Codeverzeichnissen von DISCO herausgehalten.

Um dem Verwalter des zentralen Softwarepools die Arbeit zu erleichtern, wurden Werkzeuge implementiert, die die Installation neuer Versionen eines Entwicklers als *test*-Version bzw. die Bewegung eines Moduls vom Zustand *test* in den Zustand *licensed* automatisch bewerkstelligen. Dazu mußten auch kleinere Erweiterungen im defsystem-Paket vorgenommen werden, von denen wir hoffen, sie in der offiziellen Version noch unterbringen zu können. Diese Werkzeuge untersuchen die von defsystem benutzten Systemdateien und übernehmen dann das Kopieren der Quelldateien, das Anlegen von Verzeichnissen, Kompilieren der kopierten Quelldateien, Versionskontrolle (RCS) bei Veränderung der *licensed*-Version, Führen eines Logbuches über Installationen und die Benachrichtigung der Softwareverwalter (immer) und Modulentwickler (bei geglückter Installation) per Email.

Die automatischen Werkzeuge wurden konsequent weiterentwickelt, so daß jetzt auch automatische Extraktion von Gesamtsystemen zur Auslieferung oder zum Einfrieren einer Version vorgenommen werden kann. Obwohl die Entwicklung dieser Tools einiges an Arbeit in Anspruch genommen hat, war sie doch für die effiziente Durchführung der Softwareverwaltung von DISCO unabdingbar.

6.2 Die Entwicklungsumgebung

Die Systemintegration in der Kernmaschine wie auch in COSMA wurde mit der DISCO DEVELOPMENT SHELL (kurz DDS) realisiert (vgl. [128] und [126]).

Anforderungen. Die Verwendung moderner Softwaretechniken ist gerade für die Systemintegration von besonderer Relevanz, um folgende Aspekte zu unterstützen bzw. zu ermöglichen.

- Modularität
- Experimentieren mit Kontrollflußinformation
- Integration neuer Module
- Spezifikation von Subsystemen
- Einfacher Austausch von exisitierenden Modulen mit Modulen ähnlicher Funktionalität

Gerade in forschungsorientierten Bereichen ist es von großer Bedeutung, die oben erwähnte Flexibilität zur Verfügung zu stellen, da i.a. mit einer Systemintegration bereits begonnen werden muß, bevor eine exakte Beschreibung der Module - wenn überhaupt - zur Verfügung steht. Damit ist es aber um so wichtiger, schnell und punktuell auf neue Anforderungen zu reagieren.

Um diese Art von Flexibilität auch von der Architekturseite bereitzustellen, wurde im Projekt DISCO ein *Zweiphasenmodell* für die Systemintegration realisiert:

- In einer ersten Phase wird die Architektur unabhängig von konkreten Komponenten und unabhängig vom konkreten Kontrollfluß spezifiziert und softwaretechnisch umgesetzt. Mögliche Komponenten werden als "black boxes" betrachtet, deren Interaktion in abstrakter Weise spezifiziert wird. Ein auf solche Art definiertes System bezeichen wir mit dem Begriff *Rahmensystem (frame System).*
- In der zweiten Phase wird das Rahmensystem durch Integration realer Komponenten und einer genauen Spezifikation des Kontrollflusses *instanziiert.* Dabei sollte die Integration jedes einzelnen Moduls so lokal wie möglich gestaltet werden, um die Menge von Seiteneffekten zu minimieren.

Zur Realisierung dieses Zweiphasenmodells ist es von Vorteil, die Menge der Systemkomponenten nach ihren unterschiedlichen Aufgaben zu unterteilen. In der DISCO Architektur unterscheiden wir gegenwärtig folgende Arten von Komponenten:

- Werkzeuge (z.B. graphische Werkzeuge, Debugger, Printer, Fehlerbehandlungskomponenten);

- Natürlichsprachliche Komponenten (e.g., Morphologiekomponente, Parser, Generator);
- Kontrollkomponente;

Um einen möglichst hohen Grad an Flexibilität und Robustheit zu erlangen (insbesondere in der Entwicklungsphase eines Systems), ist es die zentrale Aufgabe der Kontrollkomponente, den globalen Informationsfluß und die Interaktion zwischen einzelnen Komponenten zu überwachen. Die gegenwärtig betrachteten und auch realisierten Aufgaben der Kontrollkomponente sind:

- Steuerung des globalen Informationsflusses
- Überwachung der Auswertung von Protokollen
- Überprüfung des Datenflusses zwischen Komponenten und gegebenenfalls Aktivierung von Fehlerbehandlungsroutinen
- Verwaltung des globalen Speichers

Zusätzlich zur Systemplattform ist auch eine Kommandoebene für die direkte Kommunikation mit dem Systembenutzer realisiert. Zweck dieser Kommandoebene ist es, dem Benutzer in einer möglichst Programmiersprachen-unabhängigen Weise Zugang zu den einzelnen Komponenten und Subsystemen zu ermöglichen. Auf die genauen Details dieser Kommandoebene wird jedoch im Weiteren nicht eingegangen.

Objektorienter Systementwurf. Wenn eine neue Komponente in das bestehende System integriert werden muß, ist es erstrebenswert, sich nur auf die Aspekte konzentrieren zu müssen, die sich aus den spezifischen Erfordernissen dieser neuen Komponente ergeben. Algorithmen und Daten, die von allen Komponenten (oder Komponenten eines spezifischen Typs) gemeinsam genutzt werden, sollten demnach nur einmal spezifiziert und automatisch für eine Komponente zur Verfügung gestellt werden, wobei Seiteneffekte auf bereits existierende Komponenten auszuschließen sind.

Zur Realisierung des Zweiphasenmodells haben wir daher einen *objektorientierten Systementwurf* gewählt. Im objektorientierten Programmierparadigma wird ein Programm als aus einer Menge von Objekten bestehend spezifiziert. Der mögliche Zustand von Objekten und die mit ihnen assoziierten Aktionen werden einmal definiert wenn ein Objekt kreiert wird. Die zentralen Bausteine einer objektorientierten Programmiersprache sind Objekte, Klassen und Vererbung. Objekte sind Module, die Daten und Operationen über den Daten einkapseln. Jedes Objekt ist eine Instanz einer Klasse, die ihrerseits die Struktur und das Verhalten ihrer Instanzen definiert. Vererbung erlaubt es, bereits definierte Klassen zu spezialisieren. Das Ergebnis ist eine Vererbungshierarchie, entlang derer Klassen die Struktur und das Verhalten ihrer Superklassen erben. Der Vorteil für die Programmierung liegt darin, daß für neue Klassen nur deren essentieller Unterschied - relativ zur bestehenden Hierarchie - definiert werden muß. Damit wird in direkter Weise eine modulare und auch robuste Systemspezifikation unterstützt, die einfach benutzt und erweitert werden kann.

