Integration von CDG-Parsing und Fehlersimulation

Diplomarbeit

Universität Hamburg

Fachbereich Informatik

Arbeitsbereich Natürlichsprachliche Systeme

30. März 2000

Niels Stockfleth
Eilbeker Weg 165
22089 Hamburg

Erstbetreuer: Prof. Dr.-Ing. Wolfgang Menzel
Zweitbetreuer: Prof. Dr. Christopher Habel

Inhaltsverzeichnis

1 Einleitung **1**
1.1 Motivation . . . 1
1.2 Zielsetzung . . . 4
1.3 Aufbau der Arbeit . . . 4

2 Diagnosetechniken in Fremdsprachenlernsystemen **7**
2.1 Wissensbasierte Lernsysteme . . . 7
2.1.1 Architektur von wissensbasierten Lernsystemen . . . 8
2.1.2 Szenario eines Fremdsprachenlernsystems . . . 9
2.2 Parsing natürlicher Sprache . . . 10
2.2.1 Extra- und Ungrammatikalität . . . 11
2.2.2 Extragrammatikalische Phänomene . . . 13
2.2.3 Unifikationsbasierte Phrasenstrukturgrammatiken . . . 15
2.3 Diagnosetechniken . . . 16
2.3.1 Fehlerregeln . . . 19
2.3.2 Modifikation der Regeln . . . 20
2.3.3 Bedingungsrücknahme . . . 21
2.3.4 Modifikation des Inferenzmechanismus . . . 22

3 Fehlersimulation **23**
3.1 Grundlagen der modellbasierten Diagnose . . . 23
3.1.1 Systembeschreibung . . . 24
3.1.2 Diagnose . . . 25
3.2 Modellbildung . . . 26
3.2.1 Lexikalisches Wissen . . . 28
3.2.2 Morpho-syntaktisches Wissen . . . 29
3.3 Fehlerdiagnose . . . 33
3.3.1 Systemkonstruktion . . . 34
3.3.2 Beispiel . . . 36
3.3.3 Wertaktualisierungsoperator . . . 37
3.3.4 Fehlerfallsemantik . . . 38
3.3.5 Faktenfehler . . . 40
3.3.6 Eine realistische Übung . . . 42
3.3.7 Fehlererklärung . . . 45

4 Fehlerdiagnose mit CDG-Parsing **49**
4.1 Der Formalismus der CD-Grammatik 49
4.1.1 Dependenzrelation und Dependenzstruktur 50
4.1.2 Parsing als Disambiguierungsprozeß 50
4.1.3 Formale Definition einer CD-Grammatik 52
4.1.4 Strukturbeschreibungen 55
4.1.5 Parsing von CD-Grammatiken 57
4.2 Bewertete CD-Grammatiken 60
4.2.1 Bestrafung von Regelverletzungen 60
4.2.2 Kosten einer Strukturbeschreibung 62
4.2.3 Parsing als Optimierungsproblem 63
4.3 Fehlerdiagnose 64
4.3.1 Definition des Diagnosebegriffs 64
4.3.2 Modellierung 65
4.3.3 Fehlererklärung 67

5 Integration von CDG-Parsing und Fehlersimulation **71**
5.1 Motivation 71
5.1.1 Modellierungstechniken zur Suchraumreduktion 74
5.1.2 Transitivität und Merkmalstransport 75
5.1.3 Beschränkung auf binäre Regeln 77
5.1.4 Potential des integrativen Ansatzes 78
5.2 Verarbeitungsmodell 79
5.3 Strukturabbildungen 82
5.3.1 Konstruktion eines Fehlersimulationsmodells 83
5.3.2 Modellierung linguistischer Phänomene 85
5.4 Evaluation 88
5.5 Ein Entwicklungswerkzeug 93

6 Fazit **95**
6.1 Ergebnisse 95
6.2 Probleme 96
6.3 Ausblick 98

A Repräsentation sprachlichen Wissens **101**
A.1 Formale Syntax der Wissensbeschreibung 101
A.2 Lexikon 103
A.2.1 Relevante Merkmale der CD-Grammatik 103
A.2.2 Relevante Merkmale der Fehlersimulation 104
A.3 CD-Grammatik 106
A.3.1 Hierarchien 106
A.3.2 Analyseebenen 107
A.3.3 Regeln 108
A.4 Strukturabbildungen 118
A.4.1 Formalismus 118
A.4.2 Abbildungsregeln 118
A.5 Testsätze 122

A.5.1 Korrekte Sätze 122
A.5.2 Fehlerhafte Sätze 124

B Java-Schnittstelle des integrativen Verfahrens 129

Kapitel 1

Einleitung

Die vorliegende Arbeit befaßt sich mit maschinellen Verfahren für die Verarbeitung *fehlerbehafteter* natürlichsprachlicher Äußerungen. Ein Fehler kann als eine Nichtbeachtung von Regularitäten in einem Modell sprachlichen Wissens angesehen werden. Insbesondere werden Verfahren betrachtet, die eine *Identifikation* von Fehlern ermöglichen. Die Verarbeitungsleistungen des Menschen können für solche Verfahren als Vorbild dienen: Nicht nur, daß der Mensch sogar stark gestörte Äußerungen verstehen kann, er ist zudem in der Lage, fehlerhafte Sätze von korrekten zu unterscheiden und kann in der Regel sofort einen korrigierten Satz zur Verfügung stellen. Besitzt er noch hinreichende Kenntnisse des zugrundeliegenden grammatischen Systems, kann er auch die verletzten Regeln identifizieren und eine angemessene Fehlererklärung geben. Diese spezifischen Eigenschaften für eine maschinelle Verarbeitung, z.B. im Rahmen eines Fremdsprachenlernsystems, zur Verfügung zu stellen, ist eine schwierige Aufgabe. Schon die maschinelle Analyse von *wohlgeformten* natürlichsprachlichen Äußerungen ist aufgrund der inhärenten Mehrdeutigkeiten natürlicher Sprache nicht einfach.

1.1 Motivation

Viele Verfahren für die Diagnose von Fehlern in freien Äußerungen basieren auf der Antizipation des Fehlerfalls durch die Verwendung von Fehlerregeln ([Schneider 1998], [Schwind 1995]). Die Grammatikformalismen sind *konstruktiver* Natur, d.h. aus der vorliegenden Wissensrepräsentation wird versucht, durch einzelne Konstruktionsschritte die fehlerhafte Äußerung aus elementaren Bauteilen zu generieren. Die Vergrößerung der Regelmenge bedeutet, daß es mehr Ableitungen geben kann, wodurch die Verfahren in Effizienzprobleme geraten. Je mehr ungrammatische Phänomene betrachtet werden, desto schwieriger wird die Analyse und desto aufwendiger wird die Modellierung.

Ein Verzicht auf Fehlerregeln in konstruktiven Ansätzen konnte bisher nur für stark eingeschränkte Fehlerklassen unter erheblichen Effizienzproblemen verwirklicht werden ([Mellish 1989]).

Eine andere Sichtweise ergibt sich, wenn die Analyse als *Konsistenzproblem* betrachtet wird, in dem das (korrekte) Wissen über eine Sprache durch eine Menge von konjunktiv verknüpften Bedingungen (*constraints*) repräsentiert wird. In [Mitrovic 1999] wird zwischen einer *generativen* und einer *zustandsbewertenden* Wissensrepräsentation unterschieden:

> „*The function of generative knowledge (e.g. a rule set) is to produce actions vis-a-vis the current problem and the function of evaluative knowledge (a set of constraints) is to evaluate action outcomes as desirable or undesirable.*“ [Mitrovic 1999, S.239]

Gesucht ist diejenige Analyse, welche mit der Bedingungsmenge am verträglichsten ist. Die Verträglichkeit kann z.B. dann als maximal angesehen werden, wenn die wenigsten Bedingungen verletzt werden. Eine wohlgeformte Äußerung genügt den Regeln der Grammatik, wobei die Regeln als einschränkende Bedingungen interpretiert werden. Die Diagnosetechnik, für fehlerbehaftete Äußerungen Regeln der Grammatik zu ignorieren, wird als Bedingungsrücknahme (*constraint relaxation*) bezeichnet. Häufig wird diese Technik in unifikationsbasierten Formalismen verwendet, wobei einzelne Unifikationsgleichungen zurückgenommen werden können ([Schwind 1995],[Uszkoreit 1991],[Hagen 1994]). Die Ermittlung einer Strukturbeschreibung für fehlerbehaftete Äußerungen (*robustes Parsing*) durch Bedingungsrücknahme ist allerdings meist nicht möglich, d.h. strukturelle Fehler werden häufig durch die Verwendung von Fehlerregeln identifiziert.

CDG-Parsing

Ein Grammatikformalismus, der auf einer Menge von konjunktiv verknüpften Bedingungen basiert, ist der CDG-Formalismus (*Constraint Dependency Grammar*, [Maruyama 1990]). Eine Erweiterung des Formalismus wurde erstmals in [Menzel 1998e] für die Diagnose von Fehlern in freien Äußerungen verwendet. Das zugrundeliegende Analyseverfahren wird im folgenden als *CDG-Parsing* bezeichnet. Das Verfahren basiert auf der Repräsentation aller *möglichen* Strukturbeschreibungen, d.h. aller in Frage kommenden Beziehungen zwischen den Elementen in der Äußerung. Eine Äußerung wird analysiert, indem in Abhängigkeit der verletzten Bedingungen bestimmte Strukturbeschreibungen bevorzugt werden. Dies wird modelliert, indem jede Regel mit einer Strafe versehen wird. Werden in einer Strukturbeschreibung Regeln der Grammatik verletzt, entstehen dadurch *Kosten*, die sich aus den Strafen der verletzten Regeln ergeben. Die Analyse einer Äußerung wird damit zum *Optimierungsproblem*: Gesucht ist diejenige Strukturbeschreibung, die die *geringsten* Kosten verursacht. Meist werden die Bedingungen so formuliert, daß nur eine einzige Strukturbeschreibung als Ergebnis geliefert wird. Die verletzten Bedingungen in der optimalen Struktur können für eine Fehlererklärung herangezogen werden.

Obwohl das CDG-Parsingverfahren erfolgreich in Lernsystemen verwendet werden kann, sind doch einige Verbesserungen möglich. Da das Ziel in einer vollständigen Disambiguierung liegt, kann nicht erwartet werden, alternative Fehlererklärungen, z.B. aufgrund anderer Kombinationen von Regelverletzungen oder aufgrund der Berücksichtigung verschiedener Lesarten, zu erhalten.

Ebensowenig ist es möglich, eine Hypothese über fehlerhaftes lexikalisches Wissen des Schülers anzunehmen (*Faktenfehler*). Weiterhin ist die Generierung eines Korrekturvorschlages ohne zusätzliche Erweiterungen nicht möglich.

Ein grundsätzliches Modellierungsproblem ergibt sich aus der passiven Natur der Wissensrepräsentation: Da nur der Wahrheitswert einer Bedingung bestimmt werden kann, ist es nicht möglich, die einer Wortform zugeordnete Information zu *modifizieren*. Die Eigenschaft der Modifikation wird in der Regel benötigt, um in einer Struktur weit entfernte, abhängige Wortformen miteinander in Beziehung setzen zu können. Der Mechanismus, Informationen an andere Elemente zu übertragen, wird als Merkmalstransport bezeichnet. Auf einen Transportmechanismus kann verzichtet werden, wenn die Bedingungen einen ausreichenden *Anwendungsbereich* haben, d.h. ausreichend viele Elemente der Äußerung miteinander in Beziehung setzen können. Das CDG-Parsing gestattet aus Effizienzgründen aber nur die gleichzeitige Betrachtung von maximal vier Wortformen. Für die Überprüfung von transitiven Bedingungsketten, z.B. die Kasuskongruenz in Präpositionalgruppen, wird in der Regel ebenfalls ein Merkmalstransportmechanismus verwendet, um einerseits die kombinatorischen Möglichkeiten von Beziehungen zu reduzieren und andererseits eine linguistisch adäquate Modellierung verwenden zu können. Das Fehlen des Transportmechanismus im CDG-Parsingverfahren erschwert die Modellierung von transitiven Bedingungsketten. Eine derartige Modellierung leidet im allgemeinen unter Effizienzproblemen.

Fehlersimulation

Das Verfahren der *Fehlersimulation*, welches erstmalig in [Menzel 1988] vorgestellt wurde, erlaubt dagegen eine präzise Berechnung von verschiedenen Fehlererklärungen. Das Verfahren kann nicht für die Analyse von freien Äußerungen verwendet werden, da die strukturellen Beziehungen zwischen den Wortformen als bekannt vorausgesetzt werden müssen. Daher wurde es bisher nur für Lückentextübungen eingesetzt ([Stockfleth 1999]).

Es handelt sich um ein *modellbasiertes* Verfahren. Das Wissen wird als System ([Reiter 1987]), bestehend aus einer Menge von Komponenten, den Verbindungen zwischen ihnen und einer Verhaltensbeschreibung der Komponententypen repräsentiert. Die Regeln der Wissensbeschreibung sind elementare Bedingungen, die sich für eine Darstellung als Komponente mit einer vordefinierten Verhaltensbeschreibung eignen. Insbesondere sind nur einfache, morphosyntaktische Wohlgeformtheitsbedingungen modellierbar. Eine Systembeobachtung, welche durch die Eigenschaften der Wortformen in der Äußerung gegeben ist, bestimmt die Werte an den Ein- und Ausgängen der Komponenten. Eine *Hypothese* ist eine Annahme, welche Komponenten sich korrekt und welche sich fehlerhaft verhalten. Die Aufgabe des Diagnoseverfahrens besteht darin, diejenigen (minimalen) Hypothesen zu ermitteln, die aufgrund der strukturellen Information (Komponentennetz) und den Verhaltensbeschreibungen mit der Systembeobachtung logisch konsistent sind. Bei der vorliegenden Implementation handelt sich um ein Simulationsverfahren. Die Konsistenz wird überprüft, indem die durch die Komponentensemantik bedingten Werteänderungen solange durch das Komponentennetz propagiert werden, bis entweder eine Unverträglichkeit festgestellt oder ein Gleichgewichtszustand erreicht wird.

Das Verfahren garantiert einen hohen Grad an Korrektheit und Vollständigkeit. In praktischen Anwendungen wird für die Analyse nur wenig Zeit benötigt. Es gestattet eine exakte Identifikation und Lokalisierung von Fehlern, ermöglicht alternative Sichtweisen (Faktenfehler und Regelfehler), liefert Diagnosen für verschiedene Lesarten und benötigt keine Antizipation des Fehlerfalls. Außerdem läßt sich aus den Faktenfehlern leicht ein Korrekturvorschlag generieren. Insbesondere können Fehler in transitiven Bedingungsketten effizient berechnet werden.

Daraus ergibt sich die Grundidee, durch eine Kopplung beider Verfahren die Fehlerdiagnose zu verbessern. Das CDG-Parsing wird für eine robuste Analyse verwendet und ist zuständig für die Diagnose von strukturellen Fehlern. Die ermittelte optimale Strukturbeschreibung wird verwendet, um ein System für das Verfahren der Fehlersimulation zu konstruieren. Anschließend ermöglicht die Analyse der Fehlersimulation eine detaillierte Diagnose.

1.2 Zielsetzung

Das Ziel dieser Arbeit besteht darin, eine geeignete Kopplung des CDG-Parsings mit der Fehlersimulation zu realisieren, so daß die Vorzüge der jeweiligen Verfahren möglichst wirkungsvoll kombiniert werden können. Schon in [Menzel 1988] wurde vorgeschlagen, die für die Fehlersimulation notwendige Strukturanalyse von einem fehlersensitiven Parser generieren zu lassen. Der Vorschlag, das CDG-Parsingverfahren zu verwenden, findet sich in [Menzel 1998c].

Für eine Kopplung beider Verfahren ist es notwendig, einen Formalismus zu entwickeln, der aus den Strukturbeschreibungen des CDG-Parsings eine für das Verfahren der Fehlersimulation verwendbare Darstellung erzeugen kann. Zudem ist eine Implementation des erweiterten, fehlersensitiven Parsingverfahrens, im folgenden auch als integratives Verfahren bezeichnet, zur Verfügung zu stellen. Es kann dabei auf eine Implementation des CDG-Parsings und der Fehlersimulation zurückgegriffen werden, wobei insbesondere eine Modifikation der Fehlersimulation notwendig ist.

Für die Evaluation des integrativen Verfahrens wird ein grafisches Entwicklungswerkzeug entworfen und realisiert. Dieses Programm wird benötigt, um den Entwurf von Grammatiken im Rahmen eines Lernsystems zu unterstützen, indem die Ergebnisse, insbesondere die Strukturbeschreibungen und Regelverletzungen, am Bildschirm validiert werden können. Aufgrund von Experimenten mit einer Beispielgrammatik ist die Leistungsfähigkeit, insbesondere die Diagnosequalität und das Zeitverhalten des Ansatzes, zu beurteilen.

1.3 Aufbau der Arbeit

In Kapitel 2 werden Diagnosetechniken diskutiert, die Regelverletzungen in meist freien Äußerungen feststellen und identifizieren können. Ausgehend von einer kurzen Einführung in wissensbasierte Lernsysteme und einer Erläuterung der für das integrative Verfahren relevanten Module werden die in dieser Arbeit betrachteten Fehlerklassen eingeführt. Die bisher dafür verwendeten Diagnosetechniken werden vorgestellt und klassifiziert.

In Kapitel 3 wird das Verfahren der Fehlersimulation behandelt. Einleitend wird das grundlegende Prinzip, die modellbasierte Diagnose, erläutert. Anschließend wird die Anwendung dieser Diagnosetechnik für die Verarbeitung fehlerhafter Äußerungen beschrieben. Eine formale Definition eines Grammatikmodells für das Verfahren der Fehlersimulation wird vorgeschlagen. Darauf aufbauend wird die Semantik der Bedingungen, unter Verwendung eines Wertaktualisierungsoperators, entwickelt. Die Diagnosequalität und die Modellierung grammatischen Wissens werden anhand von Beispielen demonstriert.

Das CDG-Parsingverfahren wird in Kapitel 4 erläutert. Es wird versucht, eine formale Definition für eine Grammatik unter Berücksichtigung lexikalischer Mehrdeutigkeit anzugeben. Die Identifikation und Lokalisierung von Regelverletzungen wird durch den Begriff des *Konflikts* in einer optimalen Strukturbeschreibung charakterisiert. Die Anwendung als Diagnoseverfahren wird anhand von Modellierungstechniken und Beispielen diskutiert.

In Kapitel 5 wird das integrative Verfahren vorgestellt. Dabei werden Verbesserungsmöglichkeiten des CDG-Parsings anhand von Effizienz- und Modellierungsproblemen aufgezeigt. Die Architektur des realisierten Verfahrens wird erläutert und der Formalismus für die Transformation von Strukturbeschreibungen in Modelle der Fehlersimulation vorgestellt. Die Evaluation des Verfahrens wird anhand von zwei Grammatikvarianten mithilfe des im Rahmen dieser Arbeit entwickelten Programms *Grammar Diagnosis* durchgeführt. Die Realisierung und die Funktion des Entwicklungswerkzeugs werden kurz beschrieben.

Eine Zusammenfassung der Ergebnisse, die Betrachtung von noch bestehenden Problemen und ein Ausblick auf zukünftige Arbeiten wird in Kapitel 6 gegeben.

Im Anhang A befindet sich eine Beispielgrammatik in dem vorgestellten Formalismus für das integrative Verfahren. Varianten dieser Grammatik werden für die Problemanalyse und die Evaluation verwendet. Die wichtigsten Klassen und Methoden des implementierten Verfahrens, die die Schnittstelle für ein Anwendungsprogramm definieren, befinden sich im Anhang B.

Kapitel 2

Diagnosetechniken in Fremdsprachenlernsystemen

Dieses Kapitel versucht, einerseits das Anwendungsgebiet des in dieser Arbeit entwickelten Verfahrens aufzuzeigen und andererseits bestehende Verfahren, die bisher in diesem Zusammenhang verwendet wurden, vorzustellen und die relevanten Techniken zu klassifizieren. In Abschnitt 2.1 werden wissensbasierte Lernsysteme kurz eingeführt. Anhand eines beispielhaften Fremdsprachenlernsystems werden diejenigen Komponenten in der Systemarchitektur eines Lernsystems identifiziert, die das integrative Verfahren sinnvoll einsetzen können. Der Abschnitt 2.2 stellt Begriffe, Phänomene und Mechanismen vor, die für die Verarbeitung von fehlerhaften Äußerungen relevant sind. Die möglichen Techniken für die Realisierung von Verfahren, die sowohl eine robuste Analyse als auch eine Erkennung des Fehlers und eine Identifikation der Fehlerstelle ermöglichen, werden in Abschnitt 2.3 diskutiert.

2.1 Wissensbasierte Lernsysteme

Der Einsatz des Computers als Lehrmedium ist beinahe so alt wie der Computer selbst.[1] Die ersten Systeme in den fünfziger Jahren beschränkten sich darauf, dem Schüler einzelne Aufgaben zu stellen und auf eine Antwort zu warten. Die Fortführung der Übung mit der nächsten Frage war erst nach einer korrekten Antwort möglich. Die im System definierten Aufgaben bekamen genau eine, in späteren Systemen auch mehrere Lösungen zugeordnet. Die Rückmeldung bestand nur aus einer Zurückweisung von fehlerhaften Antworten. Die Fragen wurden in einer festen, linearen Reihenfolge präsentiert. In den sechziger Jahren wurden in Abhängigkeit der Schülerantwort verschiedene Zweige in der Darstellungsreihenfolge verwendet. Auch in diesen Systemen war die Reihenfolge (z.B. als Baumstruktur) fest ins System kodiert. Computerunterstütztes Lernen (auch *Computer Aided Learning*, *Computer Assisted Learning*, kurz CAL oder *Computer Aided Instruction*, kurz CAI) erwies sich als wenig flexibel im Umgang mit unerwarteten, d.h. nicht antizipierten Antworten des Lernenden.

[1] Die Entwicklungsgeschichte von Lernsystemen wird z.B. in [Witschital 1990] und [Lelouche 1999] dargestellt.

Ebensowenig waren die Systeme am Vorbild eines realen Lehrers mit all seinen Kompetenzen ausgerichtet. Mit der Einführung der Künstlichen Intelligenz versuchte man, sich an dem Wissen und dem Verhalten des Lehrers zu orientieren und daraus ein Modell zu entwerfen. Die resultierenden Systeme Anfang der achtziger Jahre werden als wissensbasierte Lernsysteme (auch *Intelligent Tutoring Systems*, kurz ITS oder ICAL) bezeichnet. Dem hohen Anspruch an diese Systeme, die Kompetenzen eines Tutors auf allen Ebenen zu modellieren, können sie aber noch nicht gerecht werden. Die Fähigkeiten eines Lehrers, selbständig Aufgaben zu stellen und zu lösen, die Antwort des Schülers mit der richtigen Lösung zu vergleichen und die tatsächlichen Fehler und die möglichen Ursachen zu identifizieren sowie einen geeigneten Zeitpunkt und eine geeignete Form einer Intervention zu bestimmen als auch die Kommunikationsform mit dem Schüler angemessen zu gestalten, sind dermaßen komplex, daß auch in Zukunft kein System einen Lehrer ersetzen können wird. Wird das Ziel bescheidener formuliert, z.B. in dem Sinne, daß das System das Erlernen von Zusammenhängen und Problemlösungsfähigkeiten für Teilbereiche des Lerngegenstandes unterstützt, können stark eingeschränkte Modelle der Kompetenzen einen hilfreichen Beitrag leisten. Der Einsatz des Computers als Lehrmedium hat einige Vorzüge. Er erlaubt individuelles Lernen, einfache Kontrolle des Lernfortschritts, die Entlastung des Lehrers von Routineaufgaben und kann an jedem beliebigen Ort und zu jedem beliebigen Zeitpunkt verwendet werden ([Lusti 1992]). Außerdem ist die Akzeptanz der zukunftsträchtigen Computertechnologie aus der Sicht der Schüler sehr groß, zumal schon in fast jedem Berufszweig eine grundsätzliche Vertrautheit mit diesem Arbeitsmittel erforderlich ist.

2.1.1 Architektur von wissensbasierten Lernsystemen

Für die Realisierung eines wissensbasierten Lernsystems ist es meist notwendig, eine modulare Systemarchitektur anzunehmen, deren einzelne Komponenten bestimmte Kompetenzebenen des Lehrers modellieren. Abbildung 2.1 zeigt eine mögliche Architektur nach [Lusti 1992], die insbesondere den Ablauf des Systems näher charakterisiert.

Das Unterrichtsmodul ist verantwortlich für den Inhalt und den Zeitpunkt der zu präsentierenden Information. Diese Information ist entweder eine Aufgabenstellung oder eine Fehlererklärung. Sie wird an das Kommunikationsmodul weitergeleitet, welches für die Auswahl einer geeigneten Interaktionsform sorgen muß. Das Expertenmodul versucht, die Problemlösungsfähigkeit des Lehrers abzubilden, während das Studentenmodul eine Buchhaltung über die Lösungsfähigkeiten des Schülers führt. Schwerpunkt vieler Studentenmodule ist neben der Modellierung des aktuellen Wissensstandes des Lernenden die *Fehlerdiagnose*, d.h. die Identifizierung von Abweichungen zwischen der Schülerantwort und der Expertenlösung.

Die Module realisieren verschiedene Verarbeitungsprozesse, die implizit eine Repräsentation des für die jeweilige Ebene relevanten Wissens erfordern. Die Wissensmodelle sind in der Regel nicht exklusiv für einen Verarbeitungsprozeß bestimmt, sondern werden von verschiedenen Modulen benötigt. Die

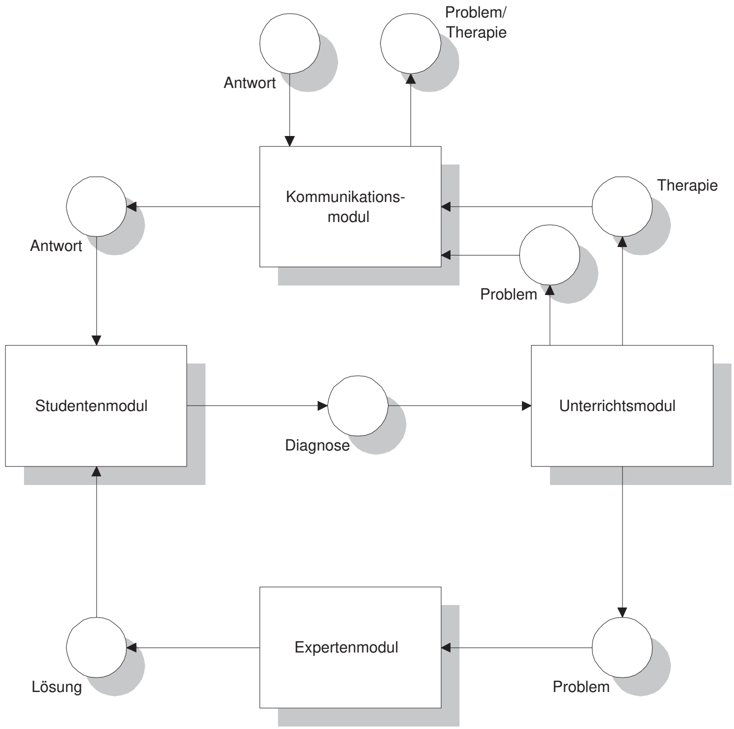

Abbildung 2.1: Architektur von wissensbasierten Lernsystemen nach [Lusti 1992].

Einschätzung des Lehrers über die Fähigkeiten des Studenten wird im Studentenmodell repräsentiert. Dieses Wissen kann verwendet werden, um im Unterrichtsmodul eine geeignete Übung auszuwählen, die z.B. gerade gelernte Sachverhalte vertieft oder Probleme des Schülers durch alternative Übungen zu beseitigen versucht. Ebenso kann eine andere Kommunikationsform, z.B. eine grafische Darstellung der Zusammenhänge, vom Schüler leichter verstanden werden als z.B. eine textuelle Erklärung eines Fehlers.

2.1.2 Szenario eines Fremdsprachenlernsystems

Das in dieser Arbeit entwickelte Verfahren kann in Fremdsprachenlernsystemen (auch *(Intelligent) Computer Aided Language Learning*, kurz CALL oder ICALL) verwendet werden. Ein Lernsystem könnte z.B. wie folgt realisiert werden: Einem Schüler wird ein kurzer Film vorgespielt. Anschließend wird er vom System aufgefordert, den Inhalt des Films in eigenen Worten zusammenzufassen. Die Aufgabe des Lernsystems soll darin bestehen, die Antworten des Schülers sowohl inhaltlich als auch grammatikalisch auf ihre Richtigkeit zu überprüfen. Das Lernziel besteht in der Unterstützung des kreativen Sprachgebrauchs in einem kommunikativ relevanten Anwendungszusammenhang. Liegt ein Fehler in der Eingabe vor, muß das System in der Lage sein, eine für den Schüler verständliche Erklärung auszugeben. Das Expertenmodul benötigt sowohl eine Repräsentation des sprachlichen Wissens eines Lehrers als auch ein

Modell für den Inhalt des Films. In diesem Szenario erscheint es sinnvoll, das Studentenmodul als Bestandteil des Expertenmoduls anzusehen ist, da die Generierung einer Lösung und die Feststellung der Abweichungen nicht voneinander unabhängig sind.[2] Der Lehrer wird versuchen, die Antwort zu analysieren und dabei die Abweichungen bezüglich einer möglichst ähnlichen, aber korrekten Äußerung festzustellen. Er ist dabei in der Lage, auch für relativ stark gestörte Äußerungen sowohl die Bedeutung als auch die strukturellen, grammatischen Beziehungen innerhalb der Äußerung zu ermitteln.

2.2 Parsing natürlicher Sprache

Menschliche Sprachverarbeitungsprozesse erfolgen kompetenzbasiert, d.h. es wird eine mentale Repräsentation sprachlichen Wissens, der grammatischen *Kompetenz*, angenommen. Ein Kriterium der Kompetenz ist z.B. die Fähigkeit, grammatisch korrekte Sätze von fehlerhaften Sätzen unterscheiden zu können ([Grewendorf 1996]). Die auf der Grundlage der Kompetenz tatsächlich beobachtbaren Verarbeitungsleistungen des Menschen bestimmen die sprachliche *Performanz*. In der Künstlichen Intelligenz wird auch von Wissensrepräsentation (Kompetenz) und Wissensverarbeitung (Performanz) gesprochen ([Hellwig 1989]).

Ein Modell der Kompetenz wird als *Grammatik* bezeichnet. Für die maschinelle Verarbeitung wird ein formales, mathematisches Modell benötigt. Dieses Modell bestimmt die als grammatisch anzusehenden Eingabefolgen über einem Alphabet. Da im folgenden meist nur syntaktische Aspekte einer natürlichen Sprache betrachtet werden, werden die Elemente des Alphabets als *Worte* oder *Wortformen* und eine beliebige Eingabefolge über dem Alphabet als *Äußerung* oder *Satz* bezeichnet. Die formale Sprache einer Grammatik ist eine Teilmenge der Menge aller möglichen Äußerungen. Ist ein Satz ein Element der formalen Sprache, wird er als *wohlgeformt* bezeichnet.

Unter dem Begriff des *Parsings* wird ein Verarbeitungsprozeß verstanden, welcher einer Äußerung bezüglich einer Grammatik eine Strukturbeschreibung zuweist.[3] Dabei ist eine Struktur mathematisch eine Menge von Elementen und Relationen auf diesen Elementen. In der Computerlinguistik wird für den Prozeß der Strukturerstellung meist implizit eine spezifische Ebene des Sprachverstehens, i.a. die Syntax, angenommen ([Hellwig 1989]). Dabei werden vornehmlich Baumstrukturen für die Beschreibung der Beziehungen zwischen den Bestandteilen der Äußerung verwendet, wobei verschiedene linguistische Theorien unterschiedliche Strukturbeschreibungen zu Grunde legen.

[2] Aufgrund der Kreativität menschlichen Sprachgebrauchs kann nicht erwartet werden, daß die Lösung des Experten und die Antwort des Schülers miteinander vergleichbar sind. Dies kann nur für stark eingeschränkte Übungen, z.B. in Form von Lückentexten, sichergestellt werden.

[3] In [Naumann 1994] wird eine Einführung in das Parsing gegeben, wobei der Schwerpunkt auf Anwendungen in der Computerlinguistik liegt. Das Parsing von Programmiersprachen wird ausführlich in [Aho 1988] behandelt.

2.2.1 Extra- und Ungrammatikalität

Ein Modell der sprachlichen Kompetenz beschreibt die Menge der wohlgeformten Sätze. Alle Sätze außerhalb dieser Menge werden nach [Carbonell 1983] als *extragrammatikalisch* bezeichnet. Im folgenden wird auch von *fehlerhaften* und *korrekten* bzw. wohlgeformten Sätzen gesprochen, wobei die Begriffe sich immer auf das Modell beziehen. Da ein Modell nur die für einen bestimmten Zweck relevante Information repräsentiert, kann nicht davon ausgegangen werden, daß die Menge der von einem menschlichen Sprecher als grammatisch angesehenen Sätze identisch ist mit der Menge der wohlgeformten Sätze des Modells. Aufgrund dieser Modellgrenzen wird es immer Sätze geben, die der Parser als fehlerhaft ansieht, während die meisten Menschen den Satz als grammatisch einstufen würden. Die vom Menschen als unakzeptabel empfundenen Sätze werden als *ungrammatikalisch* bezeichnet.[4]

Die Unterscheidung zwischen Extra- und Ungrammatikalität wird in Abbildung 2.2 dargestellt. Die Menge der wohlgeformten Sätze sei L, die Menge der von einem kompetenten Sprecher als grammatisch empfundenen Sätze sei L' und die Menge aller Äußerungen über einem Alphabet sei O mit $L \subseteq O$ und $L' \subseteq O$. Die extragrammatikalischen, fehlerhaften Sätze sind aus O/L, während die ungrammatischen Sätze aus O/L' sind. Ein Modell der Kompetenz ist *vollständig*, falls $\forall \omega \in O : \omega \in L' \Rightarrow \omega \in L$ gilt, und es ist *korrekt*, falls $L \subseteq L'$ gilt. Eine vollständige und korrekte Grammatik wird als *beobachtungsadäquat* bezeichnet.[5] Stimmen zusätzlich noch die generierten Strukturbeschreibungen des Parsers mit den erwarteten, intuitiv als korrekt empfundenen Strukturbeschreibungen überein, wird das Modell als *beschreibungsadäquat* bezeichnet. In Abbildung 2.2 werden die Modellierungsfehler verdeutlicht: Die Grammatik ist weder vollständig noch korrekt als auch nicht beschreibungsadäquat. Häufig kann die Korrektheit des Modells nur garantiert werden, wenn ein relativ stark reduzierter Teilbereich der grammatischen Kompetenz modelliert wird. Hagen (1994) stellt dazu kritisch fest:

> „*In fact, the field of applied linguistics is somewhat notorious for overstating the computer's prowess, and anyone contemplating onsite development would do well to bear in mind that if AI research has taught us anything over the past few decades, it is that the success of an intelligent tutoring system is usually inversely related to breadth of its knowledge domain.*“ [Hagen 1994, S.8]

Die Verarbeitung von fehlerhaften Sätzen ist für ein sprachverarbeitendes System in vielen Fällen unerläßlich, da meist nicht davon ausgegangen werden kann, daß ein Benutzer sich strikt an die formalen Vorgaben der Grammatik hält. Ein derartiges System benötigt einen *robusten* Parser. In der Softwaretechnik wird unter dem Begriff *Robustheit* ein stabiles Systemverhalten bei unerwarteter Eingabe verstanden. Damit wird nicht unbedingt gefordert, daß das

[4] In [Weischedel 1983] wird von relativer und absoluter Nichtwohlgeformtheit gesprochen. Ersteres bezieht sich auf die Sätze außerhalb des Modells, während ein absolut nichtwohlgeformter Satz einem ungrammatischen Satz entspricht.

[5] Die Kriterien zur Beurteilung der Adäquatheit einer Grammatik werden z.B. in [Grewendorf 1996, S.40] behandelt.

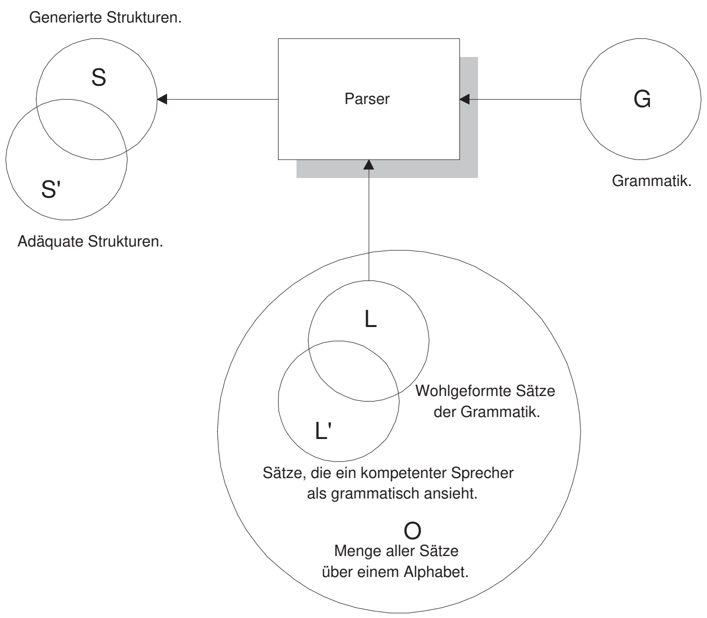

Abbildung 2.2: Modellierungsfehler der sprachlichen Kompetenz

System die Intention des Benutzers bei unerwarteter Eingabe erkennt und umsetzt. Hierfür wird der Begriff *Fehlertoleranz* verwendet. In der Sprachverarbeitung wird diese Trennung in der Terminologie meist aufgehoben:

> „*The notion of robustness in natural language processing is a rather broad one and lacks a precise definition. Usually, it is taken to describe a kind of monotonic behaviour, which should be guaranteed whenever a system is exposed to some sort of non-standard input data: A comparatively small deviation from a predefined ideal should lead to no or only minor disturbances in the system's response, whereas a total failure might only be accepted for sufficiently distorted input.*" [Menzel 1995, S.15]

Ein robuster Parser muß demnach in der Lage sein, auch für einige ungrammatische Äußerungen eine Strukturbeschreibung generieren zu können. Dabei ist es meist notwendig, weiteres Wissen, insbesondere semantische und pragmatische Aspekte, in den Verarbeitungsprozeß miteinzubeziehen. In Analogie zu technischen Systemen, in denen es üblich ist, kritische Prozesse durch eine redundante Modellierung sicherer zu gestalten, erlaubt eine redundante Wissensmodellierung der sprachlichen Kompetenz auf verschiedenen Verarbeitungsebenen einen höheren Grad an Robustheit:

> „*The more knowledge one has available, the better for natural language understanding. While this is not a particulary insight, it can be concluded* [...] *that semantic and/or domain knowledge are even more important when a system is required to be robust. When expectations are violated, higher-level knowledge is of great help in fin-*

> *ding the cause of the violation and/or arriving at a sentence interpretation even without explicitly resolving the problem.*“ [Stede 1992, S.400]

In einem Fremdsprachenlernsystem ist ein robustes Parsingverfahren noch nicht ausreichend. Das Ziel besteht nicht unbedingt darin, eine Verarbeitung von ungrammatischen Äußerungen zu ermöglichen, sondern vielmehr darin, den eigentlichen Fehler festzustellen, die Ursache des Fehlers zu ermitteln und die nicht beachteten Regularitäten der sprachlichen Kompetenz aufzuzeigen. Dazu verwendet Schwind (1988) den Begriff des *fehlersensitiven* Parsings:

> „[...]*we are attempting to perform sensitive parsing rather than robust parsing. This means that we aim to achieve a system which is very sensitive to all possible kinds of errors and which moreover tries to find the ‘why’ and the ‘how’ of errors.*“ [Schwind 1988, S.608]

Das Feststellen eines Fehlers in einer Äußerung ist sehr einfach, da nur überprüft werden muß, ob der Analyseprozeß scheitert. Wesentlich schwieriger ist es jedoch, eine für den Schüler plausible Erklärung zu generieren. Dazu ist es notwendig, die verletzten Wissensbestandteile der formalen Grammatik zu ermitteln. Diese sollten so beschaffen sein, daß aus ihnen eine adäquate, verbale Erklärung generiert werden kann. Stede (1992) stellt dazu folgendes fest:

> „*It is easy to recognize the presence of an error (the parser won't succeed), but it is very hard to identify it, which includes the task of exactly locating it.*“ [Stede 1992, S.390]

Eine Erklärung kann meist nur dann vernünftig generiert werden, wenn die strukturellen Beziehungen in der Äußerung bekannt sind, d.h. es wird ein robustes Parsingverfahren benötigt. Im Unterschied zu Anwendungen, in denen eine semantische Repräsentation der fehlerhaften Äußerung völlig ausreicht, um die Funktionalität zur Verfügung zu stellen, ist in Fremdsprachenlernsystemen eine detaillierte Modellierung der grammatischen Kompetenz, insbesondere der Syntax, notwendig. Im folgenden werden robuste Parsingverfahren, die sich ausschließlich auf eine semantische Modellierung stützen, nicht weiter betrachtet. Eine Übersicht über derartige Verfahren findet sich in [Stede 1992].