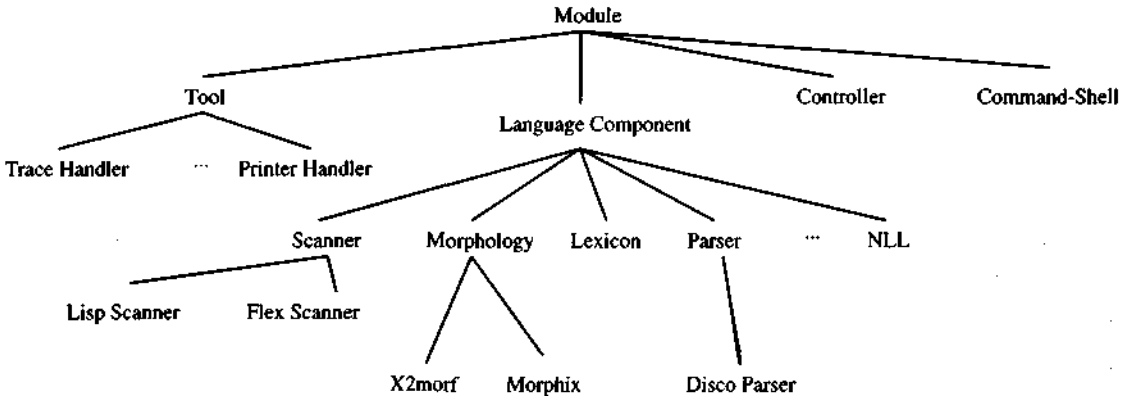


Abbildung 6.1: Ausschnitt aus der gegenwärtigen Objekthierarchie.

CommonLispObjectSystem. DiewesentlicheProgrammiersprachedesDISCO-ProjektesistCommonLisp.D

miersprache mit Namen CLOS (Common Lisp Object System) standardmäßig erweitert worden. Daher war es naheliegend, CLOS als Implementationssprache für das Zweiphasenmodell zu wählen. Ein wesentlicher Vorteil von CLOS ist, daß es auf einfache Weise möglich ist, "normalen" Lispcode mit objektorientiertem Code zu vereinen. CLOS unterstützt auch multiple Vererbung, womit es möglich ist, Aktionen (d.h. Methoden) für bestimmte Kombinationen von Klassen zu definieren. Damit wird ein großer Anteil an der Kontrolle automatisch durch CLOS realisiert.

Dies erlaubt die angestrebte lokale Integration neuer Komponenten, was ihre Integration simplifiziert und beschleunigt, da der Programmierer von lästiger Kleinarbeit befreit wird. Natürlich sollte man sich bewußt sein, daß CLOS Modularität nicht durch sich selbst realisiert oder daß es ausreichend wäre, Programme nur durch Organisation zu definieren; es ist nur ein - wenn auch komplexes - Werkzeug, das hilft, modulare Systeme zu erreichen.

Die Klassenhierarchie von DISCO. Die DOS besteht aus einer Klassenhierarchie und der Spezifikation von Methoden. Jeder Komponententyp und seine Spezialisierungen sind als CLOS Klassen definiert. Abbildung 6.1 zeigt einen Ausschnitt aus der gegenwärtigen Hierarchie.

module ist die generellste Klasse, von der alle anderen Klassen erben. Die Klasse language component subsumiert alle linguistischen Komponenten des Kernsystems. Aktive Module sind Instanzen dieser Komponente.

Neue Module werden in das System integriert, indem ihnen eine Klasse zugewiesen wird. CLOS unterstützt eine dynamische Erweiterung der Klassenhierarchie, so daß neue Module auch während der Laufzeit hinzugefügt werden können. Wenn wir zum Beispiel ein neues Parsermodul integrieren wollen, so können wir entweder die bereits existierende Klasse parser verwenden oder wir können eine neue Subklasse, sagen wir alternative-parser definieren. Im ersten Fall könnten wird direkt alle Methoden, die bereits für die Klasse parser definiert sind benutzen. Im zweiten Fall müssten wir eventuell neue Metho-

den für alternative-parser definieren. Prinzipiell könnte es auch der Fall sein, daß alternative-parser und disco-parser so viele Gemeinsamkeiten haben, daß es angebracht wäre, das parser Teilnetz zu verfeinern, um so mögliche Redundanzen zu vermeiden.

Definition von Protokollen. Protokolle werden definiert, um den Kontrollfluß zwischen Komponenten zu steuern. Protokolle sind Methoden der Klasse Controller. Sie spezifizieren die Menge der Komponenten, die zur Lösung einer bestimmten Anwendung eingesetzt werden sollen, sowie das Ein-/Ausgabeverhalten der beteiligten Komponenten. Alle gegenwärtig definierten Protokolle folgen der Struktur, wie sie in Abbildung 6.2 gezeigt ist.

(call-component Controller component$_1$)
(check-and-transform Controller component$_1$ component$_2$)
(call-component Controller component$_2$)
(check-and-transform Controller component$_2$ component$_3$)
(call-component Controller component$_3$)

(check-and-transform Controller component$_{(n-1)}$ component$_n$)
(call-component Controller component$_n$)

Abbildung 6.2: Schematische Struktur von Protokollen.

Mittels der Methode call-component wird eine individuelle Komponente aktiviert. Zwischen dem Aufruf von Protokollen wird mittels der Methode check-and-transform das Ein-/Ausgabeverhalten definiert, wobei komponentenspezifische Fehlerbehandlungsroutinen aktiviert werden können, falls keine fehlerfreie Abbildung möglich ist. In der gegenwärtigen Version des Systems besteht die Fehlerbehandlung darin, die weitere Verarbeitung zu unterbrechen und den Benutzer auf die Fehlerquellen hinzuweisen. Beispielsweise definiert die Ausgabe der Morphologiekomponente den Eingabestrom für den Parser. Durch Aktivierung der Methode

(check-and-transform Controller morphology parser)

wird überprüft, ob die Ausgabe für den Parser wohlgeformt ist. Konnte zum Beispiel ein Wort morphologisch nicht behandelt werden, so wird eine weitere Verarbeitung unterbrochen und der Benutzer über den aufgetretenen Fehler unterrichtet.