2.2.2 Extragrammatikalische Phänomene

Eine Zusammenstellung von relevanten Phänomenen, die die Verarbeitung von fehlerhaften Äußerungen betreffen, findet sich in [Carbonell 1983] und [Stede 1992]. Die Phänomene werden nach verschiedenen Verarbeitungsebenen unter Annahme einer modularen Architektur der sprachlichen Kompetenz klassifiziert.

Das lexikalische Wissen repräsentiert das Wissen über die Eigenschaften von Wortformen. Zu dieser Kategorie werden z.B. Rechtschreibfehler und Tippfehler gerechnet. Weiterhin können Wörter verwendet werden, die außerhalb der Modellgrenzen liegen. Schwierigkeiten für die Verarbeitung bereiten auch unbekannte Eigennamen, Wortkomposita und Wortsegmentierungsfehler. Sowohl lexikalische Fehler als auch pragmatische Fehler werden in dieser Arbeit nicht weiter behandelt. Mögliche Techniken für die Verarbeitung dieser Phänomene werden z.B. in [Carbonell 1983] und [Stede 1992] vorgestellt. In dieser Arbeit werden Fehler untersucht, die syntaktisches und eingeschränkt semantisches Wissen betreffen. Die relevanten Phänomene werden im Einzelnen vorgestellt:

- Fehlende Konstituenten: Diese Fehlerklasse kennzeichnet ungrammatische Äußerungen, in denen obligatorische Konstituenten fehlen, z.B. ein erforderlicher Artikel in einer Substantivgruppe oder ein fehlendes Argument eines transitiven Verbs.

- Überflüssige Konstituenten: In der Äußerung werden Wortformen verwendet, die nicht sinnvoll interpretiert werden können, d.h. sie sind in der Regel nicht in der Strukturbeschreibung enthalten. Die Analyse von nicht notwendigerweise zusammenhängenden Teilbereichen einer Äußerung wird als *partielles* Parsing bezeichnet.

- Anordnungsfehler: Die Anordnung von Wortformen in einer Äußerung ist meist strengen Regeln unterworfen: Im Französischen steht ein Adjektiv z.B. hinter dem modifizierten Substantiv, während im Deutschen eine vorangestellte Anordnung zwingend ist.

- Kongruenzverletzungen: Gerade in morphologisch reichen Sprachen sind Fehler dieser Art besonders häufig. Dazu zählen z.B. die Kongruenz im Numerus, Kasus und Genus in einer Substantivgruppe oder die Kongruenz von finitem Verb mit dem Subjekt in Numerus und Person.

- Verletzung selektionaler Beschränkungen: Die selektionalen Beschränkungen, insbesondere von Verben, sind der einzige semantische Aspekt, der in dieser Arbeit behandelt wird. Zum Beispiel kann für die Sättigung semantischer Valenzen von Verben zwischen belebten und unbelebten Konzepten unterschieden werden.

Die Ursache dieser Fehler kann neben der Verletzung von regelhaften Beschreibungen der sprachlichen Kompetenz der Zielsprache (*Regelfehler*) auch unzureichendes Faktenwissen, d.h. unzureichende Kenntnisse der lexikalischen Eigenschaften der Wortformen, sein (*Faktenfehler*). Häufig werden auch *Inferenzfehler* betrachtet, d.h. Fehler, deren Ursache eine Übertragung der Regularitäten der Muttersprache auf die Zielsprache ist.

Eine Beschränkung auf rein syntaktische Aspekte ist in einem Lernsystem, welches den kreativen Sprachgebrauch in einem motivierenden Anwendungszusammenhang ermöglicht, sicherlich nicht ausreichend. Der in Kapitel 4 vorgestellte Formalismus kann jedoch für eine eingeschränkte Modellierung von pragmatischem Wissen verwendet werden.

Die Verarbeitung von fehlerhaften Äußerungen, die insbesondere in die oben genannten Fehlerklassen fallen, erfordert häufig entweder eine Modifikation des Formalismus oder eine Modifikation der Grammatik. Die in der Literatur verwendeten Verfahren sind in der Regel Anpassungen der jeweils verwendeten Formalismen. Fast allen Formalismen liegen zwei grundsätzliche Mechanismen zugrunde, die im folgenden vorgestellt werden.

2.2.3 Unifikationsbasierte Phrasenstrukturgrammatiken

Ein *Grammatikformalismus* charakterisiert die Menge aller potentiellen Grammatiken eines bestimmten Typs, d.h. der Formalismus bestimmt, wie Grammatiken notiert werden können. Neben den syntaktischen Eigenschaften definiert er, wie die einzelnen Elemente der Grammatik interpretiert werden, um die formale Sprache zu charakterisieren. Ein einfacher Formalismus ist der der kontextfreien Phrasenstrukturgrammatiken. Eine kontextfreie Grammatik besteht aus einer endlichen Menge von Terminal- und Nichtterminalsymbolen, einer endlichen Menge von Regeln und einem Startsymbol ([Hopcroft 1994]). Die Regeln haben die Form $A \rightarrow \alpha$, wobei A ein Nichtterminalsymbol und α eine Folge von Terminal- und/oder Nichtterminalsymbolen ist. Die Semantik des Ausdrucks besagt, daß man in einer Eingabefolge ein Vorkommen von α durch das Symbol A ersetzen kann. Die formale Sprache der Grammatik besteht dann aus denjenigen Eingabefolgen, die sich auf das Startsymbol reduzieren lassen.[6] Eine logische Repräsentation einer kontextfreien Regel verdeutlicht die implizit enthaltenen Bedingungen: Eine Regel $\mathsf{NP} \rightarrow \mathsf{DET} \quad \mathsf{N}$ kann z.B. wie folgt in prädikatenlogischer Notation dargestellt werden:

$$\mathsf{DET}(x) \wedge \mathsf{N}(y) \wedge \mathsf{concat}(x, y, z) \Rightarrow \mathsf{NP}(z)$$

Die Formel $\mathsf{NP}(z)$ ist genau dann wahr, wenn die Äußerung z sich so in zwei Teiläußerungen x und y unterteilen läßt, daß x einen Artikel und y ein Substantiv darstellt und x unmittelbar vor y in der Eingabesequenz steht. In einer kontextfreien Regel sind daher zwei verschiedene Relationen enthalten, die Relation der unmittelbaren Dominanz (NP dominiert unmittelbar DET und N) und der linearen Anordnung (die Konstituente DET steht unmittelbar vor der Konstituente N). Eine Trennung beider Relationen wurde erstmals im Formalismus der GPSG (*Generalized Phrase Structure Grammar*, [Gazdar 1985]) eingeführt.

Die meisten in der Computerlinguistik gebräuchlichen Formalismen erlauben neben der Modellierung von kontextfreien Regeln einschränkende Bedingungen, die die Anwendung einer Regel nur unter bestimmten Voraussetzungen gestatten. Diese Formalismen werden als *unifikationsbasiert* bezeichnet.[7] Kennzeichnend für diese Formalismen ist einerseits die Repräsentation sprachlichen Wissens durch Merkmalsstrukturen und andererseits die Möglichkeit, Merkmalsstrukturen miteinander zu vergleichen (Subsumtionsrelation) und dadurch eine partielle Ordnung herzustellen. Der Operator, welcher für zwei gegebene Merkmalsstrukturen diejenige minimale Struktur bestimmt, welche von beiden subsumiert wird, wird als *Unifikationsoperator* bezeichnet.

Eine Merkmalsstruktur kann als eine Menge von Merkmals-Wert-Paaren oder nach [Shieber 1987] als partielle Funktion von einer Menge von Merkmalen auf eine Menge von Werten angesehen werden. Dabei können Werte entweder

[6] Es gibt zahlreiche Möglichkeiten, die formale Sprache einer kontextfreien Grammatik zu charakterisieren, z.B. durch die Menge aller Eingabefolgen, die sich aus dem Startsymbol konstruieren lassen (*Top-down* Analyse). Alternativ kann auch eine Regel als partieller Baum aufgefaßt und die formale Sprache durch diejenigen Eingabefolgen definiert werden, für die es einen überspannenden Baum gibt.

[7] Eine gute Einführung liefert [Shieber 1987].

atomar oder komplex sein, wobei komplexe Werte selbst wieder Merkmalsstrukturen sind. Zwei atomare oder zwei komplexe Werte können nicht nur gleich, sondern auch identisch sein, d.h. sie referenzieren auf einen vorhandenen Wert, anstatt diesen Wert zu duplizieren. Diese Eigenschaft wird als Koreferenz bezeichnet und gestattet es insbesondere, in unifikationsbasierten Phrasenstrukturgrammatiken Merkmalswerte zu transportieren (*feature perlocation*).

Als einfaches Beispiel wird eine Regel im Grammatikformalismus PATRII ([Shieber 1987]) betrachtet, welche die Kongruenz zwischen Subjekt und finitem Verb in Numerus und Person modelliert. Die Merkmale werden von den jeweiligen lexikalischen Köpfen an die zugehörige Phrase transportiert.

$$
\begin{array}{l}
\mathsf{X0} \quad \rightarrow \quad \mathsf{X1} \quad \mathsf{X2} \\
\langle \mathsf{X0}\ \ \mathsf{cat} \rangle = \mathsf{S} \\
\langle \mathsf{X1}\ \ \mathsf{cat} \rangle = \mathsf{NP} \\
\langle \mathsf{X2}\ \ \mathsf{cat} \rangle = \mathsf{VP} \\
\langle \mathsf{X0} \rangle = \langle \mathsf{X2} \rangle \\
\langle \mathsf{X0}\ \ \mathsf{pers} \rangle = \langle \mathsf{X1}\ \ \mathsf{pers} \rangle \\
\langle \mathsf{X0}\ \ \mathsf{num} \rangle = \langle \mathsf{X1}\ \ \mathsf{num} \rangle
\end{array}
$$

Im folgenden wird jedes regelhafte Element der Grammatik, z.B. Unifikationsgleichungen oder kontextfreie Regeln, als Bedingung (*constraint*) angesehen. Unter einer *Wohlgeformtheitsbedingung* oder einer *normalen* Regel wird eine Bedingung verstanden, die Bestandteil eines Modells der sprachlichen Kompetenz ist, d.h. die für die Beschreibung der als grammatisch anzusehenden Sätze verwendet wird. Unter *merkmalsbasierten* Wohlgeformtheitsbedingungen sollen Bedingungen verstanden werden, deren Wahrheitswerte sich aus den Elementen der beteiligten Merkmalsstrukturen ergeben. Eine *Fehlerregel* bezeichnet eine Regel, die für die Ableitung ungrammatischer Sätze verwendet wird. Eine *Fehlergrammatik* ist ein formales Modell, welches neben den grammatischen auch einige ungrammatische Sätze beschreibt.

2.3 Diagnosetechniken

Der Begriff der *Diagnostik* wird von Puppe (1996) synonym zum Begriff der Klassifikation verwendet. Darunter versteht er einen Lösungsprozeß für Probleme mit folgenden Eigenschaften:

- Der Problembereich besteht aus zwei endlichen, disjunkten Mengen von Problemmerkmalen (Merkmale, Symptome) und Problemlösungen (Lösungen, Diagnosen) und aus typischerweise unsicherem, mehrstufigem Wissen über die Beziehungen zwischen Merkmalen und Lösungen.
- Ein Problem ist durch eine eventuell unvollständig gegebene Teilmenge von Merkmalswerten charakterisiert.

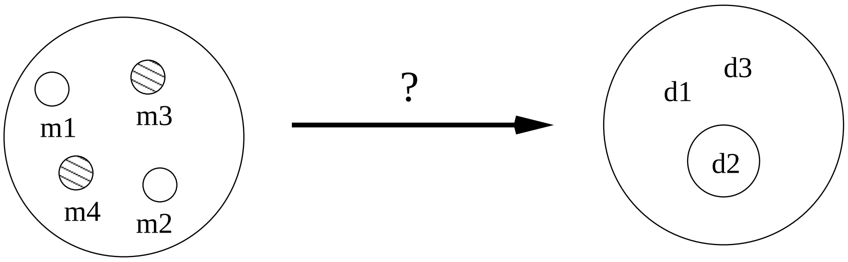

Abbildung 2.3: Grundprinzip der Diagnostik nach [Puppe 1996]. Es werden nur binäre Merkmalswerte symbolisiert.

- Das Ergebnis ist die Auswahl einer oder mehrerer der Lösungen.
- Eine Teilaufgabe der Diagnostik ist, zu bestimmen, ob und welche zusätzlichen Merkmale zur Verbesserung der Qualität der Problemlösung angefordert werden sollen.

[Puppe 1996, S.3]

Die letzte Teilaufgabe der Diagnostik, Messungen zur Verbesserung der Qualität durchzuführen, macht in Fremdsprachenlernsystemen nur in Ausnahmefällen einen Sinn.[8] Das Grundprinzip wird in Abbildung 2.3 veranschaulicht. Diagnostik kann als eine Abbildung aller möglichen Kombinationen der Symptomwerte in die Potenzmenge der Lösungen angesehen werden.

In vielen Anwendungsfeldern der Diagnostik, z.B. der Medizin oder in technischen Systemen, reicht es meistens aus, nur Einfehlerhypothesen zu betrachten. Diese Grundannahme ist in einem sprachverarbeitenden System aber nicht gerechtfertigt, da Mehrfachfehler häufig zu beobachten sind. Die Auffassung der Diagnostik als ein Auswahlprozeß ist nur für eingeschränkte Übungsformen, z.B. Lückentextübungen, sinnvoll. Für die Unterstützung des kreativen Sprachgebrauchs ist es unerläßlich, freie Äußerungen verarbeiten zu können. Puppe unterscheidet die Problemklasse der Diagnostik von der schwierigeren Problemklasse der Konstruktion:

> „Nicht alle Problembereiche lassen sich durch Auswahl lösen. Oft muß die Lösung aus primitiven Bausteinen zusammengesetzt werden [...]. Diese Problemklasse nennen wir Konstruktion, zu der viele Untertypen wie Konfigurierung, Planung, Zuordnung usw. gehören. Im allgemeinen gilt, daß konstruktive Probleme schwerer zu lösen sind als Diagnostikprobleme, da der zu betrachtende Raum der möglichen Lösungen bei der Konstruktion in der Regel wesentlich größer ist.“ [Puppe 1996, S.1]

Möchte man das abstrakte Modell von Puppe für den Anwendungsbereich von Fremdsprachenlernsystemen verwenden, können z.B. für die Problemmerkmale die beobachteten Eigenschaften der Wortformen, repräsentiert durch Merk-

[8] Im Falle von unbekannten Wörtern könnte der Schüler aufgefordert werden, ein anderes, aber bedeutungsgleiches Wort für das unbekannte anzugeben.

malsstrukturen, in der Äußerung verwendet werden. Die Bedingungen der Grammatik können als Problemlösungen angesehen werden. Die Aufgabe besteht dann darin, für eine fehlerhafte Äußerung diejenigen Bedingungen der Grammatik zu identifizieren, die eine Erklärung des Fehlers ermöglichen.

Die Diagnose von Fehlern in freien Äußerungen kann als Konstruktionsproblem angesehen werden: Es wird der Versuch unternommen, mit den vorhandenen Regeln der Grammatik eine Strukturbeschreibung bzw. eine inhaltliche Bedeutung der Äußerung zu konstruieren. Diejenigen Elemente der Grammatik, die während des Konstruktionsprozesses nicht berücksichtigt oder nur in modifizierter Form verwendet werden konnten, kennzeichnen die Menge von (primitiven) Diagnosen.

Die vorherrschenden Diagnosetechniken in Fremdsprachenlernsystemen basieren entweder auf der Verwendung von Fehlerregeln, also der expliziten Formulierung des Fehlerfalls, oder auf der Möglichkeit der Bedingungsrücknahme, d.h. Regeln können bis zu einem gewissen Grad unbeachtet bleiben. Die meisten Arbeiten beschränken sich dabei auf die Betrachtung von syntaktischen Phänomenen, wie z.B. Menzel (1995) bemerkt:

> „*Most of the work has been concerned with the problem from a purely syntactic point of view and usually relied on two basic techniques: error anticipation and constraint relaxation.*“ [Menzel 1995, S.20]

Betrachtet man die Wissensbasis für die Modellierung der sprachlichen Kompetenz, so besteht sie aus logischer Sicht aus einer Menge von Fakten und einer Menge von Regeln. Regeln sind dabei Prädikate, deren Argumente auch Variablen enthalten dürfen. Sie klassifizieren im Unterschied zu dem Faktenwissen eine ganze Klasse von Beispielen. Dafür wird ein Inferenzmechanismus benötigt, welcher aus vorhandenen Fakten und Regeln neue Fakten ableiten kann. In [Menzel 1992a] werden die Diagnosetechniken nach fakten- und regelbasierten Techniken klassifiziert. Fakten kennzeichnen Einzelbeispiele, während Regeln Klassen von Beispielen charakterisieren. Die Anwendung von einer rein faktenbasierten Wissensbasis kann nur in stark eingeschränkten Lernsystemen sinnvoll angewendet werden. Für jede Aufgabe wird eine Menge von Lösungen festgelegt, wobei jede Lösung mit einer Fehlererklärung annotiert wird. Die Wissensbasis kann dann wie folgt als Menge dargestellt werden:

$$\text{Wissen} = \{(\text{Aufgabe}_1, \{(\text{Fehler}_{1,1}, \text{Erklärung}_{1,1}), \ldots\}), \ldots\}$$

Der Aufwand für die Antizipation des Fehlerfalls kann in einer derartigen Wissensrepräsentation als maximal angesehen werden.

Wissensbasierte Lernsysteme charakterisieren sich durch die Verwendung von Regeln in der Wissensbasis. In Abschnitt 2.2.3 haben wir zwei grundsätzlich verschiedene Elemente der Wissensbasis vorgestellt: Unifikationsgleichungen und kontextfreie Regeln. Während eine nicht erfolgreiche Unifikation in nichtrobusten Parsern immer zu einem Abbruch der Analyse führt, kann eine einfache Nichtbeachtung der Unifikationsgleichung schon ausreichen, um eine Strukturbeschreibung generieren zu können. Für kontextfreie Regeln kann aber nicht so argumentiert werden: Da eine kontextfreie Regel verschiedene Bedingungen an die Eingabefolge stellt, ist eine Bedingungsrücknahme nicht ohne weiteres möglich. Dies liegt schon in der Natur der Repräsentation: Während Unifikationsgleichungen konjunktiv verknüpft werden, sind kontextfreie Regeln immer als Disjunktion zu interpretieren. Eine Nichtbeachtung einer Regel reduziert daher die von der Grammatik beschriebene Satzmenge, während eine Nichtbeachtung einer Unifikationsgleichung die Menge vergrößert, wodurch die fehlerhafte Äußerung enthalten sein könnte.

Um die Satzmenge zu vergrößern, gibt es mehrere Möglichkeiten. Es können Unifikationsgleichungen aufgehoben oder weitere kontextfreie Regeln eingeführt werden. Eine dritte Möglichkeit besteht darin, die kontextfreien Regeln so zu modifizieren, daß die fehlerhafte Äußerung akzeptiert wird. Für diesen Fall wird meist ein zweistufiger Verarbeitungsprozeß, erst die normale Grammatik und dann die modifizierte Form, verwendet. Eine vierte Variante betrifft die Modifikation des Inferenzmechanismus: Die Interpretation einer kontextfreien Regel kann so geändert werden, daß auch dann eine Regelanwendung möglich ist, wenn die Eingabefolge bestimmten Abweichungsklassen entspricht. Auch hier ist ein Zweistufenprozeß nötig. Diese vier Ansätze werden im folgenden anhand von Beispielen aus der Literatur erläutert. In einigen Systemen wird eine Kombination der Techniken, insbesondere die Kombination von Fehlerregeln und Bedingungsrücknahme, benutzt. Die verwendeten Grammatikformalismen lassen sich meist auf unifikationsbasierte Phrasenstrukturgrammatiken zurückführen.

2.3.1 Fehlerregeln

Eine normale Regel beschreibt Klassen von Sätzen mit dem Ziel, daß sie mit denjenigen Sätzen übereinstimmen, die ein kompetenter Sprecher als grammatisch einstufen würde. Fehlerregeln sind Regeln, welche Klassen von ungrammatischen Sätzen generieren. Im Vergleich mit rein faktenbasierten Techniken, die nur einzelne Sätze antizipieren können, ist der Modellierungsaufwand deutlich geringer. Allerdings scheint eine Antizipation aller Fehlerfälle unmöglich. Insbesondere für Anordnungsfehler, fehlende und überflüssige Konstituenten wächst die benötigte Regelmenge aufgrund der kombinatorischen Fehlermöglichkeiten erheblich. In allen auf Fehlerregeln basierenden Systemen werden daher in der Regel nur *typische* Fehlerfälle erfaßt. Die typischen Fehlerfälle werden aus im Fremdsprachenunterricht beobachteten Textsammlungen ermittelt. Für eine Teilklasse dieser Fälle können verallgemeinerte Fehlerregeln formuliert werden.

Das System aus [Sanders 1991] verwendet das sogenannte *Miami Corpus*, welches 400 Seiten geschriebenen Text aus dem Deutschunterricht für amerikanische Collegestudenten enthält. Die Textanalyse ergab 24 Prozent morphologische Fehler, wobei die meisten Genusfehler waren. Weiterhin waren 29 Prozent syntaktische, 27 Prozent orthographische und 20 Prozent semantische Fehler. Die Fehlerregeln des Systems unterstützen nur morpho-syntaktische Fehler.

Die Beschreibung typischer Fehlerfälle ist sicherlich abhängig von der Muttersprache des Lernenden, insbesondere derjenigen Fehlerfälle, die auf negativem Sprachtransfer beruhen. Das System aus [Schneider 1998] betrachtet als Zielgruppe Menschen, die die amerikanische Gebärdensprache (*American Sign Language*, kurz ASL) für die Kommunikation verwenden und Englisch lernen wollen. Die Autoren bemerken die erheblichen Unterschiede beider Sprachen:

> „*It is important to note that ASL is not simply a translation of standard English into manual gestures, but rather is a complete language with its own syntax, which is significantly different from English.*“ [Schneider 1998]

Besondere Schwierigkeiten, sowohl in der Modellierung als auch in einer effizienten Realisierung, bereiten häufig beobachtete Fehlerfälle, in denen wichtige Konstituenten weggelassen werden. Fehlende Konstituenten sind z.B. Artikel, Hauptverben, Kopulaverben, Subjekte und Präpositionen. Neben der erheblichen Vergrößerung der Regelmenge wird eine Analyse auch dadurch erschwert, daß die Fehlerregeln für fehlende Konstituenten fast an jeder beliebigen Stelle im Parsingprozeß angewendet werden können. Die Autoren verwenden daher eine Heuristik, die von maximal einer fehlenden Konstituente ausgeht.

Werden Fehlerregeln verwendet, um fehlerhafte Äußerungen verarbeiten zu können, ist es meist zwingend erforderlich, die Anwendung einer Fehlerregel gegenüber einer alternativen normalen Regel zu dispräferieren. Dies ist nicht nur aus Effizienzgründen notwendig. Es sollte auch immer versucht werden, möglichst *wenig* Fehlerregeln in einer Ableitung zu verwenden.[9] In dem auf ATNs (*Augmented Transition Networks*) basierenden System von [Weischedel 1978] wird kein Präferenzmechanismus benötigt, da Konfliktsituationen mit konkurrierenden Regeln vermieden werden:

> „*A path is taken if and only if the particular incorrect form is present.*“ [Weischedel 1978, S.231]

Fehlerfälle werden durch zusätzliche Kanten und Zustände im ATN modelliert. Die Systeme von [Schwind 1986] und [Chen 1989] sind in Prolog bzw. in dem in Prolog enthaltenen Formalismus DCG (*Definite Clause Grammar*) realisiert. Die Präferenz von normalen Regeln vor Fehlerregeln wird durch den Auswertungsmechanismus von Prolog garantiert. Als Beispiel betrachtet Schwind (1986) die Position eines Adjektivs in einer Substantivgruppe. Die Zielsprache ist Deutsch, während als Muttersprache Französisch angenommen wird. Da im Französischen das Adjektiv hinter dem Substantiv in einer Nominalgruppe steht, wird eine Fehlerregel verwendet, die diese Anordnung berücksichtigt:

```
np(X,X0) :- art(X,X1), ng(X1,X0,F), F.
ng(X,X0,correct) :- ag(X,X1),noun(X1,X0).
ng(X,X0,error) :- noun(X,X1), ag(X1,X0).
```

Diese vereinfachte Darstellung wird im Fehlerfall das Prädikat `error` auswerten, welches dann eine Fehlererklärung generieren kann.

2.3.2 Modifikation der Regeln

Unter dieser Technik werden Verfahren verstanden, welche die normalen Regeln einer Grammatik so manipulieren, daß eine Verarbeitung der ungrammatischen

[9] Das Ziel sollte darin bestehen, diejenigen Fehlerregeln in einer Ableitung zu verwenden, die eine verständliche und angemessene Erklärung ermöglichen. Das Kriterium der Kardinalität muß nicht hinreichend sein.

Sätze möglich wird. Dazu wird immer eine vorgeschaltete Analyse mit der korrekten Grammatik benötigt. Erst wenn keine Strukturbeschreibung generiert werden kann, werden Regeln der Grammatik modifiziert.

Das in [Weischedel 1983] vorgestellte Verfahren verwendet sogenannte Metaregeln (*Meta-Rules*). Die Metaregeln werden angewendet, wenn eine Blockade der Analyse stattfindet. Sie ordnen einem Systemzustand des Parsers eine Aktion zu, die eine Modifikation bewirkt und sind von der allgemeinen Form

$$< Konfiguration > \rightarrow < Aktion > .$$

In der Arbeit von Weischedel et al. wird diese Technik anhand des ATN-Formalismus demonstriert.

In [Uszkoreit 1991] wird eine Technik für unifikationsbasierte Formalismen vorgestellt. Das Prinzip besteht darin, für jede Bedingung der Grammatik eine Funktion anzugeben, die im Falle der Nichterfüllbarkeit statt der ersteren verwendet wird. Dabei werden verschiedene Grade der Nichtwohlgeformtheit unterstützt. Die Funktion hat die Form $r : C \times L \rightarrow C'$, wobei C die Menge der Bedingungen der korrekten Grammatik, L die Menge der Abschwächungsebenen und C' die Menge der modifizierten Bedingungen darstellt. Die Anwendung der Regeln wird so geordnet, daß nur im Falle einer Analyseblockade die modifizierten Regeln angewendet werden. Dadurch wird eine Grammatik G durch eine geringfügig modifizierte Variante G' mit $L(G) \subseteq L(G')$ ersetzt. Ein Vorteil besteht darin, daß Zwischenergebnisse der vorherigen Analysen beibehalten werden können.

2.3.3 Bedingungsrücknahme

Die Technik der Bedingungsrücknahme ist dadurch gekennzeichnet, daß bestimmte Bedingungen der Grammatik einfach ignoriert werden. Dies ist nur für konjunktiv verknüpfte, d.h. einschränkende Bedingungen möglich.

Eine erste Form der Bedingungsrücknahme wird schon im System von [Weischedel 1978] verwendet. Die Grammatik, repräsentiert als ATN, verwendet einschränkende Bedingungen an den Kanten. Das System erlaubt es, diese Bedingungen zu ignorieren und trotzdem in den Folgezustand zu wechseln.

In den Systemen von [Schwind 1988], [Schwind 1995] und [Hagen 1994] wird die Bedingungsrücknahme auf Unifikationsgleichungen bezogen. Hagen behält im Falle einer fehlgeschlagenen Unifikation die Merkmale des lexikalischen Kopfes (*major part of speech*) und merkt sich das widersprüchliche Merkmal für Erklärungszwecke. Schwind et al. geben eine Unifikationssemantik für den Fehlerfall an, d.h. eine Wertaktualisierung auch für inkompatible Merkmalsstrukturen. Allerdings scheint es so, als wären nur atomare Werte in der Merkmalsstruktur erlaubt. Dies wäre eine erhebliche Einschränkung in der Modellierungsmächtigkeit. Schwind et al. verwenden zusätzlich noch Fehlerregeln, während die Grammatikmodellierung von Hagen scheinbar vollständig prozedural erfolgt.

Eine spezialisierte Diagnoseprozedur, die vergleichbar ist mit einer Bedingungsrücknahme von Unifikationsgleichungen ohne Merkmalsstrukturen mit komplexen Werten, wird in [Menzel 1988] vorgestellt. Im Unterschied zu den Verfahren von Schwind et al. können auch Faktenfehler, d.h. fehlerhaftes lexikalisches

Wissen, diagnostiziert werden. Daraus lassen sich dann Korrekturvorschläge generieren. Insbesondere weist das Verfahren eine hohe Diagnosequalität auf. Dieses Verfahren wird in Kapitel 3 beschrieben. Die Verarbeitung von freien Äußerungen, wie es in Schwind et al. durch die Verwendung von Fehlerregeln möglich ist, kann aber nicht geleistet werden.

Das Verfahren aus [Menzel 1998e] kann die Technik der Bedingungsrücknahme für die Verarbeitung von freien Äußerungen verwenden. Ein Formalismus, der eine Struktur allein aufgrund von Bedingungsverletzungen ermitteln kann, benötigt eine vollständige Repräsentation aller möglichen Strukturbeschreibungen. Das Verfahren wird in Kapitel 4 behandelt. In dem Formalismus können sowohl normale Regeln als auch Fehlerregeln modelliert werden. In einigen Fällen, z.B. für die Berücksichtigung von Wortartenvertauschungen oder unerwarteten Unterordnungen, ist eine Verwendung von Fehlerregeln durchaus sinnvoll. In [Mitrovic 1999] wird ein auf konjunktiv verknüpften Bedingungen basierender SQL-Tutor vorgestellt, der u.a. syntaktische Fehler in SQL-Anfragen identifizieren kann. Dabei werden die Fehlerklassen wie in [Menzel 1998e] durch die Verletzung von normalen Regeln definiert.

2.3.4 Modifikation des Inferenzmechanismus

Neben den beiden Möglichkeiten, zusätzliche disjunktive Elemente einzuführen (Fehlerregeln) bzw. konjunktive Elemente zu entfernen (Bedingungsrücknahme), gibt es die Möglichkeit, den Inferenzmechanismus anzupassen. Dazu werden die Ergebnisse einer fehlgeschlagenen Analyse mit der korrekten Grammatik benötigt. Anstatt Bedingungen zu ersetzen oder Fehlerregeln zu betrachten wird die Interpretation der Regelanwendung verändert. Ein Verfahren, welches auf diesem Prinzip beruht, geht auf [Mellish 1989] zurück. Er verwendet einen Bottom-Up Chartparser (vgl. z.B. [Allen 1995]) für die Analyse einer fehlerhaften Äußerung. Die Teilergebnisse der Analyse werden dann von einem modifizierten Top-Down Chartparser verwendet. Die Kombinationsregeln des Parsers werden dabei für spezifische Fehlerklassen angepaßt, insbesondere gibt es spezielle Chartregeln für fehlende Wörter (*Empty category rule*), für überflüssige Wörter (*Garbage rule*) und für unbekannte Wörter (*Unknown word rule*). Die Ergebnisse liegen nur für einzelne Fehler ohne Berücksichtigung von morphologischen Fehlern vor. Mellish erwartet eine drastische Verschlechterung der Verarbeitungszeiten im Falle von Mehrfachfehlern. Ein Vorteil des Prinzips liegt darin, daß keine Antizipation von Fehlern in der Grammatik vorgenommen werden muß. Die unterstützten Fehlerklassen werden durch die Verwendung spezieller Inferenzmechanismen antizipiert.

Kapitel 3

Fehlersimulation

Das Prinzip, das Verhalten eines Systems aus dem Verhalten seiner Teile zu folgern, bildet die Grundlage für das in diesem Kapitel behandelte Verfahren der *Fehlersimulation*. Die verwendete Diagnosetechnik wird *modellbasierte Diagnose* genannt. Diese Technik, die als formale Beschreibung die Prädikatenlogik verwendet, wird in Abschnitt 3.1 erläutert. Ein System besteht dabei aus einer Menge von Systemkomponenten, den Verbindungen zwischen den Komponenten und der Beschreibung des Verhaltens der Komponenten. Der Begriff der Fehlersimulation resultiert daher, daß das Verhalten der Systemkomponenten in Abhängigkeit einer Korrektheitsannahme simuliert wird, d.h. daß Werte an den Ein- und Ausgängen der Komponenten solange verändert werden, bis ein Gleichgewichtszustand erreicht ist, welcher die Systembeobachtung rekonstruiert.

In dieser Arbeit wird unter dem Begriff Fehlersimulation ein spezielles Verfahren verstanden, welches die Technik der modellbasierten Diagnose auf den Anwendungsbereich der Diagnose von grammatikalischen Fehlern in Fremdsprachenlernsystemen überträgt. Das Ziel ist es, die hohe Präzision und Qualität der modellbasierten Diagnose in diesem Anwendungszusammenhang zu nutzen.

Die modellbasierte Diagnose geht von einer statischen Strukturbeschreibung aus, d.h. es müssen sowohl die Systemkomponenten als auch die Verbindungen zwischen den Komponenten bekannt sein. Diese Voraussetzung bedeutet für den Anwendungsbereich Sprachlernsysteme, daß nur ein Teilbereich der linguistischen Phänomene auf der Ebene der Syntax modelliert werden kann. Die Modellierung dieses Teilbereichs ist Gegenstand des Abschnittes 3.2. Die Phänomene beschränken sich dabei weitgehend auf die Modellierung von Kongruenzbeziehungen.

Die Darstellung eines Grammatikmodells als ein System im Sinne der modellbasierten Diagnose wird in Abschnitt 3.3 behandelt. Anhand von Beispielen wird die Leistungsfähigkeit des Verfahrens verdeutlicht.

3.1 Grundlagen der modellbasierten Diagnose

Die Technik der modellbasierten Diagnose basiert auf der Annahme, daß sich das Verhalten eines Systems aus dem Verhalten seiner Teile folgern läßt. Liefert ein System nicht die Ergebnisse, die aufgrund des spezifizierten Verhaltens seiner Komponenten zu erwarten wären, liegt es nahe, für die Ursache des Fehlverhaltens einzelne Komponenten verantwortlich zu machen. Das diagnostische Prinzip besteht daher darin, Annahmen über die Korrektheit einzelner Komponenten aufzustellen und diese dann an dem modifizierten System zu überprüfen. Die im folgenden verwendete Terminologie basiert auf den Arbeiten von de Kleer und Reiter, in denen der System- und Diagnosebegriff formal charakterisiert wird ([de Kleer 1992], [Reiter 1987]).

3.1.1 Systembeschreibung

Definition 3.1.1 (System) *Ein System ist ein Tripel* (*SD*, *COMPS*, *OBS*) *mit*

1. *SD, einer Systembeschreibung, eine Menge von Sätzen der Prädikatenlogik erster Ordnung;*

2. *COMPS, den Systemkomponenten, eine endliche Menge von Konstanten;*

3. *OBS, einer Systembeobachtung, eine Menge von Sätzen der Prädikatenlogik erster Ordnung.*

Diese Definition führt den Begriff der Systemkomponenten ein. Die Systemkomponenten werden miteinander über gemeinsame Übertragungskanäle, den sogenannten *Ports*, verbunden. Diese Ports enthalten die Ein- und Ausgangswerte der Komponenten ([Davis 1984]). Die Systembeschreibung enthält in vielen Anwendungsfällen folgende Information:

- Typbeschreibung der verwendeten Komponenten;
- Beschreibung der Verbindungsstruktur zwischen den Komponenten;
- Verhaltensbeschreibung der Komponententypen;
- Wertebereich der Ports der Komponenten.

Weiterhin werden häufig noch Formeln benötigt, die grundlegende Operationen und Relationen definieren, z.B. um Ausdrücke zu vergleichen oder auszuwerten. Die Verhaltensbeschreibung kann einerseits nur das korrekte Verhalten, andererseits das korrekte und das fehlerhafte Verhalten der Komponenten spezifizieren. Die Beschreibung für das korrekte Verhalten wird als *Normalfallsemantik* bezeichnet. Falls das Verhalten im Fehlerfall bekannt ist, wird für die Verhaltensbeschreibung der Begriff der *Fehlerfallsemantik* verwendet. [1]

Die grafische Darstellung eines Systems wird als Komponentennetz bezeichnet. In Abbildung 3.1 ist das klassische Beispiel aus [Davis 1984] als Komponentennetz dargestellt.

[1]Eine Einteilung in Normal- und Fehlerfallsemantik ist nur dann möglich, wenn sich die Beobachtungen im Fehlerfall von denen im Normalfall unterscheiden. Zum Beispiel kann eine stehengebliebene Uhr zweimal am Tag die korrekte Uhrzeit anzeigen.

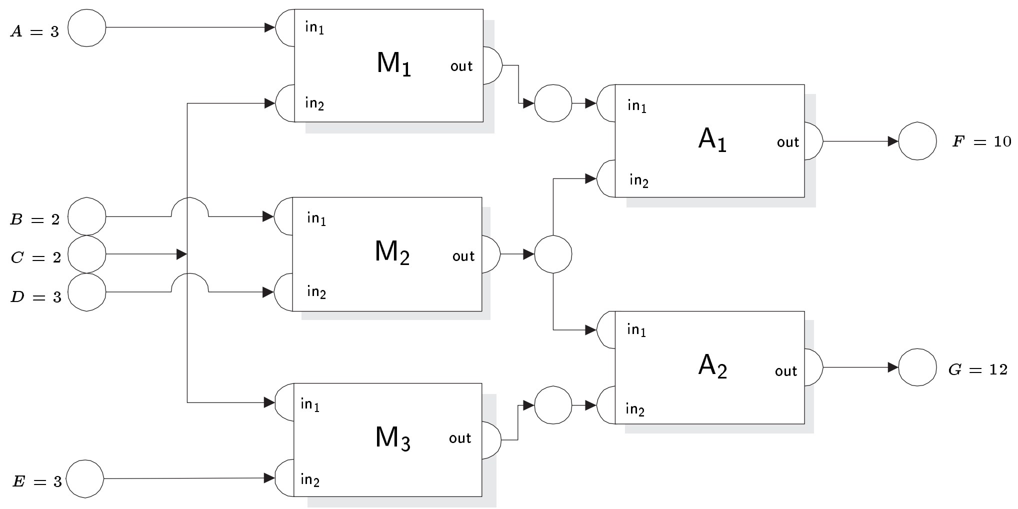

Abbildung 3.1: Komponentennetz für die Berechnung von $F = AC + BD$ und $G = CE + BD$. Die Komponenten M_1 bis M_3 sind Multiplizierer, A_1 und A_2 sind Addierer.

3.1.2 Diagnose

Unter einer Diagnose für ein System wird eine Menge von Systemkomponenten verstanden, für die sich aus der Annahme eines fehlerhaften Verhaltens dieser Komponenten und eines korrekten Verhaltens aller anderen Komponenten die Systembeobachtung ableiten läßt. Die Annahme über das fehlerhafte Verhalten einer Komponente c wird mit der Formel $\mathsf{ab}(c)$ ausgedrückt.[2]

Definition 3.1.2 (Diagnose) *Eine Diagnose für ein System* $(SD, COMPS, OBS)$ *ist eine Menge* $D \subseteq COMPS$ *mit*

$$OBS \cup \{\mathsf{ab}(c) | c \in D\} \cup \{\neg\mathsf{ab}(c) | c \in COMPS - D\} \quad \textit{ist konsistent.}$$

Für ein System kann es sehr viele Diagnosen geben, da potentiell jede Teilmenge aus der Menge aller Komponenten eine Diagnose sein kann. Für eine Menge mit n Elementen gibt es bekanntlich $\sum_{k=0}^{n} \binom{n}{k} = 2^n$ mögliche Teilmengen. Daher werden in der Regel nur minimale Diagnosen betrachtet.