Zur Unterstützung der Grammatikentwicklung können Subsysteme definiert werden, die nur die für diesen Zweck relevanten Komponenten aktivieren. In jedem Fall steht die gleiche Funktionalität und auch die gleiche Menge von Werkzeugen zur Verfügung . Prinzipiell ist es auch möglich, dem Benutzer zu erlauben, eigene Protokolle zu definieren, zum Beispiel um Module eigener Entwicklung zu testen. Dies ist möglich, da die Definition von Protokollen standardisiert vorgenommen wird.

Die Architektur ist nicht auf eine Pipeline-Verarbeitung eingeschränkt, wie dies das obige Beispiel nahelegen könnte. Sie ist prinzipiell auch für kaskadierte oder blackboard-orientierte Architekturen geeignet. Im letzteren Fall könnte entweder der Arbeitsspeicher des Kontrollers verwendet oder eine Klasse black-board definiert werden.

6.3 Portierungen

Bei der Entwicklung des DISCO-Kernsystems wurde stets darauf geachtet, daß der Code den neuesten Lisp-Standards entspricht und auch möglichst portabel ist. DISCO folgt nach Ablauf des Projekts vollständig den Richtlinien des Lisp-Standardisierungskomitees X3J13. Dies ist ein wichtiger Faktor im Hinblick auf eine Verbreitung des Systems; es sichert aber auch die Wartbarkeit des Codes bei einer längerfristigen Nutzung.

Das System wurde im wesentlichen in Franz Allegro Common Lisp entwickelt und ist in den Versionen 4.1 und 4.2 lauffähig. Außerdem wurde es bereits auf verschiedene andere Lispversionen portiert, vor allem auf LUCID Common Lisp (V4.0.X und V4.1.x) aber auch auf die frei erhältliche Lispumgebung CLisp.

Diese Portierungen haben es uns ermöglicht, das System nicht nur dem Projekt PRAC-MA des SFB 314 der Univ. des Saarlandes sondern auch dem CSLI Stanford zur Verfügung zu stellen. Dort wird es im Rahmen des VerbMobil Verbundprojektes genutzt, um eine HPSG-Grammatik des Englischen zu entwickeln.

Da für die am CSLI verwandte Hardwareplattform (Hewlett Packard PA Rechner) nur eine ältere Version von LUCID Common Lisp eingesetzt werden konnte, wurde die Portabilität des Codes im DISCO-System auf eine harte Probe gestellt. Trotzdem war es möglich, das System innerhalb weniger Wochen in voller Funktionalität stabil zum Laufen zu bringen.

Die Portierung auf die erwähnte LUCID Version hat es außerdem möglich gemacht, DISCO auf IBM RS6000 und IBM PowerPC Rechnern zu benutzen und der Forschungs-gruppe der IBM Stuttgart zur Verfügung zu stellen. Neben diesen beiden Hardwareplatt-formen läuft DISCO auf SPARC und HP-PA wie auch auf Intel 80386 und 80486 Rechnern einwandfrei.

Es werden zahlreiche verschiedene Betriebssysteme unterstützt: SunOS und Solaris, HP-UX, IBM AIX und Linux. Da einige Teile des Systems als externe Prozesse realisiert sind (Fegramed, Scanner, E-Mail Interface), war eine mit dem POSIX Standard verträgli-che Schnittstelle unerläßlich. Die externen Prozesse sind dem ANSI C Standard gemäß implementiert und konnten daher so schnell wie das DISCO-Kernsystem selbst auf die ver-schiedenen Plattformen und Betriebssysteme übertragen werden.

Durch die Portierung wurde der DISCO-Code einem wertvollen und harten Test in Bezug auf Portabilität und Robustheit unterzogen, den er nach unserer Meinung mit gutem Erfolg bestanden hat. Außerdem erlaubte es uns den Vergleich verschiedener Lispversionen in Bezug auf Laufzeiteffizienz, Platzverbrauch, Einhaltung des vorgeschlagenen Common Lisp Standard usw.

Die Portierung auf Macintosh Lisp ist begonnen worden und aufgrund der zuvor unternommenen Bemühungen schon so gut wie abgeschlossen.

Bibliographie

[1] Hassan Ait-Kaci and Roger Nasr. Login: A logic programming language with built-in inheritance. *The Journal of Logic Programming*, 3:185-215, 1986.

[2] Jan Alexandersson, Elisabeth Maier, and Norbert Reithinger. A robust and efficient three-layered dialog component for a speech-to-speech translation System, submitted to Conference on Applied Natural Language Processing, Stuttgart, 1994.

[3] J. F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123-154, 1984.

[4] James Allen. Recognizing intentions from natural language utterances. In Michael Brady and Robert C. Berwick, editors, *Computational Models of Discourse.* MIT Press, 1983.

[5] D. E. Appelt. *Planning English Sentences.* Cambridge University Press, Cambridge, 1985.

[6] J. A. Austin. *How to Do Things with Words.* Harvard University Press, 1962.

[7] Franz Baader, Hans-Jürgen Bürckert, Bernhard Nebel, Werner Nutt, and Gert Smolka. On the expressivity of feature logics with negation, functional uncertainty, and sort equations. *Journal of Logic, Language and Information,* 2:1-18, 1993.

[8] Rolf Backofen. Integration von Funktionen, Relationen und Typen beim Sprachentwurf. Teil II: Attributterme und Relationen. Master's thesis, Universität Erlangen-Nürnberg, 1989.

[9] Rolf Backofen. On the decidability of functional uncertainty. In *Proc. of the 31 st ACL*, pages 201-208, Columbus, Ohio, 1993. ACL.

[10] Rolf Backofen. Regulär path expressions in feature logic. In Claude Kirchner, editor, *Proc. of the Rewriting Techniques and Applications (RTA '93),* pages 121-135, Montreal, Canada, 1993. Also available as Research Report RR-93-17, Deutsches Forschungszentrum für Künstliche Intelligenz, GmbH.

[11] Rolf Backofen. Regular path expressions in feature logic. *Journal of Symbolic Computation,* 17:412-455, 1994. Short versions appeared as [9] and [10].