Definition 3.1.3 (Minimale Diagnose) *Eine minimale Diagnose für ein System* $(SD, COMPS, OBS)$ *ist eine Diagnose* D*, für die folgendes gilt:*

$$\forall D' : D' \quad \textit{ist Diagnose für} \quad (SD, COMPS, OBS) \land D' \subseteq D \Rightarrow D' = D.$$

Nur unter bestimmten Voraussetzungen sind minimale Diagnosen charakteristisch für alle möglichen Diagnosen, d.h die Hypothese, wenn D eine minimale Diagnose ist, dann ist auch jede Menge D' mit $D \subset D'$ eine Diagnose, gilt nicht

[2] Die Bezeichnung ab steht für das Wort *abnormal*. $\neg\mathsf{ab}$ kennzeichnet Korrektheit.

immer ([de Kleer 1992]). Das System aus Abbildung 3.1 enthält vier minimale Diagnosen:

$$D_1 = \{\mathsf{A}_1\}, D_2 = \{\mathsf{M}_1\}, D_3 = \{\mathsf{M}_2, \mathsf{M}_3\}, D_4 = \{\mathsf{A}_2, \mathsf{M}_2\}.$$

Für die Berechnung einer Diagnose eines Systems sind folgende Schritte zu unterscheiden:

1. Generierung von Hypothesen, d.h. Einteilung in korrekte und fehlerhafte Komponenten.

2. Überprüfung von Hypothesen auf Konsistenz. Dabei wird meistens von einer „geschlossenen Welt" ausgegangen ([Genesereth 1987],*closed world assumption*), d.h. wenn eine Formel p nicht bewiesen werden kann, gilt $\neg p$. Viele konsistenzüberprüfende Verfahren, wie z.B. das Resolutionsverfahren, basieren auf dieser Annahme.

Neben dem einfachen *generate and test*-Verfahren, in dem alle Hypothesen der Reihe nach generiert und überprüft werden, wurde von [Reiter 1987] ein Verfahren vorgestellt, welches eine effiziente, nicht redundante Hypothesengenerierung ermöglicht. Dies gelingt durch einen Buchhaltungsmechanismus, welcher sowohl die vom Beweiser verwendeten Formeln als auch die bereits aufgestellten Hypothesen verwaltet. Die logische Trennung von Konsistenzüberprüfung und Hypothesengenerierung wird auch in *Assumption Based Truth Maintainance Systems* (ATMS) verwendet, die von de Kleer erstmals vorgestellt wurden ([de Kleer 1986]). Für das Beispiel aus Abbildung 3.1 verwendet [Davis 1984] ein Simulationsverfahren, in dem das Verhalten der Komponenten die Wertaktualisierung an den Ports bestimmt. Sind weitere Komponenten von einer Wertänderung abhängig, ist eine erneuerte Simulation der abhängigen Komponenten notwendig. Die Propagation von Werten und die damit verbundene Reaktivierung von Komponenten wird als *Constraint Propagation* bezeichnet.

3.2 Modellbildung

Es stellt sich die Frage, ob es möglich ist, einen Grammatikformalismus zu entwickeln, welcher die Technik der modellbasierten Diagnose für die Analyse natürlichsprachlicher Äußerungen verwendet. Die meisten Grammatikformalismen unterteilen die Wohlgeformtheitsbedingungen in zwei grundlegende Arten von Modellinformation:

1. Strukturbildende Mechanismen, die ausgehend von einer Klassifizierung von Wortformen eine hierarchische Gliederung und meistens auch die lineare Anordnung der Satzbestandteile beschreiben (z.B. kontextfreie Phrasenstrukturgrammatiken [Gazdar 1985] oder Dependenzgrammatiken [Kunze 1975]).

2. Zusätzliche Bedingungen für die Wohlgeformtheit eines Satzes, die aufgrund grammatikalischer Merkmale formuliert werden (z.B. die Unifikationsgleichungen in unifikationsbasierten Grammatiken [Shieber 1987]).

Die Voraussetzung für die modellbasierte Diagnose ist eine Kenntnis der Systemstruktur, d.h. der Komponenten und der Verbindungen zwischen ihnen. Die Wohlgeformtheitsbedingungen des Grammatikformalismus müssen also als Systemkomponenten darstellbar sein, wobei die Beziehungen zwischen ihnen über die Verwendung gemeinsamer Ports ausgedrückt werden können. Dies gestaltet sich problemlos für merkmalsbasierte Wohlgeformtheitsbedingungen, wenn die Struktur der Äußerung vorgegeben werden kann. Ist z.B. in dem Satz

(1) „Die gelbe Vase steht auf dem Tisch."

bekannt, daß „die gelbe Vase" eine Substantivgruppe konstituiert und „Vase" das Subjekt, „die" und „gelbe" ein das Subjekt modifizierender Artikel bzw. Adjektiv sind, können Regeln formuliert werden, die z.B. die Kasus-, Numerus- und Genuskongruenz der beteiligten Wortformen überprüfen.

Im unifikationsbasierten Formalismus `PATRII` können Kongruenzbeziehungen durch Unifikationsgleichungen angegeben werden. Die Substantivgruppe könnte z.B. durch folgende Regel analysiert worden sein:

$$
\begin{array}{l}
\mathsf{NP} \quad \rightarrow \quad \mathsf{DET} \quad \mathsf{ADJ} \quad \mathsf{N} \\
\langle \mathsf{DET\ case} \rangle = \langle \mathsf{N\ case} \rangle \\
\langle \mathsf{DET\ num} \rangle = \langle \mathsf{N\ num} \rangle \\
\langle \mathsf{DET\ gender} \rangle = \langle \mathsf{N\ gender} \rangle \\
\langle \mathsf{ADJ\ case} \rangle = \langle \mathsf{N\ case} \rangle \\
\langle \mathsf{ADJ\ num} \rangle = \langle \mathsf{N\ num} \rangle \\
\langle \mathsf{ADJ\ gender} \rangle = \langle \mathsf{N\ gender} \rangle
\end{array}
$$

Für jede dieser Gleichungen kann eine Komponente im System eingeführt werden, deren Werte an den Ein- und Ausgangsports sich aus den Merkmalsstrukturen der verwendeten Wortformen ergeben. Damit kann die Systembeobachtung direkt aus der Äußerung konstruiert werden.

Problematisch sind die strukturbildenden Mechanismen. Die strukturellen Beziehungen, insbesondere die Dominanz und die lineare Anordnung, könnten im Prinzip ebenfalls als Systemkomponenten agieren. Die Anordnungsforderungen sind unkritisch, da die Systembeobachtung der aktuellen Position der verwendeten Wortform in der Äußerung entspricht. Allerdings scheint es nicht möglich, für die Dominanzforderung eine Systembeobachtung zur Verfügung zu stellen, da die lineare Folge von Wortformen keinen Hinweis dafür liefert. Hier ist ein Strukturerstellungsprozeß, also ein Parser, notwendig, um überhaupt eine Wertebelegung für die Ports im System zu finden. Die Dominanzforderungen können nur unter stark eingeschränkten Anwendungsbedingungen verwendet werden, wobei alle Wortformen von genau einer Konstituente dominiert werden ([Menzel 1990]).

Im folgenden wird daher die Modellierung von merkmalsbasierten Wohlgeformtheitsbedingungen diskutiert. Ausgehend von der im Verfahren der Fehlersimulation verwendeten Repräsentation lexikalischen Wissens werden morphosyntaktische Wohlgeformtheitsbedingungen aufgestellt. Da die Struktur einer

Äußerung für das Verfahren der Fehlersimulation als bekannt vorausgesetzt werden muß, sind die bisherigen Anwendungssysteme für den Sprachunterricht auf Lückentextübungen beschränkt gewesen.

3.2.1 Lexikalisches Wissen

Unter einer Äußerung ω wollen wir eine Folge von Wortformen über einem Alphabet Σ verstehen, d.h. $\omega \in \Sigma^*$. Die Symbole aus Σ bezeichnen Wortformen als kleinste atomare Einheit. Jeder Wortform wird eine Informationsstruktur zugeordnet, welche die für den Anwendungsbereich relevante grammatikalische Information enthält. Als Repräsentationform werden in der Sprachverarbeitung häufig Merkmalsstrukturen verwendet, die im wesentlichen eine Ansammlung von Paaren aus Eigenschaften (Merkmalen) und Werten sind. Formal kann diese Struktur als partielle Funktion aufgefaßt werden: [3]

Definition 3.2.1 (Merkmalsstruktur) *Sei M eine endliche Menge von Merkmalen. Weiterhin sei Q eine endliche Menge von atomaren Werten. Wir definieren eine Merkmalsstruktur als eine partielle Funktion f, die einem Merkmal aus M eine Menge von Werten aus Q zuordnet:*

$$f : M \rightarrow 2^Q.$$

Mit $\mathrm{dom}(f) \subseteq M$ *wird der Definitionsbereich der Merkmalsstruktur f bezeichnet. Die Menge aller Merkmalsstrukturen bezüglich M und Q wird mit $F_{M,Q}$ bezeichnet, im folgenden verkürzt als F.*

Eine Merkmalsstruktur $f \in F$, die z.B. dem Merkmal cat den Wert $\{\mathsf{det}\}$ und dem Merkmal case den Wert $\{\mathsf{acc}, \mathsf{nom}\}$ zuordnet, notieren wir als

$$[\mathsf{cat} :\{\mathsf{det}\} \;\; \mathsf{case} :\{\mathsf{acc}, \mathsf{nom}\}].$$

In dieser Arbeit wollen wir unter dem Begriff eines Lexikons die Zuordnung von Wortformen aus Σ auf eine Menge von Merkmalsstrukturen verstehen. Die formale Definition definiert ein Lexikon als eine Relation zwischen Wortformen und Merkmalsstrukturen.

Definition 3.2.2 (Lexikon) *Sei Σ eine endliche Menge von Wortformen. Weiterhin sei F die Menge aller Merkmalsstrukturen. Unter einem Lexikon $L_{\Sigma,Q}$, kurz L, verstehen wir eine Teilmenge aus dem kartesischen Produkt $\Sigma \times F$, wobei für jede Wortform mindestens eine Merkmalsstruktur existieren muß:*

$$L \subseteq \Sigma \times F \quad \wedge \quad \forall w \in \Sigma : \exists f \in F : (w, f) \in L.$$

Ein Lexikoneintrag $l \in L$, der z.B. der Wortform „der" eine Menge von Merkmalsstrukturen zuordnet, wird notiert als

$$\begin{array}{lll} \mathsf{der} & : & [\mathsf{cat} :\{\mathsf{det}\} \;\; \mathsf{num} :\{\mathsf{sing}\} \;\; \mathsf{gender} :\{\mathsf{masc}\} \;\; \mathsf{case} :\{\mathsf{nom}\}], \\ & & [\mathsf{cat} :\{\mathsf{det}\} \;\; \mathsf{num} :\{\mathsf{sing}\} \;\; \mathsf{gender} :\{\mathsf{fem}\} \;\; \mathsf{case} :\{\mathsf{gen}, \mathsf{dat}\}], \\ & & [\mathsf{cat} :\{\mathsf{det}\} \;\; \mathsf{num} :\{\mathsf{plur}\} \;\; \mathsf{gender} :\{\mathsf{masc}, \mathsf{fem}, \mathsf{neutr}\} \;\; \mathsf{case} :\{\mathsf{gen}\}]. \end{array}$$

[3]Diese Definition berücksichtigt keine komplexen Werte eines Merkmals, die insbesondere in unifikationsbasierten Verfahren von zentraler Bedeutung sind ([Shieber 1987]).

In dieser Formulierung läßt sich der Begriff der lexikalischen Mehrdeutigkeit exakt fassen:

Definition 3.2.3 (Mehrdeutigkeit) *Eine Wortform $w \in \Sigma$ heißt mehrdeutig, falls*

$$|\{f|(w,f) \in L\}| > 1 \quad \vee \quad \exists (w,f) \in L : \exists m \in \mathrm{dom}(f) : |f(m)| > 1$$

ist, ansonsten heißt sie eindeutig.

Eine Lesart einer Äußerung $\omega \in \Sigma^*$ wird formal als eine Zuordnung von den Wortformen der Äußerung auf eine Merkmalsstruktur verstanden.

Definition 3.2.4 (Lesart) *Sei $\omega = w_1 \ldots w_n \in \Sigma^*$ eine Äußerung. Eine Lesart r von ω bezüglich eines Lexikons L sei folgende Funktion:*

$$r : \Sigma^* \to F^* \quad \mathit{mit} \quad r(w_1 \ldots w_n) = f_1 \ldots f_n \wedge \forall 1 \leq i \leq n : (w_i, f_i) \in L.$$

Die Menge aller Lesarten sei $R(\omega)$. Es gilt:

$$|R(w_1 \ldots w_n)| = \prod_{1 \leq i \leq n} |\{f|(w_i, f) \in L\}|.$$

Die Anzahl aller Lesarten einer Äußerung wächst damit exponentiell in der Länge der Äußerung, allerdings geht nur das erste Kriterium der lexikalischen Mehrdeutigkeit in die Berechnung mit ein. Ein Verzicht auf eine Mengenrepräsentation der Werte eines Merkmals würde die Menge der Lesarten deutlich vergrößern. Die Wortform „der“ aus obigem Beispiel hätte statt drei dann sechs Lesarten. Neben der Reduktion von Lesarten bietet die Mengenrepräsentation insbesondere im Verfahren der Fehlersimulation noch den Vorteil, daß bestimmte Wohlgeformtheitsbedingungen effizient berechnet werden können, da der endliche Wertebereich der Merkmale in der Regel wenige Elemente beinhaltet. Der Wertebereich eines Merkmals läßt sich direkt aus den Lexikoneinträgen eines Lexikons bestimmen. Die folgende Funktion bestimmt für ein Merkmal und ein Lexikon den Wertebereich des Merkmals:

$$\begin{aligned} &\mathrm{domain} : M \times 2^{\Sigma \times F} \to 2^Q \quad \mathrm{mit} \\ &\mathrm{domain}(m, L) = \bigcup_{(w,f) \in L \wedge m \in \mathrm{dom}(f)} f(m). \end{aligned}$$

Wir verwenden folgende Notation, um z.B. die Werte der Wortart cat festzulegen:

$$\mathsf{cat} : \{\mathsf{det}, \mathsf{adj}, \mathsf{n}\}.$$

3.2.2 Morpho-syntaktisches Wissen

Im folgenden werden die im Rahmen der Fehlersimulation möglichen Wohlgeformtheitsbedingungen eingeführt, deren Formulierung wesentlich auf der Mengenrepräsentation der Wertebereiche von Merkmalen basiert.

Modellvariablen und Wertebindung

Bestimmte merkmalsbasierte Wohlgeformtheitsbedingungen erlauben eine Darstellung als Komponente im Sinne der modellbasierten Diagnose. Zu diesen Bedingungen zählen Kongruenzforderungen, z.B. die Numerus-, Kasus- und Genuskongruenz zwischen Artikel, Adjektiv und Substantiv in Nominalgruppen. Für die Formulierung dieser Bedingungen werden Symbole eingeführt, denen Wortformen des Eingabesatzes zugeordnet werden. Diese Symbole werden im folgenden als Modellvariablen bezeichnet. Ursprünglich wurde das Verfahren der Fehlersimulation in Lückentextübungen eingesetzt, in denen bestimmte Teile eines unvollständigen Satzes mit Wortformen vom Schüler auszufüllen waren ([Stockfleth 1999]). In der Übung

(U1) „Das Kind sucht DET ADJ N."

wird ein Artikel, ein Adjektiv und ein Substantiv vom Schüler gefordert. Die Symbole DET, ADJ und N sind die Modellvariablen der Übung. Eine Wertebindung ordnet jeder Modellvariablen die vom Schüler eingegebene Wortform zu. Dazu reicht es aus, jeder Modellvariablen einen Positionsindex zuzuordnen.

Definition 3.2.5 (Wertebindung) *Sei $\omega = w_1 \dots w_n \in \Sigma^*$ eine Äußerung. Sei V eine Menge von Modellvariablen mit $|V| = n$. Eine Wertebindung b von V für eine Äußerung ω ist eine bijektive Funktion, die jeder Modellvariablen einen Positionsindex zuordnet:*

$$b : V \to \{1, \dots, n\}$$

Eine Lesart einer Menge von Modellvariablen erweitert die Definition 3.2.4. Für eine Lesart r für eine Äußerung ω bezeichnet $r^{(i)}$ das i-te Element der Folge von $r(\omega)$.

Definition 3.2.6 (Lesart von Modellvariablen) *Sei V eine Menge von Modellvariablen. Sei r eine Lesart einer Äußerung ω bezüglich eines Lexikons L. Sei eine Wertebindung b von V für ω gegeben. Eine Lesart r' von V ist dann folgende Funktion:*

$$r' : V \to F \quad mit \quad \forall v \in V : r'(v) = f \Leftrightarrow r^{(b(v))} = f$$

Die Lesarten der Modellvariablen werden benötigt, um verschiedene Systembeobachtungen charakterisieren zu können (vgl. Abschnitt 3.3.1).

Merkmalsbasierte Wohlgeformtheitsbedingungen

Die Wortform jeder gebundenen Modellvariablen liefert über das Lexikon eine Menge von Merkmalsstrukturen. Aufgrund der lexikalischen Repräsentation aus Abschnitt 3.2.1 sind die Werte eines Merkmals immer Mengen von atomaren Symbolen. Wohlgeformtheitsbedingungen sind dann auf dieser Information basierende Ausdrücke, analog zu den Unifikationsgleichungen aus Abschnitt 3.2. Insbesondere sind folgende Typen von Bedingungen im Verfahren der Fehlersimulation von Bedeutung:

- Bedingungen, welche den Wert eines Merkmals einer Modellvariablen mit einem Vorgabewert, z.B. aufgrund der kontextuellen Einbettung der Übung, vergleichen. Gibt es gemeinsame Elemente im Wert des Merkmals und dem Vorgabewert, so ist die Bedingung erfüllt. Diese Bedingungen notieren wir als Prädikate der Form

  $$\mathsf{restrict}(\mathit{Modellvariable}, \mathit{Merkmal}, \mathit{Vorgabewert}).$$

- Bedingungen, welche den Wert von Merkmalen zweier Modellvariablen auf gemeinsame Elemente überprüfen. Gibt es gemeinsame Elemente der Werte der Merkmale der Modellvariablen, so ist die Bedingung erfüllt. Wir verwenden dafür folgende Notation:

  $$\mathsf{agree}(\mathit{Modellvariable}_1, \mathit{Modellvariable}_2, \mathit{Merkmal}_1, \mathit{Merkmal}_2).$$

  Sind beide Merkmale identisch, wird eine verkürzte Darstellung verwendet:

  $$\mathsf{agree}(\mathit{Modellvariable}_1, \mathit{Modellvariable}_2, \mathit{Merkmal}).$$

Das agree-Prädikat wird als transitiv und symmetrisch angenommen, d.h.

$$\mathsf{agree}(v_1, v_2, m_1, m_2) \wedge \mathsf{agree}(v_2, v_3, m_2, m_3) \Rightarrow \mathsf{agree}(v_1, v_3, m_1, m_3)$$
$$\mathsf{agree}(v_1, v_2, m_1, m_2) \Rightarrow \mathsf{agree}(v_2, v_1, m_2, m_1).$$

Fehlersimulationsmodell

Ein Grammatikmodell, welches die im vorherigen Abschnitt definierten Prädikate verwendet, wollen wir als *Fehlersimulationsmodell* bezeichnen.

Definition 3.2.7 (Fehlersimulationsmodell) *Ein Fehlersimulationsmodell ist ein Tripel (L, V, P) mit einem Lexikon L, einer endlichen Menge von Modellvariablen V und einer Menge von Wohlgeformtheitsbedingungen P, wobei jedes Element aus P folgende Form haben kann:*

$$p \in P \Rightarrow p = \mathsf{restrict}(v, m, s) \vee p = \mathsf{agree}(v_1, v_2, m_1, m_2)$$
$$v, v_1, v_2 \in V, m, m_1, m_2 \in M, s \subseteq \mathrm{domain}(m, L).$$

Die Menge M bezeichnet die Menge der Merkmale des Lexikons (siehe Definition 3.2.2).

Im folgenden wollen wir entsprechende Wohlgeformtheitsbedingungen für die Übung (U1) entwickeln. Die verwendeten Merkmale und Wertebereiche der Merkmale sind in folgender Tabelle zusammengefaßt.

Merkmal	Wertebereich
cat	{det, adj, n}
case	{nom, gen, acc, dat}
num	{sing, plur}
gender	{masc, fem, neutr}
atype	{weak, strong, mixed}
atype-sel	{weak, strong, mixed}

Das Merkmal cat definiert die unterstützten Wortarten: Artikel, Adjektive und Substantive. Die Werte der Kasus-, Numerus- und Genusmerkmale werden von den Merkmalen case, num und gender festgelegt. Die Beugung des Adjektivs ist vom Artikel abhängig. Dies kann modelliert werden, indem die Adjektive in verschiedene Klassen (Merkmal atype) eingeteilt werden. Wir verwenden dabei die klassische Einteilung in schwach, stark und gemischt gebeugte Adjektive ([Eisenberg 1998]). Das Merkmal atype−sel mit denselben Werten modelliert die Forderung des Artikels, welche Klasse das verwendete Adjektiv haben soll.

Aufbauend auf dieser lexikalischen Repräsentation können folgende Wohlgeformtheitsbedingungen der Übung (U1) formuliert werden:

- Die Wortformen der Modellvariablen DET, ADJ und N sollten ein Artikel, ein Adjektiv und ein Substantiv sein.

$$\mathsf{restrict}(\mathsf{DET}, \mathsf{cat}, \{\mathsf{det}\})$$
$$\mathsf{restrict}(\mathsf{ADJ}, \mathsf{cat}, \{\mathsf{adj}\})$$
$$\mathsf{restrict}(\mathsf{N}, \mathsf{cat}, \{\mathsf{n}\})$$

- Das Verb „suchen“ fordert eine Ergänzung im Akkusativ. Dies ist eine typische Kontextforderung, die von der verwendeten Übung abhängig ist.

$$\mathsf{restrict}(\mathsf{N}, \mathsf{case}, \{\mathsf{acc}\})$$

- Artikel, Adjektiv und Substantiv stimmen in Kasus, Numerus und Genus überein.[4]

$$\mathsf{agree}(\mathsf{DET}, \mathsf{N}, \mathsf{num})$$
$$\mathsf{agree}(\mathsf{DET}, \mathsf{N}, \mathsf{gender})$$
$$\mathsf{agree}(\mathsf{DET}, \mathsf{N}, \mathsf{case})$$
$$\mathsf{agree}(\mathsf{ADJ}, \mathsf{N}, \mathsf{num})$$
$$\mathsf{agree}(\mathsf{ADJ}, \mathsf{N}, \mathsf{gender})$$
$$\mathsf{agree}(\mathsf{ADJ}, \mathsf{N}, \mathsf{case})$$

- Der Artikel fordert ein passend gebeugtes Adjektiv.

$$\mathsf{agree}(\mathsf{DET}, \mathsf{ADJ}, \mathsf{atype\text{-}sel}, \mathsf{atype})$$

Die Abbildung 3.2 zeigt eine grafische Darstellung für das im Beispiel entwickelte Fehlersimulationsmodell.

[4] Die Transitivität des agree-Prädikates garantiert die Kasus-, Numerus- und Genuskongruenz zwischen Artikel und Adjektiv.

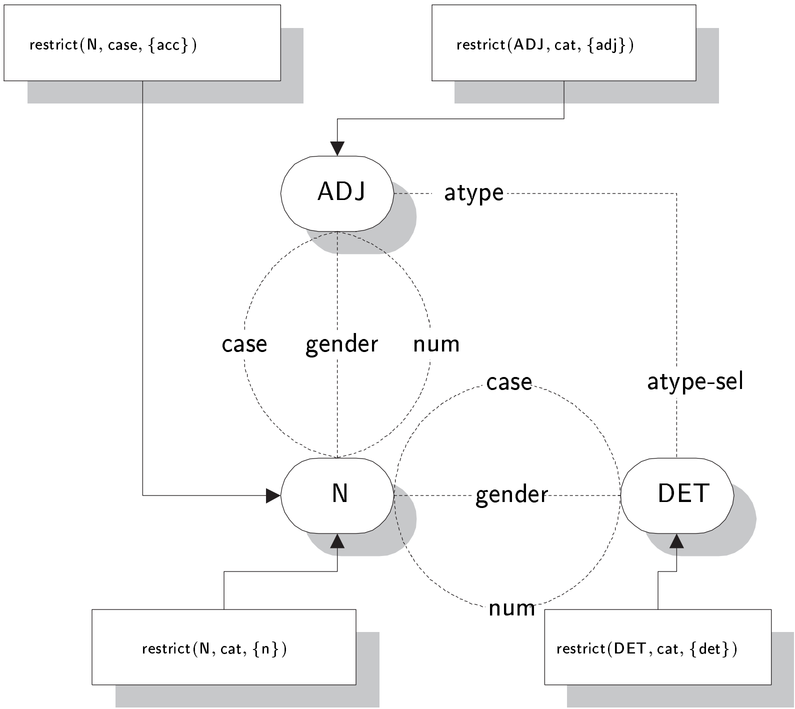

Abbildung 3.2: Fehlersimulationsmodell für eine Nominalphrase.

3.3 Fehlerdiagnose

Ein Fehlersimulationsmodell beschreibt deklarativ, welche Wohlgeformtheitsbedingungen zwischen Wortformen bestehen, für die eine statische Struktur vorgegeben wird. Im Beispiel wurde eine Struktur für eine Substantivgruppe angenommen. Dieser Abschnitt beschreibt, wie eine vorliegende Schüleräußerung für ein gegebenes Fehlersimulationsmodell verarbeitet werden kann und wie insbesondere Verletzungen von Wohlgeformtheitsbedingungen erkannt, lokalisiert und für Erklärungszwecke verwendet werden können. Hat der Schüler z.B. in der Übung (U1) den Satz

(2) * „Das Kind sucht *den gelbe Ball.*“

geäußert, so sollte eine Erklärung wie z.B.

(E1) „Das Adjektiv *gelbe* stimmt nicht mit dem Substantiv *Ball* im Kasus überein.“

einzig aus dem Modell gefolgert werden können. Wir werden sehen, daß das Verfahren alternative Erklärungen liefern kann. Unter Einbeziehung der Merkmalswerte ist folgende Erklärung möglich:

(E2) „*Gelbe* steht im Nominativ, aber *Ball* steht im Akkusativ. [5]“

[5] Das Substantiv „Ball“ kann sowohl Akkusativ als auch Nominativ sein. Daher wäre eine Erklärung, die sich auf den Kasus der Nominalgruppe bezieht, für (E1) und (E2) besser geeignet.

Unter der Annahme, daß der Schüler den korrekten Kasus des Adjektivs nicht kennt, kann ein Korrekturhinweis generiert werden:

(E3) „Korrigiere das Adjektiv *gelbe*! Es steht im Nominativ, muß aber im Akkusativ stehen.“

Weiterhin kann ein Korrekturvorschlag automatisch erzeugt werden:

(E4) „Der korrekte Satz lautet: Das Kind sucht den *gelben* Ball.“

Um dies zu erreichen, wird die Modellbeschreibung für die modellbasierte Diagnose aufbereitet, d.h. es wird ein System nach Definition 3.1.1 konstruiert. Insbesondere müssen die restrict- und agree-Prädikate mit einer formalen Semantik, sowohl im Normal- als auch im Fehlerfall, hinterlegt werden.

3.3.1 Systemkonstruktion

Eine Wertebindung nach Definition 3.2.5 ordnet den verwendeten Modellvariablen eine Wortform der Äußerung zu. Da aufgrund der lexikalischen Repräsentation eine Wortform mehrere Merkmalsstrukturen im Lexikon besitzen kann, existieren für eine Äußerung mehrere Lesarten (siehe Definition 3.2.4). Die folgende Darstellung kann als eine Konstruktionsvorschrift für die Erstellung eines Systems aus einem Fehlersimulationsmodell und einer Lesart einer Äußerung verstanden werden, d.h. für verschiedene Lesarten entstehen dadurch verschiedene Systeme. Der Gesamtprozeß muß dies berücksichtigen, indem für jede Lesart das entsprechend konstruierte System analysiert wird. Die minimalen Diagnosen der einzelnen Systeme müssen verglichen und eine Entscheidung zu Gunsten möglichst genau einer Lesart getroffen werden.

Ein Fehlersimulationsmodell $M = (L, V, P)$ enthält eine Menge von Prädikaten, die Wohlgeformtheitsbedingungen. Das zu konstruierende System sei $(SD, COMPS, OBS)$. Die Systembeschreibung SD enthält dabei die Beschreibung der Komponententypen SD_{type}, die Wertebereiche der Ports SD_{domain}, die Verbindungsstruktur SD_{connect}, die Verhaltensbeschreibung $SD_{\text{behaviour}}$ sowie weitere Formeln für die Auswertung von Ausdrücken auf Mengen, die hier nicht weiter angegeben werden.

Komponenten, Typen und Ports

Die Systemkomponenten ergeben sich direkt aus den verwendeten Prädikaten des Modells, d.h. $COMPS = P$. Es sind zwei verschiedene Komponententypen vorhanden.

Für die Typbeschreibung werden folgende Formeln verwendet:

$$\begin{aligned} SD_{\text{type}} \quad = \quad & \{\mathsf{type{-}agree}(x) | x \in COMPS \wedge x = \mathsf{agree}(v_1, v_2, m_1, m_2)\} \cup \\ & \{\mathsf{type{-}restrict}(x) | x \in COMPS \wedge x = \mathsf{restrict}(v, m, s)\} \end{aligned}$$

Für jede Komponente $x \in COMPS$ gibt es einen Eingangsport $\mathsf{in}(x)$ und einen Ausgangsport $\mathsf{out}(x)$. Die restrict- bzw. agree-Prädikate vergleichen die Werte von Merkmalen, die sich aus den Merkmalsstrukturen der Modellvariablen ergeben. Ein Port setzt sich somit aus einem Paar aus Modellvariable und Merkmal zusammen. Folgende Funktionen ordnen einer Komponente dieses Paar zu:

$$\begin{aligned}
&\mathrm{inport}(\mathsf{agree}(v_1, v_2, m_1, m_2)) = (v_1, m_1)\\
&\mathrm{outport}(\mathsf{agree}(v_1, v_2, m_1, m_2)) = (v_2, m_2)\\
&\mathrm{inport}(\mathsf{restrict}(v, m, s)) = \mathsf{unique}\\
&\mathrm{outport}(\mathsf{restrict}(v, m, s)) = (v, m).
\end{aligned}$$

Das restrict Prädikat enthält einen Vorgabewert, der durch den Kontext der Übung bestimmt ist und keine Entsprechung in der Schüleräußerung hat. Für jedes restrict-Prädikat wird für den Vorgabewert ein Eingangsport erzeugt, der von keiner anderen Komponente verwendet werden kann.

Wertebereiche

Der Wert eines Ports kann eine Teilmenge des Wertebereichs des Merkmals bezüglich eines Lexikons sein. Eine Sonderstellung nimmt dabei der Vorgabewert des restrict-Prädikates ein.

$$\begin{aligned}
SD_{\mathrm{domain}} \quad = \quad & \{\mathsf{in}(x) \subseteq \mathrm{domain}(L, m) | x \in COMPS \wedge \mathrm{inport}(x) = (v, m)\} \cup\\
& \{\mathsf{out}(x) \subseteq \mathrm{domain}(L, m) | x \in COMPS \wedge \mathrm{outport}(x) = (v, m)\} \cup\\
& \{\mathsf{in}(x) = s | x \in COMPS \wedge x = \mathsf{restrict}(v, m, s)\}
\end{aligned}$$

Verbindungsstruktur

Die Verbindungsstruktur ergibt sich direkt aus den gemeinsam verwendeten Ports der Komponenten (sei $x, x' \in COMPS$, $x \neq x'$):

$$\begin{aligned}
SD_{\mathrm{connect}} \quad = \quad & \{\mathsf{in}(x) = \mathsf{in}(x') | \mathrm{inport}(x) = \mathrm{inport}(x')\} \cup\\
& \{\mathsf{in}(x) = \mathsf{out}(x') | \mathrm{inport}(x) = \mathrm{outport}(x')\} \cup\\
& \{\mathsf{out}(x) = \mathsf{out}(x') | \mathrm{outport}(x) = \mathrm{outport}(x')\}.
\end{aligned}$$

Verhaltensbeschreibung

Die Verhaltensbeschreibung ist unabhängig von dem verwendeten Fehlersimulationsmodell. Eine Komponente verhält sich korrekt, wenn die Werte der Ein- und Ausgangsports gemeinsame Elemente besitzen. Durch die explizite Modellierung des Vorgabewertes einer restrict-Komponente ergibt sich für beide Komponententypen eine identische Verhaltensbeschreibung. Als *vorläufige* Komponentensemantik betrachten wir folgende Formeln:

$$\begin{aligned}
SD_{\mathrm{behaviour}} \quad = \quad & \{\mathsf{type{-}agree}(x) \wedge \neg\mathsf{ab}(x) \Rightarrow \mathsf{in}(x) \cap \mathsf{out}(x) \neq \emptyset\} \cup\\
& \{\mathsf{type{-}restrict}(x) \wedge \neg\mathsf{ab}(x) \Rightarrow \mathsf{in}(x) \cap \mathsf{out}(x) \neq \emptyset\}.
\end{aligned}$$

Die Formeln berücksichtigen nicht die Transitivitätsforderung des agree-Prädikates. Dieses Problem wird im Abschnitt 3.3.3 behandelt.

Systembeobachtung

Die Systembeobachtung ist abhängig von der vorliegenden Äußerung $\omega \in \Sigma^*$. Die Beziehung zwischen den Wortformen der Äußerung und den Modellvariablen wird durch eine Wertebindung b nach Definition 3.2.5 festgelegt. Sei r eine *ausgewählte* Lesart für ω nach Definition 3.2.4. Aus b und r ergibt sich nach Definition 3.2.6 eine Lesart r' der Modellvariablen, d.h. jeder Modellvariablen wird eine eindeutige Merkmalsstruktur zugeordnet. Daraus resultiert dann ein eindeutiger Wert für jeden Port.

$$\begin{aligned} OBS \; = \; & \{\mathsf{out}(x) = f(m) | \mathrm{outport}(x) = (v, m) \wedge r'(v) = f\} \cup \\ & \{\mathsf{in}(x) = f(m) | \mathrm{inport}(x) = (v, m) \wedge r'(v) = f\} \end{aligned}$$

3.3.2 Beispiel

Der formale Konstruktionsprozeß für ein System wird im folgenden für ein vereinfachtes Fehlersimulationsmodell der Übung (U1) aus Abschnitt 3.2.2 erläutert. Dabei wollen wir zur Veranschaulichung nur die Kongruenzbeziehungen im Kasus der beteiligten Wortformen betrachten. Es ergibt sich folgendes Fehlersimulationsmodell $M = (L, \{\mathsf{DET}, \mathsf{ADJ}, \mathsf{N}\}, P)$ mit einem Lexikon L und den Prädikaten

$$\begin{aligned} P = \{ \; & \mathsf{restrict}(\mathsf{N}, \mathsf{case}, \{\mathsf{acc}\}) \quad [res], \\ & \mathsf{agree}(\mathsf{DET}, \mathsf{N}, \mathsf{case}) \quad [agr_1], \\ & \mathsf{agree}(\mathsf{ADJ}, \mathsf{N}, \mathsf{case}) \quad [agr_2]\}. \end{aligned}$$

Aus dem Modell läßt sich eine Systembeschreibung konstruieren. Die Komponenten sind $COMPS = \{res, agr_1, agr_2\}$, wobei wir die Komponenten durch die in eckigen Klammern angegebenen Namen abkürzen. Die Typenbeschreibung liefert

$$SD_{\mathrm{type}} = \{\mathsf{type{-}restrict}(res), \mathsf{type{-}agree}(agr_1), \mathsf{type{-}agree}(agr_2)\}.$$

Die Verbindungsstruktur ist

$$SD_{\mathrm{connect}} = \{\mathsf{out}(res) = \mathsf{out}(agr_1), \mathsf{out}(agr_2) = \mathsf{out}(agr_1)\}.$$

Die Wertebereiche ergeben sich entsprechend. Da eine Systembeobachtung alle Werte bis auf den Eingangsport der Komponente *res* festlegt, wird nur dieser Wert angegeben.

$$SD_{\mathrm{domain}} = \{\mathsf{in}(res) = \{\mathsf{acc}\}, \ldots\}.$$

Das konstruierte System ist in Abbildung 3.3 dargestellt. Für die Schülereingabe

(3) * „Das Kind sucht *den gelbes Ball.*“

ergibt sich die minimale Diagnose $D = \{\mathsf{ab}(agr_2)\}$.

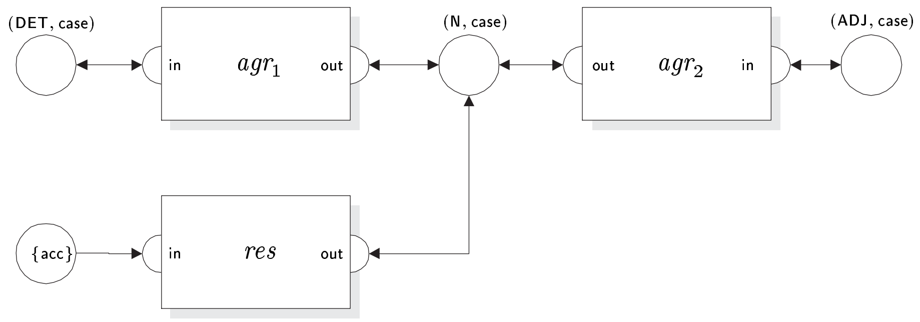

Abbildung 3.3: Komponentendarstellung für die Kasuskongruenz.

3.3.3 Wertaktualisierungsoperator

Die Verhaltensbeschreibung aus Abschnitt 3.3.1 erfüllt nicht die Forderung der Transitivität: In dem Eingabesatz (2) aus Abschnitt 3.3

(*) „Das Kind sucht *den gelbe Ball.*“

kongruieren sowohl der Artikel als auch das Adjektiv mit dem Substantiv im Kasus. Folglich werden alle Komponenten als korrekt angenommen. Das Problem könnte gelöst werden, indem eine weitere Kongruenzbedingung zwischen Artikel und Adjektiv eingeführt wird. Allerdings benötigt die Herstellung des transitiven Abschlusses theoretisch exponentiell viele Komponenten. In dem Satz

(4) „Der Bergsteiger *mit den kurzen roten Socken ...*“

müßten allein für die Kasuskongruenz zehn statt vier Komponenten verwendet werden.

In [Menzel 1992a] wird für die Sicherstellung der Transitivität eine Komponentensemantik vorgeschlagen, die nicht nur den Wahrheitswert überprüft, sondern außerdem noch eine *Wertaktualisierung* an den Ports durchführt. Der Wertaktualisierungsoperator $a := b$ weist dem Port a den Wert b zu. Der Wahrheitswert des Operators ist genau dann wahr, wenn $b \neq \emptyset$. Die Semantik des agree-Prädikates kann dann wie folgt dargestellt werden:[6]

$$\mathsf{type{-}agree}(x) \wedge \neg\mathsf{ab}(x) \Rightarrow \mathsf{in}(x) := \mathsf{out}(x) := \mathsf{in}(x) \cap \mathsf{out}(x).$$

Die Normalfallsemantik des restrict-Prädikates unterscheidet sich davon nur gering :

$$\mathsf{type{-}restrict}(x) \wedge \neg\mathsf{ab}(x) \Rightarrow \mathsf{out}(x) := \mathsf{in}(x) \cap \mathsf{out}(x).$$

Es findet keine Wertaktualisierung am Eingangsport statt, da dieser Wert durch den Kontext vorgegeben wird.

[6] Der Wertaktualisierungsoperator wird als rechtsassoziativ angenommen, d.h. $a := b := c$ entspricht $a := (b := c)$. Der Versuch, einem Port die leere Menge zuzuweisen, wird immer zurückgewiesen.