[12] Rolf Backofen, Lutz Euler, and Günther Görz. Towards the integration of functions, relations and types in an AI programming language. In *Proc. GWAI,* 1990. Also available as Research Report RR-91-32, Deutsches Forschungszentrum für Künstliche Intelligenz, GmbH.

[13] Rolf Backofen, Lutz Euler, and Günther Görz. Distributed disjunctions for LIFE. In H. Boley and M. M. Richter, editors, *Proc. of the International Workshop on Processing Declarative Knowledge (PDK'91),* pages 161-170, Berlin, 1991. Springer Verlag.

[14] Rolf Backofen, Hans-Ulrich Krieger, Stephen P. Spackman, and Hans Uszkoreit (eds.). Report of the EAGLES workshop on implemented formalisms at DFKI, Saarbrücken. Technical Report D-93-27, Deutsches Forschungszentrum für Künstliche Intelligenz, Saarbrücken, 1993.

[15] Rolf Backofen and Gert Smolka. A complete and recursive feature theory. In *Proc. of the 31 st ACL,* pages 193-200, Columbus, Ohio, 1993. ACL. Full version available as Research Report RR-92-30, Deutsches Forschungszentrum für Künstliche Intelligenz, GmbH, and to appear in *Theoretical Computer Science.*

[16] Rolf Backofen, Harald Trost, and Hans Uszkoreit. Linking typed feature formalisms and terminological knowledge representation languages in natural language front-ends. In W.Bauer, editor, *Proceedings of the GI Congress, Knowledge-Based Systems 1991,* Berlin, 1991. Springer. Also available as Research Report RR-91-28, Deutsches Forschungszentrum für Künstliche Intelligenz, GmbH.

[17] Rolf Backofen and Christoph Weyers. *UDiNe*—a feature constraint solver with distributed disjunction and classical negation. Technical report, Deutsches Forschungszentrum für Künstliche Intelligenz, GmbH, 1994. Forthcoming.

[18] Afzal Ballim and Yorick Wilks. *Artificial Believers: The Ascription of Belief.* Lawrence Erlbaum Associates, 1991.

[19] Jochen Bedersdorfer, Karsten Konrad, Ingo Neis, Oliver Scherf, Jörg Steffen, and Michael Wein. Eine Spezifikationssprache für Transformationen auf getypten Merkmalsstrukturen. In Harold Boley, François Bry, and Ulrich Geske, editors, *Proceedings of the Workshop on 'Neuere Entwicklungen der Deklarativen KI-Programmierung',* pages 17-30. Research Report RR-93-35 Deutsches Forschungszentrum für Künstliche Intelligenz, GmbH, 1993.

[20] Gabriel Bes, editor. *The Construction of a Natural Language and Graphics Interface. Results and Perspectives from the ACORD Project.* Research Reports ESPRIT. Heidelberg: Springer, 1991.

[2l] H. C. Bunt. Information dialogues as communicative action in relation to partner modelling and information processing. In M.M Taylor, F. Neel, and D. G. Bouwhuis, editors, *The Structure of Multimodal Dialogue.* Elsevier Science Publishers B.V., 1989.

[22] Stephan Busemann. Generalisierte Phrasenstruktur-Grammatiken und ihre Verwendung zur maschinellen Sprachverarbeitung. Research Report RR-90-17, Deutsches Forschungszentrum für Künstliche Intelligenz, GmbH, 1990.

[23] Stephan Busemann. Structure-driven generation from separate semantic representations. In *Proceedings of the 5 th Conference of the European Chapter of the ACL,* pages 113-118, 1991.

[24] Stephan Busemann. Using pattern-action rules for the generation of GPSG structures from MT-oriented semantics. In *Proceedings 12 th International Joint Conference on Artificial Intelligence,* pages 1003-1009, Sydney, 1991. also available as Research Report RR-91-16, Deutsches Forschungszentrum für Künstliche Intelligenz, GmbH, 1991.

[25] Stephan Busemann. *Generierung natürlicher Sprache mit Generalisierten Phrasenstrukturgrammatiken.* Springer, Berlin, Heidelberg, 1992. IFB Bd. 313.

[26] Stephan Busemann. Lexical choice and knowledge representation. In Heinsohn and Hollunder [51], pages 33-39.

[27] Stephan Busemann. A holistic view of lexical choice. In Helmut Horacek, editor, *New Concepts in Natural Language Generation: Planning, Realization, and Systems,* pages 302-308. Frances Pinter, 1993.

[28] Stephan Busemann. Implicit relationships between grammar and control. In Michael Herweg, editor, *Deklarative und prozedurale Aspekte der Sprachverarbeitung. 4.Fachtagung der Sektion Computerlinguistik der Deutschen Gesellschaß für Sprachwissenschaft,* Hamburg, 1993.

[29] Stephan Busemann. Towards Configurable Generation Systems. Some Initial Ideas. In Busemann and Harbusch [31], pages 57-64.

[30] Stephan Busemann. The SeReal System: Putting semantic-head-driven generation to the limits. Technical Document D-94-xx, Deutsches Forschungszentrum für Künstliche Intelligenz,GmbH, 1994. in preparation.

[31] Stephan Busemann and Karin Harbusch, editors. *DFKI Workshop on Natural Language Systems: Re-usability and Modularity. Proceedings,* Technical Document D-93-03, Deutsches Forschungszentrum für Künstliche Intelligenz,GmbH, 1993.

[32] Stephan Busemann and Hans-Joachim Novak. Generierung natürlicher Sprache. In Günther Görz, editor, *Einführung in die Künstliche Intelligenz*, pages 499-558. Addison-Wesley, Bonn, 1993. Also available as Research Report RR-92-50, Deutsches Forschungszentrum für Künstliche Intelligenz, GmbH.

[33] Stephan Busemann, Stephan Oepen, Elizabeth Hinkelman, Günter Neumann, and Hans Uszkoreit. COSMA-multi-participant NL interaction for appointment scheduling. to appear as Research Report RR-94-xx, Deutsches Forschungszentrum für Künstliche Intelligenz, GmbH, 1994. in preparation.

[34] Phillip R. Cohen and Hector J. Levesque. Persistence, intention, and commitment. In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication.* MIT Press, 1990.