Die Auswertung eines Prädikates erzwingt eine Reduktion der Wertemenge der beteiligten Ports. Die Konsistenzprüfung einer Hypothese kann als ein *Constraint Satisfaction Problem* (CSP) aufgefaßt werden, in dem die Variablen den Ports und die Wertebereiche der Variablen der initialen Wertebelegung entsprechen. Die Prädikate sind dann binäre Bedingungen über den Werten der beteiligten Ports. Eine Reduktion eines Wertebereichs erzwingt in einem CSP die erneuerte Überprüfung von abhängigen Bedingungen. Eine Hypothese ist genau dann konsistent, falls alle Bedingungen erfüllt sind und keine Reaktivierungen mehr nötig sind. Die Normalfallsemantik kann als eine Konstruktionsvorschrift für die Herstellung lokaler Konsistenz im CSP angesehen werden.

Für die Übung aus dem vorherigen Abschnitt und dem Eingabesatz (2) ergeben sich zwei Diagnosen, wobei die erste der Akkusativlesart der Substantivgruppe entspricht und die zweite die ungewöhnliche, semantisch inadäquate Nominativlesart vorschlägt:

$$\begin{aligned} D_1 &= \{\mathsf{ab}(agr_2)\} \\ D_2 &= \{\mathsf{ab}(res), \mathsf{ab}(agr_1)\}. \end{aligned}$$

Die Diagnose D_1 liefert die Erklärung (E1) und unter Einbeziehung der Werte der beteiligten Ports die Erklärung (E2) aus Abschnitt 3.3:

(E1) „Das Adjektiv *gelbe* stimmt nicht mit dem Substantiv *Ball* im Kasus überein.“

(E2) „*Gelbe* steht im Nominativ, aber *Ball* steht im Akkusativ.“

Eine schematische Verbalisierung der Diagnose D_2 könnte z.B. folgende, kaum verständliche Erklärung liefern:

(E5) „Das Verb fordert eine Ergänzung im Akkusativ. Der Artikel *den* und das Substantiv *Ball* kongruieren nicht im Kasus.“

Die Ermittlung einer Diagnose erfordert die Aufstellung von Hypothesen, deren Konsistenz anschließend geprüft wird. Ein konkret realisiertes Berechnungsverfahren findet sich in [Stockfleth 1999]. Im Unterschied zur Definition 3.1.3 werden als minimale Diagnosen nur diejenigen betrachtet, die eine minimale Anzahl von Komponenten verletzen. Damit wird D_2 aus obigem Beispiel nicht als minimale Diagnose angesehen. In praktischen Anwendungen besteht das Problem meistens nicht darin, daß zu wenig Diagnosen vorhanden sind bzw. daß die das Problem besser charakterisierende Diagnose fehlt, sondern vielmehr in einer Selektion bzw. einer Aggregation der vielen vorhandenen Diagnosen, insbesondere dann, wenn mehrere Fehler in der Äußerung vorhanden sind.

3.3.4 Fehlerfallsemantik

Wird eine Komponente als fehlerhaft angenommen, so wurden bisher alle Werte an den Ein- und Ausgangsports erlaubt. In unseren Fall ist es jedoch möglich, das Verhalten im Fehlerfall genau zu spezifizieren. Dies geschieht durch Prädikatsvervollständigung ([Genesereth 1987, S.122]), indem wir annehmen, daß das

Verhalten einer Komponente gleichzeitig auch den Komponententyp und die Korrektheitsannahme festlegt.[7] Für die Fehlerfallsemantik des restrict- und des agree-Prädikates ergeben sich nach [Menzel 1992a] die Formeln[8]

$$\mathsf{type{-}restrict}(x) \wedge \mathsf{ab}(x) \Rightarrow \mathsf{out}(x) := \mathsf{out}(x)/\mathsf{in}(x)$$

$$\begin{aligned}&\mathsf{type{-}agree}(x) \wedge \mathsf{ab}(x) \Rightarrow\\ &\quad(\mathsf{in}(x) := \mathsf{in}(x) \cap \mathsf{out}(x) \wedge \mathsf{out}(x) := \mathsf{out}(x)/\mathsf{in}(x))\vee\\ &\quad(\mathsf{in}(x) := \mathsf{in}(x)/\mathsf{out}(x) \wedge \mathsf{out}(x) := \mathsf{in}(x) \cap \mathsf{out}(x))\vee\\ &\quad(\mathsf{in}(x) := \mathsf{in}(x)/\mathsf{out}(x) \wedge \mathsf{out}(x) := \mathsf{out}(x)/\mathsf{in}(x)).\end{aligned}$$

Die Aktualisierungsvorschrift des agree-Prädikates stellt alle Möglichkeiten für die Konstruktion von disjunkten Wertemengen dar. Dabei entsprechen die drei disjunktiv verknüpften Teilformeln[9] folgenden Fehlerkonstellationen:

- Es liegt ein Fehler am Eingang der Komponente vor. Der Wert des Ausgangsport ist jedoch korrekt.
- Der Wert des Ausgangsport ist fehlerhaft und der Wert des Eingangsport ist korrekt.
- Sowohl am Eingang als auch am Ausgang liegen fehlerhafte Werte vor.

Die Auswertung dieses Ausdrucks kann mehrere Wertaktualisierungen liefern, wodurch der Suchraum des Konsistenzverfahren beträchtlich wachsen kann. In [Menzel 1992a] wurde daher aus Effizienzgründen nur der Fall des fehlerhaften Ausgangs realisiert. Dies hat die Nachteile, daß einerseits das Verfahren nicht mehr vollständig ist und andererseits die Anordnung der Modellvariablen in den agree-Prädikaten relevant ist, was den Regelentwurf deutlich erschwert. Das in dieser Arbeit verwendete und erweiterte Verfahren aus [Stockfleth 1999] unterstützt alle drei Fehlerfälle.

Die Verwendung einer Fehlerfallsemantik hat folgende Vorteile:

- Es können Fehlerannahmen ausgeschlossen werden, wenn es keine gültige Wertaktualisierung gibt, wodurch der Suchraum verkleinert wird.
- Eine Fehlerannahme schränkt die Wertemenge weiter ein, wodurch eine Inkonsistenz früher entdeckt werden kann.
- Erst eine Fehlerfallsemantik ermöglicht eine effiziente Ermittlung eines Korrekturvorschlages (siehe Abschnitt 3.3.5).

[7] Eine Formel wird vervollständigt, indem eine Implikation durch eine Äquivalenz ersetzt wird, d.h. aus einer Folgerung $A \Rightarrow B$ wird eine Definition $A \Leftrightarrow B$. Angewendet auf die Normalfallsemantik der Komponententypen kann daraus die Fehlerfallsemantik abgeleitet werden.

[8] Das Komplement einer Menge A bzgl. einer Menge B ist $A/B = \{a \in A | a \notin B\}$.

[9] In einem Ausdruck der Form $a := c_1 \wedge b := c_2$ wird zuerst c_1 und c_2 ausgewertet. Erst wenn weder c_1 noch c_2 die leere Menge darstellen, erfolgt eine Wertaktualisierung der Ports. Eine Disjunktion von Zuweisungen $a := c_1 \vee b := c_2$ kann mehrere Wertaktualisierungen liefern, falls sowohl c_1 als auch c_2 nicht leer sind.

Für das Konsistenzverfahren zur Überprüfung von Hypothesen bedeutet die Verwendung von Fehlerfallsemantiken, daß die zu überprüfenden Bedingungen in Abhängigkeit der Korrektheitsannahme einfach nur ausgetauscht werden müssen, z.B. indem die Normalfallsemantik einer Komponente durch die Fehlerfallsemantik ersetzt wird.

3.3.5 Faktenfehler

Bisher wurden nur Fehler betrachtet, die auf einer Verletzung von Regeln in Form von restrict- und agree-Prädikaten basieren. Diese Art von Fehlern wollen wir als *Regelfehler* bezeichnen. Eine andere Sichtweise auf einen Fehler ergibt sich, wenn auch der initiale Wertebindungsprozeß in Zweifel gezogen werden kann. In der Schülereingabe (2)

(*) „Das Kind sucht *den gelbe Ball.*"

wird aufgrund des verwendeten Lexikons angenommen, daß das Adjektiv im Nominativ steht. Der Schüler könnte aber angenommen haben, daß das Adjektiv im Akkusativ steht. Unter dieser Annahme hätte er alle Regeln erfüllt. Der Fehler besteht daher in der Unkenntnis der Merkmalsbelegungen einer Wortform. Diese Art von Fehlern bezeichnen wir als *Faktenfehler*.

Für die Modellierung von Faktenfehlern wird folgendes Prädikat zusätzlich eingeführt:

$$\mathsf{value}(Modellvariable, Merkmal, Wert).$$

Diese Prädikate müssen nicht im Fehlersimulationsmodell formuliert werden, da sie automatisch für jede Lesart der verwendeten Wortformen mit Hilfe des Lexikons generiert werden können. Für unser Beispiel werden automatisch folgende Prädikate generiert:

$$\begin{aligned} P' = \{ \quad & \mathsf{value}(\mathsf{DET}, \mathsf{case}, \{\mathsf{acc}\}), \\ & \mathsf{value}(\mathsf{ADJ}, \mathsf{case}, \{\mathsf{nom}\}), \\ & \mathsf{value}(\mathsf{N}, \mathsf{case}, \{\mathsf{acc}, \mathsf{nom}, \mathsf{dat}\})\}. \end{aligned}$$

Die Darstellung als Komponente geschieht analog zum restrict-Prädikat, der Eingangswert entspricht allerdings nicht einem durch den Kontext vorgegebenen Wert, sondern dem Wert des Merkmals der verwendeten Wortform in einer Lesart. Eine value-Komponente verhält sich korrekt, wenn der Wert am Ausgang der Komponente eine Teilmenge des Eingangs ist: [10]

$$\mathsf{type{-}value}(x) \land \neg \mathsf{ab}(x) \Rightarrow \mathsf{out}(x) := \{v | v \subseteq \mathsf{in}(x) \land v \neq \emptyset\}.$$

Diese Komponentensemantik erweist sich aber als ungünstig, da eine Menge mit n Elementen 2^n Teilmengen besitzt, d.h. es existieren exponentiell viele verschiedene Wertaktualisierungen. In [Menzel 1992a] wurde gezeigt, daß auf eine explizite Modellierung von value-Komponenten verzichtet werden kann,

[10] Eine Disjunktion von alternativen Wertzuweisungen wird für die Normalfallsemantik der value-Komponente durch eine Menge von Werten repräsentiert.

indem die vorhandene Struktur der agree- und restrict-Komponenten beibehalten wird und dabei die Fehlerfallsemantik der Komponenten ersetzt wird. Aus diesem Grund unterscheiden wir zwei Modi für die Simulation von Fehlersimulationsmodellen:

- Regelfehlersimulation: Es wird die Komponentensemantik aus Abschnitt 3.3.3 und 3.3.4 verwendet.

- Faktenfehlersimulation: Es wird die Normalfallsemantik der Regelfehlersimulation verwendet. Für die Fehlerfallsemantik werden folgende Formeln verwendet:[11]

$$\begin{array}{ll} \mathsf{type}{-}\mathsf{restrict}(x) \wedge \mathsf{ab_f}(x) \Rightarrow & \mathsf{out}(x) := \mathsf{in}(x)/\mathsf{out}(x) \\ \mathsf{type}{-}\mathsf{agree}(x) \wedge \mathsf{ab_f}(x) \Rightarrow & (\mathsf{in}(x) := \mathsf{out}(x) := \mathsf{in}(x)/\mathsf{out}(x)) \vee \\ & (\mathsf{in}(x) := \mathsf{out}(x) := \mathsf{out}(x)/\mathsf{in}(x)) \end{array}$$

Die Definition der Fehlerfallsemantik in dem Modus der Faktenfehlersimulation bewirkt, daß nach der Auswertung die beteiligten Ports gemeinsame Werte haben. Für das Beispiel gilt nach der Auswertung von $\mathsf{ab_f}(agr_2)$ die Formel $\neg\mathsf{ab_f}(agr_2)$! Dies erfordert eine spezielle Behandlung im Konsistenzverfahren, da einmal korrigierte Bedingungen nicht mehr ausgewertet werden dürfen. Außerdem ist zu beachten, daß die Wertemengen der Ports durch diese Semantik nicht unbedingt verkleinert werden, wie es die allgemeinen Lösungsverfahren im CSP erwarten. Durch dieses nicht monotone Verhalten können Zyklen entstehen, die durch einen Buchhaltungsmechanismus verhindert werden müssen. Die Details des Berechnungsverfahren sind in [Stockfleth 1999] erläutert.

Für den Eingabesatz (2) liefert die Faktenfehlersimulation die Diagnose $D'_3 = \{\mathsf{ab_f}((agr_2))\}$. Eine einfache Transformationsvorschrift berechnet aus diesem Ergebnis die fehlgeschlagenen value-Komponenten, indem die Werte an den beteiligten Ports vor und nach der Simulation miteinander verglichen werden. Ist die Wertemenge eines beteiligten Ports nach der Simulation keine Teilmenge der Wertemenge vor der Simulation, so wurde genau dieser Port korrigiert. Die transformierte Diagnose für (2) ist

$$D_3 = \{\mathsf{ab_f}(\mathsf{value}(\mathsf{ADJ}, \mathsf{case}, \{\mathsf{nom}\}))\}.$$

Daraus kann die Erklärung (E3) generiert werden, indem der korrekte Wert miteinbezogen wird:

(E3) „Korrigiere das Adjektiv *gelbe*! Es steht im Nominativ, muß aber im Akkusativ stehen."

Der korrekte Wert kann weiterhin verwendet werden, um einen Korrekturvorschlag zu generieren.[12] Dafür muß das Lexikon oder ein definierter Ausschnitt des Lexikons nach Merkmalsstrukturen durchsucht werden, die mit der aus den Werten der Ports konstruierten Merkmalsstruktur kompatibel sind. Im Abschnitt 3.3.7 wird darauf noch eingegangen.

[11] Für die Kennzeichnung der Abnormalität im Modus der Faktenfehlersimulation wird das Prädikat $\mathsf{ab_f}$ verwendet.

[12] Siehe Erklärung (E4) in Abschnitt 3.3.

3.3.6 Eine realistische Übung

Die triviale Übung aus Abschnitt 3.3.2 wird sicherlich niemals in einem Fremdsprachenlernsystem Verwendung finden, da selbst für die einfache Substantivgruppe wesentliche Wohlgeformtheitsbedingungen fehlen. Es müssen mindestens die Kongruenzbeziehungen zwischen Artikel, Adjektiv und Substantiv in Kasus, Genus und Numerus sowie die Flexionsforderung des Artikels bezüglich des verwendeten Adjektivs unterstützt werden. Das im Abschnitt 3.2.2 entwickelte Fehlersimulationsmodell kann hierfür verwendet werden.

Für die Generierung eines Korrekturvorschlags ist es sinnvoll, die Wortarten der verwendeten Wortformen aufgrund des Übungskontextes einzuschränken. Diese Einschränkung soll als nicht revidierbar angesehen werden, d.h. die Hypothesengenerierung darf hierfür keine verletzten Komponenten erzeugen. In dem Formalismus werden nicht verletzbare Prädikate durch ein vorangestelltes Ausrufezeichen notiert:

$$!\mathsf{restrict}(\mathsf{ADJ}, \mathsf{cat}, \{\mathsf{adj}\}), !\mathsf{restrict}(\mathsf{N}, \mathsf{cat}, \{\mathsf{n}\}), !\mathsf{restrict}(\mathsf{DET}, \mathsf{cat}, \{\mathsf{det}\}).$$

Die Diagnosen im vorherigen Abschnitt setzen eine triviale lexikalische Modellierung voraus, die in natürlicher Sprache nicht gegeben ist. Insbesondere kann selbst durch die Darstellung von Disjunktionen als Menge von Symbolen in den Merkmalsstrukturen nicht verhindert werden, daß einige Wortformen mehrere Merkmalsstrukturen benötigen.

Die Abbildung 3.4 stellt ein erweitertes Komponentennetz für den Eingabesatz

(5) „Das Kind sucht *der gelbe Ball.*“

dar.[13] Die Merkmalsstrukturen der Wortformen werden dabei durch kleine Kreise repräsentiert. Die gestrichelten Linien, die alle in einem Kreis zusammenführen, verbinden eine Merkmalsstruktur mit den zugehörigen Merkmalen und Werten.

Der Eingabesatz hat insgesamt $3 \cdot 1 \cdot 4 = 12$ Lesarten.[14] Die Analyse der Äußerung konstruiert daher zwölf Komponentennetze, die alle jeweils für den Modus der Regel- und Faktenfehleranalyse simuliert werden müssen. Jedes einzelne Komponentennetz liefert eine Menge von minimalen Diagnosen, wobei wir als Minimalitätskriterium die Anzahl der verletzten Komponenten verwenden. Allerdings gibt es natürlich Lesarten, die weniger Komponentenverletzungen aufweisen als andere. Als Diagnose der Äußerung verstehen wir daher die Vereinigung der *global* minimalen Diagnosen der einzelnen Komponentennetze bezogen auf den verwendeten Simulationsmodus. Alle global minimalen Diagnosen haben gleiche Kardinalität, und es gibt keine andere minimale Diagnose, die eine geringere Kardinalität aufweist.

Formal gesehen reicht es nicht aus, eine Diagnose nur durch die verletzten Komponenten zu repräsentieren. Es wird zusätzlich noch die Wertebelegung der Ports vor der Simulation, d.h. die zugehörige Lesart, und die Wertebelegung nach der Simulation für jede einzelne, minimale Diagnose benötigt. Diese

[13] Die Einschränkungen der Wortarten sind nicht enthalten.

[14] Falls keine Mengenrepräsentation verwendet würde, wären es $6 \cdot 3 \cdot 15 = 270$ Lesarten!

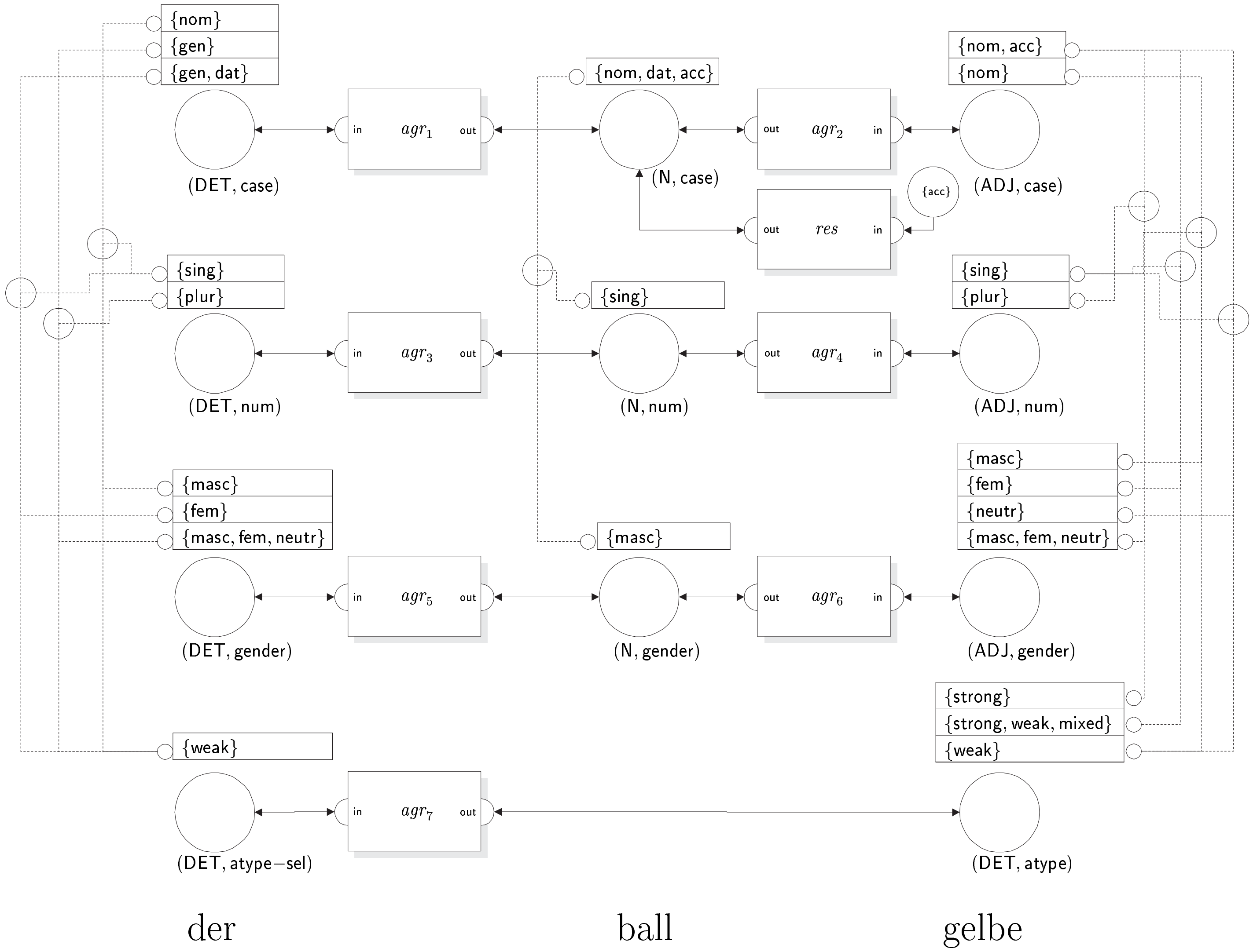

Abbildung 3.4: Lexikalische Mehrdeutigkeit im Komponentennetz.