[35] Abdel Kader Diagne. DiTo - A Diagnostic Tool for German Syntax. Data Base and User's Manual. Draft-Version, to appear as Technical Document, Deutsches Forschungszentrum für Künstliche Intelligenz, GmbH, in preparation.

[36] Abdel Kader Diagne and John Nerbonne. Flexible semantics communication in integrated speech/language architectures. In Günter Görz, editor, *KONVENS 92*, pages 348-352, Berlin, 1992. Springer.

[37] Ludwig Dickmann, Julia Heine, Judith Klein, Fred Oberhauser, Hannes Pirker, and Julia Simon. Abschlußbericht der Arbeitsgruppe zur Bewertung und Kodierung des SADAW Lexikons. Unpublished Manuscript, Deutsches Forschungszentrum für Künstliche Intelligenz, GmbH. 1993.

[38] F. J. H. Dols and K. van der Sloot. Modelling mutual effects in belief-based interactive Systems. In W. J. Black, G. Sabah, and T. J. Wachtel, editors, *Abduction, Beliefs and Context: Proceedings of the second ESPRIT PLUS Workshop in computational pragmatics,* 1992.

[39] J. Dörre and A. Eisele. Feature logic with disjunctive unification. In *Proc. of the 13th COLING*, pages 100-105, Helsinki, Finland, 1990.

[40] Jochen Dörre and Andreas Eisele. Determining consistency of feature terms with distributed disjunctions. In Dieter Metzinger, editor, *Proc. of the 13th German Workshop on Artificial Intelligence,* volume 216 of *Informatik Fachberichte,* pages 270-279. Springer, Berlin, 1989.

[41] J. Earley. An efficient context-free parsing algorithm. In B. J. Grosz, K. Sparck Jones, and B. L. Webber, editors, *Natural Language Processing*, pages 25-33. Kaufmann, Los Altos, CA, 1986.

[42] Kurt Eberle and Walter Kasper. Tense, aspect, and temporal structure in French. In Kamp [60], pages 4-40.

[43] T. Ellman. Explanation-based learning: A survey of programs and perspectives. *ACM Computing Surveys,* 21(2):163-221, 1989.

[44] R. E. Fikes and N. J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence,* 2:189-208, 1971.

[45] Daniel Flickinger and John Nerbonne. Inheritance and complementation: A case study of *easy* adjectives and related nouns. *Computational Linguistics,* 19(3):269-309, 1992. Also available as Research Report RR-91-30, Deutsches Forschungszentrum für Künstliche Intelligenz, GmbH.

[46] A. Galton. Towards an integrated logic of space, time and motion. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence,* volume 2, pages 1550-1555, 1993.

[47] Jean Mark Gawron, John Nerbonne, and Stanley Peters. The absorption principle and e-type anaphora. In Jon Barwise, Jean Mark Gawron, and Gordon Plotkin, editors, *Proceedings of the Second Conference on Situation Theory and Its Applications,* Stanford University, 1991. CSLI Lecture Notes. Also available as Research Report RR-91-12, Deutsches Forschungszentrum für Künstliche Intelligenz, GmbH.

[48] Jean Mark Gawron, John Nerbonne, and Stanley Peters. Anaphora, quantification and absorption. Technical Report CSLI-91-153, Center for the Studies of Linguistics and Information, 1991.

[49] G. Görz. *Strukturanalyse natürlicher Sprache.* Addison-Wesley, Bonn, 1988.

[50] Hubert Haider and Klaus Netter, editors. *Representation and Derivation in the Theory of Grammar. Studies in NLLT 22.* Reidel, Dordrecht, 1991.

[51] Jochen Heinsohn and Bernhard Hollunder, editors. *Proceedings DFKI Workshop on Taxonomic Reasoning.* Deutsches Forschungszentrum für Künstliche Intelligenz, GmbH, 1992.

[52] Elizabeth Hinkelman. *Linguistic and Pragmatic Constraints on Utterance Interpretation.* PhD thesis, University of Rochester, 1990.

[53] Elizabeth A. Hinkelman. Abuction, Beliefs and Context. In *Proceedings of the Second ESPRIT PLUS Workshop m Computational Pragmatics,* 1992.

[54] Elizabeth A. Hinkelman. Intonation and the reqest/question distiction. In *Proceedings of the International Conference on Spoken Language Processing,* 1992.

[55] Elizabeth A. Hinkelman, Dan Fass, and James Martin, editors. *Special issue on Non-Literal Language,* Computational Intelligence, 1992.

[56] Elizabeth A. Hinkelman and Stephen P. Spackman. Abductive Speech Act Recognition, Corporate Agents and the COSMA System. In W.J. Black, G. Sabah, and T.J. Wachtel, editors, *Abduction, Beliefs and Context: Proceedings of the Second ESPRIT PLUS Workshop in Computational Pragmatics.* Academic Press, 1992. Also available as Research Report RR-93-31, Deutsches Forschungszentrum für Künstliche Intelligenz, GmbH.

[57] Elizabeth A. Hinkelman and Stephen P. Spackman. Communicating with multiple agents. In *Proceedings of the 15th International Conference on Computational Linguistics, COLING-94,* Kyoto, Japan, 1994.

[58] Elizabeth A. Hinkelman, Markus Vonerden, and Christoph Jung. Natural Language Software Registry. Technical Document D-93-10, Deutsches Forschungszentrum für Künstliche Intelligenz, GmbH, 1993.

[59] Ch. G. Jung. P*I* SA - ein STRlPS-basiertes Planmodul in RHET, Abschlußbericht des Fortgeschrittenenpraktikums, unpublished manuscript, Deutsches Forschungszentrum für Künstliche Intelligenz, GmbH,1994.

[60] Hans Kamp, editor. *Tense and Aspect in English and French.* DYANA-Deliverable R2.3.B. Center for Cognitive Science, Edinburgh, 1991.

[61] R. M. Kaplan and J. T. Maxwell III. An algorithm for functional uncertainty. In *Proceedings of the 12th International Conference on Computational Linguistics,* pages 297-302, Budapest, Hungary, 1988.

[62] Ronald M. Kaplan and Joan Bresnan. Lexical-Functional Grammar: A formal System for grammatical representation. In J. Bresnan, editor, *The Mental Representation of Grammatical Relations,* pages 173-381. MIT Press, Cambridge (MA), 1982.

[63] Ronald M. Kaplan and Annie Zaenen. Long-distance dependencies, constituent structure, and functional uncertainty. In M. Baltin and A. Kroch, editors, *Alternative Conceptions of Phrase Structure.* University of Chicago Press, Chicago, 1988.

[64] Walter Kasper. The construction of semantic representations from F-STRUCTURES. In Bes [20].

[65] Walter Kasper. Presuppositions, composition, and simple subjunctives. *Journal of Semantics,* 9:197-221, 1992.

[66] Walter Kasper. Semantische Repräsentation und LFG. Arbeitspapiere des SFB 340 *Sprachtheoretische Grundlagen für die Computerlinguistik* 10, IMS, Stuttgart, 1992.

[67] Walter Kasper. Dynamic interpretation of NLL. Ms. Deutsches Forschungszentrum für Künstliche Intelligenz Saarbrücken, 1993.

[68] Walter Kasper. Integration of syntax and semantics in feature structures. In Busemann and Harbusch [31].

[69] Walter Kasper, Kurt Eberle, and Christian Rohrer. Contextual constraints for MT. In *Proceedings of the Fourth International Conference on Theoretical and Methodological Issues in Machine Translation,* 1992.

[70] Walter Kasper, Marc Moens, and Henk Zeevat. Anaphora resolution. In Bes [20].

[71] Bill Keller. Feature logics, infinitary descriptions and the logical treatment of grammar. Cognitive Science Research Report 205, University of Sussex, School of Cognitive and Computing Sciences, 1991.

[72] Christel Kemke. Modelling neural networks by networks of finite automata. In D.W. Ruck, editor, *Science of Artificial Neural Networks*, pages 116-122, 1992. also in: SPIE's Aerospace Sensing, 1992, Orlando.

[73] Christel Kemke and Habibatou Kone. INCOPA - an incremental connectionist parser. In *Proc. of the World Congress on Neural Networks*, volume 3, pages 41-44, 1993.

[74] Christel Kemke and Christoph Schommer. PAPADEUS - parallel parsing of ambiguous sentences. In *Proc. of the World Congress on Neural Networks*, volume 3, pages 79-82, 1993.

[75] Christel Kemke and A. Wiehert. Hierarchical self-organizing feature maps for speech regognition. In *Proc. of the World Congress on Neural Networks*, volume 3, pages 45-47, 1993.

[76] Bernd Kiefer and Oliver Scherf. Gimme more HQ parsers, the generic parser class of DISCO. Technical Document, Deutsches Forschungszentrum für künstliche Intelligenz, 1994, forthcoming.

[77] Judith Klein and Ludwig Dickmann. DiTo-Datenbank. Dokumentation zu Verbrektion und Koordination. Document D-92-04, Deutsches Forschungszentrum für Künstliche Intelligenz, 1992.

[78] Judith Klein, Ludwig Dickmann, Kader Diagne, John Nerbonne, and Klaus Netter. DiTo - ein Diagnostikwerkzeug für deutsche Syntax. In Günter Görz, editor, *KONVENS 92,* pages 380-384. Springer, 1992.

[79] Judith Klein, Ludwig Dickmann, Kader Diagne, John Nerbonne, and Klaus Netter. A Diagnostic Tool for German Syntax. In Busemann and Harbusch [31].

[80] Judith Klein and Klaus Netter. DiTo-Ein Diagnostik-Werkzeug für die syntaktische Analyse. In A. Raasch, editor, *Angewandte Linguistik 1992. 23. Jahrestagung der Gesellschaft für Angewandte Linguistik,* Saarbrücken, 1992.

[8l] Sarit Kraus and Daniel Lehmann. Knowledge, belief and time. In *Theoretical Computer Science,* volume 58, pages 155-174, 1988.

[82] Brigitte Krenn and Martin Volk. DiTo-Datenbank. Dokumentation zu Funktionsverbgefügen und Relativsätzen. Technical Document D-93-24, Deutsches Forschungszentrum für Künstliche Intelligenz, GmbH, 1993.

[83] Hans-Ulrich Krieger. Derivation without lexical rules. In C.J. Rupp, M.A. Rosner, and R.L. Johnson, editors, *Constraints, Language and Computation,* pages 277-313. Academic Press, 1994. also available as Research Report RR-93-27, Deutsches Forschungszentrum für Künstliche Intelligenz, GmbH. Also published in IDSIA Working Paper No. 5, Lugano, November 1991.

[84] Hans-Ulrich Krieger. Typed feature formalisms as a common basis for linguistic specification. In *Machine Translation and the Lexicon.* Springer, Berlin, 1994. A version of this paper is available as a Deutsches Forschungszentrum für Künstliche Intelligenz Research Report.

[85] Hans-Ulrich Krieger and John Nerbonne. Feature-based inheritance networks for computational lexicons. In Ted Briscoe, Ann Copestake, and Valeria de Paiva, editors, *Default Inheritance Within Unification-Based Approaches to the Lexicon.* Cambridge University Press, Cambridge, 1992. also available as Research Report RR-91-31, Deutsches Forschungszentrum für Künstliche Intelligenz, GmbH. Also published in Proceedings of the ACQUILEX Workshop on Default Inheritance in the Lexicon, Technical Report No. 238, University of Cambridge, Computer Laboratory, October 1991.

[86] Hans-Ulrich Krieger, John Nerbonne, and Hannes Pirker. Feature-based allomorphy. In *31st Annual Meeting of the Association for Computational Linguistics,* Ohio State University, 6 1993. Association for Computational Linguistics. Also available as Research Report RR-93-28, Deutsches Forschungszentrum für Künstliche Intelligenz, GmbH.

[87] Hans-Ulrich Krieger and Ulrich Schäfer. *TDL* - a type description language for unification-based grammars. In *Proceedings of the Workshop on 'Neuere Entwicklungen der Deklarativen KI-Programmierung',* Berlin, 1993. Report RR-93-35, Deutsches Forschungszentrum für Künstliche Intelligenz, GmbH.

[88] Hans-Ulrich Krieger and Ulrich Schäfer. *TDL* - ExtraLight User's Guide. Document D-93-09, Deutsches Forschungszentrum für Künstliche Intelligenz, GmbH, 1993.

[89] Hans-Ulrich Krieger and Ulrich Schäfer. *TDL*—a type description language for constraint-based grammars. In *Proceedings of the 15th International Conference on Computational Linguistics, COLING-94,* Kyoto, Japan, 1994.

[103] John Nerbonne. NLL models. unpublished mansucript, Deutsches Forschungszentrum für Künstliche Intelligenz Saarbrücken, 1992.

[104] John Nerbonne. Constraint-based semantics. In Paul Dekker and Martin Stokhof, editors, *Proceedings of the 8th Amsterdam Colloquium*, pages 425-444. Institute for Logic, Language and Computation, 1992. Also available as Research Report RR-92-18, Deutsches Forschungszentrum für Künstliche Intelligenz, GmbH.

[105] John Nerbonne. Evaluierungsworkshop. *Künstliche Intelligenz*, 92(3), September 1992.

[106] John Nerbonne. Feature-based lexicons—an example and a comparison to DATR. In Dorothee Reimann, editor, *Beiträge des ASL-Lexikon-Workshops, Wandlitz (bei Berlin)*, pages 36-49, 1992. Also available as Research Report RR-92-04, Deutsches Forschungszentrum für Künstliche Intelligenz, GmbH.

[107] John Nerbonne. A feature-based syntax/semantics interface. In Alexis Manaster-Ramer and Wlodek Zadrozny, editors, *Proceedings of the Second International Conference on the Mathematics of Language*, Annals of Mathematics and Artificial Intelligence, 1992. Also available as Research Report RR-92-42, Deutsches Forschungszentrum für Künstliche Intelligenz, GmbH.

[108] John Nerbonne. Natural language disambiguation and taxonomic reasoning. In Heinsohn and Hollunder [51], pages 40-47.

[109] John Nerbonne. Nominal comparatives and generalized quantifiers. In Jürgen Allgayer, editor, *Processing Plurals and Quantifiers*, Stanford, 1992. CSLI.

[110] John Nerbonne. Representing grammar, meaning and knowledge. In Susanne Preuß and Birte Schmitz, editors, *Proceedings of the Berlin Workshop on Text Representation and Domain Modeling: Ideas from Linguistics and AI*, pages 130-147, Technische Universität Berlin, 1992. KIT FAST 97. Also available as Research Report RR-92-20, Deutsches Forschungszentrum für Künstliche Intelligenz.

[111] John Nerbonne, Masayo Iida, and William Ladusaw. Semantics of common noun phrase anaphora. In Aaron Halpern, editor, *Proceedings of the 9th West Coast Conference on Formal Linguistics*, pages 379-394, Stanford, 1990. CSLI.

[112] John Nerbonne, Joachim Laubsch, Abdel Kader Diagne, and Stephan Oepen. Natural Language Semantics and Compiler Technology. Research Report RR-92-55, Deutsches Forschungszentrum für Künstliche Intelligenz, GmbH, 1992.

[113] John Nerbonne, Joachim Laubsch, Abdel Kader Diagne, and Stephan Oepen. Software for applied semantics. In Chu-Ren Huang, Claire Hsun hui Chang, Keh jiann Chen, and Cheng-Hui Liu, editors, *Proceedings of Pacific Asia Conference on Formal and Computational Linguistics*, pages 35-56, Taipei, 1993. Academica Sinica. Also

[90] Hans-Ulrich Krieger and Ulrich Schäfer. *TDL*—a type description language for HPSG. Part 1: overview. Technical document, Deutsches Forschungszentrum für Künstliche Intelligenz, GmbH, 1994. Forthcoming.

[91] Hans-Ulrich Krieger and Ulrich Schäfer. *TDL*—a type description language for HPSG. Part 2: user guide. Technical document, Deutsches Forschungszentrum für Künstliche Intelligenz, GmbH, 1994. Forthcoming.

[92] Joachim Laubsch and John Nerbonne. An overview of *NLL*. Technical report, Hewlett-Packard Laboratories, Palo Alto, July 1991.

[93] D. Lewis. *Scorekeeping in a Language Game,* volume l, chapter 13. Oxford University Press, Oxford, 1983. Philosophical Papers.

[94] Andreas Lux, Frank Bomarius, and Donald Steiner. A Model for Supporting Human Computer Cooperation. In *AAAI Workshop on Cooperation among Heterogeneous Intelligent Systems,* San Jose, Ca., 1992.

[95] Miroslav Martinovic and Tomek Strzalkowski. Comparing two grammar-based generation-algorithms: A case study. pages 81-88, Newark, Delaware, 1992.

[96] John Maxwell and Roland Kaplan. An overview over disjunctive constraint satisfaction. In *Proceedings of the International Parsing Workshop 1989,* pages 18-27, 1989.

[97] J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence,* 4, 1969.

[98] Ralph Meyer. Towards a conceptual model of belief and intention in dialogue situations. In W. J. Black, G. Sabah, and T. J. Wachtel, editors, *Abduction, Beliefs and Context: Proceedings of the second ESPRIT PLUS Workshop in computational pragmatics,* 1992.

[99] Bradford W. Miller. The rhet reference manual. technical report 326, University of Rochester, Rochester, USA, November 1990.

[100] S. Minton, J. G. Carboneil, C. A. Knoblock, D. R.Kuokka, O. Etzioni, and Y.Gil. Explanation-based learning: A problem solving perspective. *Artificial Intelligence,* 40:63-118, 1989.

[101] J.D. Moore and C.L. Paris. Planning text for advisory dialogues. In *Proc. Conf. of the 27th Annual Meeting of the ACL,* pages 203-211, Vancouver, 1989.

[102] John Nerbonne. Feature-based disambiguation. In Rod Johnson, Mike Rosner, and C.J.Rupp, editors, *Constraint Propagation, Linguistic Description and Computation,* 1991.

[125] Günter Neumann. Reversibility and modularity in natural language generation. In *Proceedings of the ACL - Workshop Reversible Grammar in Natural Language processing,* pages 31-39, Berkeley, 1991.

[126] Günter Neumann. Design principles of the DISCO System. In *Proceedings of the Twente Workshop on Language Technology (TWLT5),* University of Twente, 1993.

[127] Günter Neumann. Grammatikformalismen in der Generierung und ihre Verarbeitung. *Künstliche Intelligenz.* Themenheft: Generierung 93(2), 1993.

[128] Günter Neumann. The DlSCO Development Shell and its Application in the COSMA System. In Busemann and Harbusch [31], pages 65-74.

[129] Günter Neumann. Application of explanation-based learning for efficient processing of constraint-based grammars. In *Proceedings of the Tenth IEEE Conference on Artificial Intelligence for Application,* Marriott Riverwalk, San Antonio, Texas, March 1994.

[130] Günter Neumann and Wolfgang Finkler. A head-driven approach to incremental and parallel generation of syntactic structures. In *Proceedings of the 13 th International Conference on Computational Linguistics (COLING'90),* pages 288-293, Helsinki, 1990. Also available as Research Report RR-91-07, Deutsches Forschungszentrum für Künstliche Intelligenz, GmbH.

[131] Günter Neumann and Gertjan van Noord. Self-monitoring with reversible grammars. In *Proceedings of the 14 th International Conference on Computational Linguistics (COLING'92),* Nantes, 1992.

[132] Günter Neumann and Gertjan van Noord. Reversibility and self-monitoring in natural language generation. In Tomek Strzalkowski, editor, *Reversible Grammar in Natural Language Processing,* pages 59-96. Kluwer Academic Publishers, 1994.

[133] N. J. Nilsson. *Principles of Artificial Intelligence.* Springer Verlag, 1982.

[134] Stephan Oepen. Compiling *NLL* into the COSMA Internal Representation, 1993. Unpublished Manuscript.

[135] Stephan Oepen. German nominal syntax in HPSG. On syntactic categories and syntagmatic relations. Master's thesis, Freie Universität Berlin, Fachbereich Germanistik, 1993.

[136] Stephan Oepen and Pierre Sablayrolles. COSMA internal representation language. Technical notes. Unpublished Manuscript, Deutsches Forschungszentrum für Künstliche Intelligenz, GmbH, 1992.

[137] C. R. Perrault. An application of default logic to speech act theory. In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication.* MIT Press, 1990.

[138] C. Raymond Perrault and James F. Allen. A plan-based analysis of indirect speech acts. *American Journal of Computational Linguistics,* 6:167-182, July-December 1980.

[139] Carl J. Pollard and Ivan A. Sag. *Information-Based Syntax and Semantics. Vol. 1: Fundamentals.* Univ. of Chicago Press, Chicago, 1987. Center for the Studies of Language and Information, CSLI Lecture Note No. 13.

[140] Carl J. Pollard and Ivan A. Sag. *Head Driven Phrase Structure Grammar.* Studies in Contemporary Linguistics. CSLI, Stanford & University of Chicago Press, 1994.

[141] D. A. Randall, Z. Cui, and A. G. Cohn. An interval logic for space based on "connection". In *Proceedings of the Tenth European Conference on Artificial Intelligence,* pages 394-398, August 1992.

[142] M. Rayner. Applying explanation-based generalization to natural language processing. In *Proceedings of the International Conference on Fifth Generation Computer Systems,* Tokyo, 1988.

[143] Pierre Sablayrolles and Achim Schupeta. Conflict Resolving Negotiation for COoperative Schedule Management Agents (COSMA). Technical Memo TM-93-02, Deutsches Forschungszentrum für Künstliche Intelligenz, GmbH, 1993.

[144] C. Samuelsson and M. Rayner. Quantitative evaluation of explanation-based learning as an optimization tool for a large-scale natural language System. In *IJCAI-91,* pages 609-615, Sydney, Australia, 1991.

[145] Ulrich Schäfer and Hans-Ulrich Krieger. *TDL extra-light* user's guide: Franz Allegro Common LISP version. Document D-93-09, Deutsches Forschungszentrum für Künstliche Intelligenz, GmbH, 1992.

[146] Achim W. Schupeta. Cosma: Ein verteilter terminplaner als fallstudie der verteilten ki. In Juergen Mueller and Donald Steiner, editors, *Kooperierende Agenten,* Saarbrücken, Germany, 1993.

[147] John Searle. *Speech Acts.* Cambridge University Press, New York, 1969.

[148] Stuart M. Shieber, Gertjan van Noord, Fernando C. N. Pereira, and Robert C. Moore. A semantic-head-driven generation algorithm for unification-based formalisms. *Computational Linguistics,* 16(l):30-42, 1990.

[149] Gert Smolka. Feature constraint logics for unification grammars. *Journal of Logic Programming,* 12:51-87, 1992.

[150] P. Tadepalli. A formalization of explanation-based macro-operator learning. In *IJCAI-91,* pages 616-622, Sydney, Australia, 1991.

[151] David R. Traum and Elizabeth A. Hinkelman. Conversation acts in task-oriented spoken dialogue. *Computational Intelligence,* 8(3):575-599, 1992. Special Issue on Non-literal language. Also available as Research Report RR-93-32, Deutsches Forschungszentrum für Künstliche Intelligenz, GmbH.

[152] Harald Trost. The application of two-level morphology to non-concatenative German morphology. In *Proceedings of the 13 th International Conference on Computational Linguistics,* pages 371-376, 1990. Also available as Research Report RR-90-15, Deutsches Forschungszentrum für Künstliche Intelligenz, GmbH.

[153] Harald Trost. A morphological component for the recognition and generation of word forms in natural language understanding Systems: Integrating two-level morphology and feature unification. *Applied Artificial Intelligence,* 4(4):411 - 457, 1991.

[154] Harald Trost. X2MORF: A morphological component based on augmented two-level morphology. In *Proceedings of the 12 th International Joint Conference on Artificial Intelligence (IJCAI '91),* 1991. Also available as Research Report RR-91-04, Deutsches Forschungszentrum für Künstliche Intelligenz, GmbH.

[155] Hans Uszkoreit. Adding control information to declarative grammars. In *Proceedings of the 29 th Annual Meeting of the Association of Computational Linguistics in Berkeley,* pages 237-245, 1991. Also available as Research Report RR-91-29, Deutsches Forschungszentrum für Künstliche Intelligenz, GmbH.

[156] Hans Uszkoreit, Rolf Backofen, Stephan Busemann, Abdel Kader Diagne, Elizabeth A. Hinkelman, Walter Kasper, Bernd Kiefer, Hans-Ulrich Krieger, Klaus Netter, Günter Neumann, Stephan Oepen, and Stephen P. Spackman. DISCO-An HPSG-based NLP System and its Application for Appointment Scheduling. In *Proceedings of the 15th International Conference on Computational Linguistics, COLING-94,* Kyoto, Japan, 1994.

[157] Terry Winograd and Fernando Flores. *Understanding Computers and Cognition.* Ablex, Norwood, NJ, 1986.