Information ist vor allem notwendig, um einen Korrekturvorschlag generieren zu können.

Der Eingabesatz liefert folgende, global minimalen Diagnosen:

$$
\begin{aligned}
D_1 &= \{\mathsf{ab}(\mathsf{restrict}(\mathsf{N}, \mathsf{case}, \{\mathsf{acc}\}))\} \\
D_2 &= \{\mathsf{ab_f}(\mathsf{value}(\mathsf{DET}, \mathsf{case}, \{\mathsf{nom}\})), \mathsf{ab_f}(\mathsf{value}(\mathsf{ADJ}, \mathsf{case}, \{\mathsf{nom}\}))\} \\
D_3 &= \{\mathsf{ab_f}(\mathsf{value}(\mathsf{DET}, \mathsf{case}, \{\mathsf{nom}\})), \mathsf{ab_f}(\mathsf{value}(\mathsf{ADJ}, \mathsf{gender}, \{\mathsf{fem}\}))\} \\
D_4 &= \{\mathsf{ab_f}(\mathsf{value}(\mathsf{DET}, \mathsf{case}, \{\mathsf{gen}, \mathsf{dat}\})), \mathsf{ab_f}(\mathsf{value}(\mathsf{N}, \mathsf{gender}, \{\mathsf{masc}\}))\} \\
D_5 &= \{\mathsf{ab_f}(\mathsf{value}(\mathsf{DET}, \mathsf{case}, \{\mathsf{nom}\})), \mathsf{ab_f}(\mathsf{value}(\mathsf{ADJ}, \mathsf{gender}, \{\mathsf{neutr}\}))\}
\end{aligned}
$$

Die Diagnose D_1 ist das Ergebnis der Regelfehlersimulation und symbolisiert die Forderung des Verbs, eine Ergänzung im Akkusativ statt im Nominativ zu verwenden. Die Diagnosen D_2, D_3, D_4 und D_5 sind das Ergebnis der Faktenfehlersimulation. Allen Diagnosen liegen verschiedene Lesarten zugrunde:

Lesart	Diagnose	Kasus Artikel	Genus Artikel	Kasus Adjektiv	Genus Adjektiv
1	D_1, D_2	$\{\mathsf{nom}\}$	$\{\mathsf{masc}\}$	$\{\mathsf{nom}\}$	$\{\mathsf{masc}\}$
2	D_3	$\{\mathsf{nom}\}$	$\{\mathsf{masc}\}$	$\{\mathsf{nom}, \mathsf{acc}\}$	$\{\mathsf{fem}\}$
3	D_4	$\{\mathsf{gen}, \mathsf{dat}\}$	$\{\mathsf{fem}\}$	$\{\mathsf{nom}, \mathsf{acc}\}$	$\{\mathsf{fem}\}$
4	D_5	$\{\mathsf{nom}\}$	$\{\mathsf{masc}\}$	$\{\mathsf{nom}, \mathsf{acc}\}$	$\{\mathsf{neutr}\}$

In welcher Reihenfolge und wieviele Diagnosen dem Schüler präsentiert werden sollen, hängt von der verwendeten Fehlerselektionsstrategie ab.

Die Diagnose D_4 erfordert als Korrekturvorschlag die Änderung des Genus des Substantivs. Da Substantive aber invariant im Genus sind, wäre es sinnvoll, derartige Diagnosen vollständig zu verbieten. Der Formalismus wurde deshalb dahingehend erweitert, daß in einzelnen Prädikaten Merkmale der Modellvariablen geschützt werden können. Im Komponentennetz kann der zugehörige Port im Modus der Faktenfehlersimulation unter Verwendung der Fehlerfallsemantik nicht verändert werden, wodurch bestimmte Wertaktualisierungen nicht möglich sind. Notiert wird dieser Schutzmechanismus durch ein der Modellvariablen vorangestelltes Ausrufezeichen:

$$\mathsf{agree}(\mathsf{ADJ}, !\mathsf{N}, \mathsf{gender}), \mathsf{agree}(\mathsf{DET}, !\mathsf{N}, \mathsf{gender}).$$

Die Erweiterung bewirkt, daß die Diagnose D_4 nicht generiert wird. [15]

[15] Der Schutz des Genusmerkmals verhindert, daß in dem Satz „Ich suche die gelbe Ball." der schwer interpretierbare Korrekturhinweis „Ändere das Genus des Substantivs!" sowie eine Korrektur der Äußerung, die das Substantiv z.B. durch „Vase" ersetzt, generiert wird. Stattdessen wird das Genus des Artikels und des Adjektivs als fehlerhaft angenommen, wodurch der Korrekturvorschlag „Ich suche den gelben Ball." ermöglicht wird. Leider geht die plausible Erklärung „Ball ist nicht feminin, sondern maskulin!", die ohne Schutzmechanismus diagnostiziert werden kann, verloren. Es wäre daher am besten, wenn der Schutzmechanismus für das Genusmerkmal erst dann aktiviert wird, wenn eine Korrektur der Äußerung vom Schüler gewünscht wird. In diesem Fall müßte die Faktenfehlersimulation unter Berücksichtigung des geschützten Merkmals wiederholt werden. Ein derartiges Vorgehen muß allerdings nicht für alle geschützten Merkmale sinnvoll sein. Daher wäre ein zusätzlicher Schutzoperator für die zweistufige Faktenfehlersimulation notwendig. Dies wird in der aktuellen Implementation noch nicht berücksichtigt.

3.3.7 Fehlererklärung

Ausgehend von einer kurzen Diskussion eines Verbalisierungsverfahrens für Diagnoseergebnisse wird die Ermittlung eines Korrekturvorschlages aus den verletzten value-Komponenten vorgestellt. Anschließend werden verschiedene Selektionsstrategien für die Auswahl von minimalen Diagnosen behandelt.

Fehlerverbalisierung

Die symbolische Darstellung einer Diagnose eignet sich nicht für die direkte Repräsentation in einem Fremdsprachenlernsystem. In [Menzel 1992a] wird ein musterbasierter Mechanismus für die Verbalisierung von Diagnosen vorgestellt. Für ein verletztes agree-Prädikat könnte z.B. folgendes Ersetzungsschema verwendet werden:

$$\mathsf{agree}(x, y, m) \Rightarrow \text{„}x \text{ stimmt nicht mit } y \text{ im } m \text{ überein.“}$$

Für die Erklärung von Regel- oder Faktenfehlern können die aktuellen Werte der beteiligten Ports in der Verbalisierung verwendet werden. Ein weiteres Schema könnte dann eine Erklärung mit Bezug auf die beteiligten Werte der Merkmale ausgeben:

$$\mathsf{agree}(x, y, m) \Rightarrow \text{„}x \text{ ist } x(m)\text{, aber } y \text{ ist } y(m)\text{.“}$$

Für die einzelnen Symbole werden noch Übersetzungsregeln benötigt, z.B. ADJ $\Rightarrow$ „Adjektiv“.

Werden mehrere Komponenten innerhalb einer Diagnose verletzt, könnten zwar die einzelnen Komponenten verbalisiert und mit „und“ verknüpft werden, dies ergäbe jedoch sehr umständliche und unleserliche Erklärungen. In [Menzel 1992a] wird daher ein Verdichtungsverfahren verwendet, welches mehrere verletzte Komponenten, die einem bestimmten Muster genügen, durch eine aggregierende symbolische Darstellung ersetzt. Auf das Verdichtungsergebnis können dann die obigen Erklärungsschemata angewandt werden. Folgendes Verdichtungsschema vereinfacht z.B. Komponenten, die sich auf ein gemeinsames Merkmal des Eingangsports beziehen:

$$\mathsf{and{-}source{-}agree}(\{x_1, \ldots, x_n\}, y, m) \Leftarrow \mathsf{agree}(x_1, y, m) \wedge \ldots \wedge \mathsf{agree}(x_n, y, m).$$

Im allgemeinen gibt es mehrere, konkurrierende Verdichtungsmöglichkeiten für eine Diagnose.

Eine konkrete Realisierung dieser Verfahren findet sich in [Stockfleth 1999].

Korrekturen

Der Modus der Faktenfehlersimulation aus Abschnitt 3.3.5 liefert nach der Auswertung eine Wertebelegung an den Ports, die allen Prädikaten genügt. Aus den Werten der Ports kann eine Merkmalsstruktur konstruiert werden, die verwendet wird, um aus einem Ausschnitt des Lexikons Wortformen zu finden, welche eine kompatible Merkmalsstruktur besitzen. Eine Merkmalsstruktur f_1 ist kompatibel mit einer Merkmalsstruktur f_2, falls für alle Merkmale m von

f_1, die in f_2 vorkommen, gilt: $f_1(m) \cap f_2(m) \neq \emptyset$. Die gefundenen Wortformen können dann als Korrekturvorschlag verwendet werden. Der Formalismus wurde um eine einfache Indizierung von Lexikoneinträgen erweitert. Wortformen, die untereinander für eine Korrektur in Frage kommen, werden in einer Menge notiert:

```
CORRECTION
  det :    {ein,eine,einer,eines,einen,einem,
            der,die,das,den,dem,den}
  ball :   {ball,bälle,balle,balles,bällen}
  gelb :   {gelbe,gelben,gelbes,gelber,gelbem}
END
```

Die Faktenfehlerdiagnosen D_2, D_3, D_5 aus Abschnitt 3.3.6 liefern alle die korrigierte Phrase „den gelben Ball", obwohl unterschiedliche Lesarten verwendet wurden. Die Diagnose D_4 kann nicht korrigiert werden, da alle aufgrund des Korrekturindizes in Frage kommenden Wortformen eine inkompatible Merkmalsstruktur aufweisen. Theoretisch kann die Faktenfehlersimulation eine Merkmalsstruktur liefern, für die es im gesamten Lexikon keine kompatible Merkmalsstruktur gibt, da die Wertaktualisierung nicht monoton ist.
In [Stockfleth 1999] wird dafür ein Beispiel angegeben.

Fehlerselektion

Eine Übersicht von Selektionsstrategien findet sich in [Menzel 1992a]. Es folgt ein kurzer Ausschnitt einiger Strategien:

- Eine Entscheidung kann abhängig gemacht werden von dem verwendeten Simulationsmodus, z.B. „Bevorzuge Faktenfehler vor Regelfehler."
- Faktenfehler sind syntaktisch tiefer in der Struktur eingebunden als Regelfehler. In einigen Fällen sollten syntaktisch höher eingestufte Fehler, wie z.B. die Diagnose D_1 aus Abschnitt 3.3.6, bevorzugt werden.
- Ein anderes Kriterium ist die Auswahl von Diagnosen, die sich am einfachsten verdichten lassen und z.B. nach der Verdichtung die geringste Kardinalität aufweisen.
- Häufig hilft auch die Präferenz von Diagnosen, die sich auf ein bestimmtes Merkmal beziehen, z.B. „Bevorzuge eine Verletzung im Genus vor einer Verletzung im Kasus."
- Diagnosen, die einen Korrekturvorschlag erlauben, sollten vor nicht korrigierbaren Diagnosen bevorzugt werden.
- Die Selektionsstrategie kann sich in Abhängigkeit vom aktuellen Studentenmodell ändern. Kann z.B. aufgrund der bisherigen Erfahrung im Übungsverlauf davon ausgegangen werden, daß der Schüler bestimmte Merkmale der Wortformen beherrscht, sind Regelfehlerdiagnosen zu bevorzugen.

- Die Diagnosen können aufgrund der unterschiedlichen Lesarten geordnet werden, indem einige Lesarten gegenüber anderen präferiert werden.
- Im Falle von identischen Korrekturvorschlägen sollte nur eine Faktenfehlerdiagnose verwendet werden.

Eine gute Selektionsstrategie zu entwickeln, ist sicherlich keine einfache Aufgabe, insbesondere weil die skizzierten Kriterien häufig miteinander konkurrieren. Wird der Eingabesatz „die gelbe Ball“ für die einfache Übung aus Abschnitt 3.3.6 verwendet, gibt es insgesamt schon sechzehn Diagnosen, die alle genau zwei Komponenten verletzen. Allein die Anzahl alternativer Diagnosen, häufig durch die vielen möglichen Lesarten begründet, verlangt nicht nur eine Ordnung der Diagnosen, sondern auch eine Reduktion der Diagnosemenge auf eine vernünftige Größe, will man das Verfahren der Fehlersimulation in einem Fremdsprachenlernsystem anwenden.

Kapitel 4

Fehlerdiagnose mit CDG-Parsing

Dieses Kapitel behandelt ein Verfahren, welches auch für fehlerhafte, d.h. von einer normativen Grammatik abweichende Eingaben Strukturbeschreibungen generieren kann. Das Verfahren basiert auf dem Grammatikformalismus *Constraint Dependency Grammar* (kurz CDG oder CD-Grammatik). Dieser Formalismus wird im Abschnitt 4.1 vorgestellt. Eine für die Fehlerdiagnose von freien Äußerungen nötige Erweiterung des Formalismus, im folgenden als *bewertete CD-Grammatik* bezeichnet, wird im Abschnitt 4.2 behandelt. Neben der Möglichkeit der robusten Verarbeitung von nicht wohlgeformten Sätzen kann der Formalismus auch für die Realisierung von Fremdsprachenlernsystemen verwendet werden. Der in diesem Anwendungszusammenhang zentrale Begriff der Diagnose einer Äußerung ist Gegenstand des Abschnittes 4.3. Anhand von Beispielen werden Modellierungstechniken für Grammatikregeln und Verfahren für die Generierung von Fehlererklärungen erläutert.

4.1 Der Formalismus der CD-Grammatik

In diesem Abschnitt wird der Formalismus der *Constraint Dependency Grammar* vorgestellt. Er wurde zuerst von [Maruyama 1990] vorgeschlagen und basiert auf der in der Dependenzgrammatik ([Tesnière 1959]) gebräuchlichen Relation der *Dependenz*. In [Harper 1993] wird der Formalismus um die Verarbeitung von Wortgraphen, das für Spracherkennungssysteme übliche Ausgabeformat, erweitert. In dieser Arbeit wird nur ein Spezialfall dieser Erweiterung benötigt, welcher die Verarbeitung von lexikalisch mehrdeutigen Wortformen erlaubt. Dependenzrelationen beziehen sich in der in Abschnitt 4.1.3 vorgestellten Formalisierung nicht mehr auf Wortformen, sondern auf lexikalisch eindeutige Repräsentanten. Im Abschnitt 4.1.4 wird der Begriff der *Strukturbeschreibung* formal definiert. Diese Definition erweist sich insbesondere für die Behandlung von bewerteten CD-Grammatiken als sinnvoll. Die Darstellung des Parsingverfahrens als *Constraint Satisfaction Problem* ist Gegenstand des Abschnittes 4.1.5.

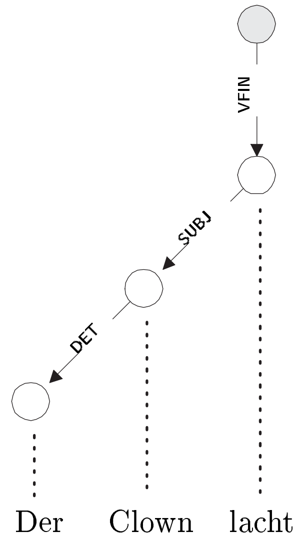

Abbildung 4.1: Dependenzstruktur.

4.1.1 Dependenzrelation und Dependenzstruktur

Eine Dependenzrelation ist eine binäre Relation, die Bestandteile einer Äußerung in Beziehung setzt, notiert als $\mathrm{dep}(\cdot,\cdot)$. Als Bestandteile werden dabei in der Regel die auf der Oberfläche realisierten Wortformen angesehen. Die Relation ist antisymmetrisch, antireflexiv und antitransitiv ([Menzel 2000]). Gilt $\mathrm{dep}(x, y)$, so sagt man, daß x von y abhängig ist (bzw. x ist y untergeordnet oder x modifiziert y), bzw. komplementär dazu y dominiert (unmittelbar) x. In dieser Arbeit wird statt der unbenannten Dependenzrelation eine Menge von feiner differenzierenden Dependenzrelationen verwendet (z.B. subjekt-von, direktes-objekt-von). In dem Satz

(6) „Der Clown lacht."

ist z.B. das Substantiv „Clown" abhängig vom finiten Verb „lacht". Dabei erfüllt das Substantiv die syntaktische Funktion des Subjekts. Dependenzstrukturen können als verbundene, gerichtete Graphen angesehen werden ([Melcuk 1988]). In den meisten Dependenztheorien werden nur Baumstrukturen zugelassen. Eine mögliche Dependenzstruktur für Satz (6) ist in Abbildung 4.1 dargestellt. Die Dependenzrelationen werden durch Kantenbezeichnungen repräsentiert. Der graue Kreis in der Abbildung symbolisiert einen speziellen Wurzelknoten, welcher keine Entsprechung auf der Oberfläche der Äußerung besitzt. Die gerichteten Kanten in der grafischen Darstellung der Dependenzstruktur repräsentieren die unmittelbare Dominanz, d.h. für $\mathrm{dep}(x, y)$ geht eine Kante vom Knoten y zum Knoten x.

4.1.2 Parsing als Disambiguierungsprozeß

Das Parsingprinzip unterscheidet sich wesentlich von Ansätzen, die auf kontextfreien Regeln basieren. In diesen Ansätzen wird eine Strukturbeschreibung

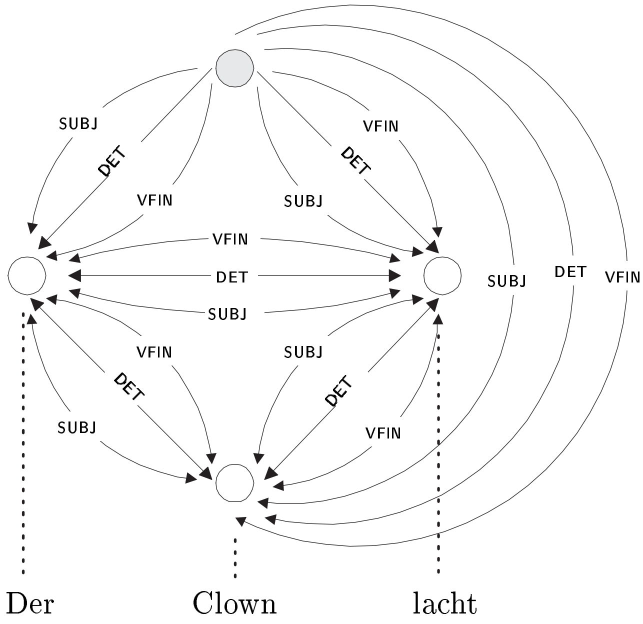

Abbildung 4.2: Dependenzgraph.

meist *konstruktiv* ermittelt, d.h. ausgehend von partiellen Strukturen werden durch Regeln neue Strukturen erstellt. In CD-Grammatiken hingegen wird von einer kompakten Repräsentation *aller* möglichen Strukturbeschreibungen einer Äußerung ausgegangen, aus der anhand der Regeln als ungültig identifizierte Kandidaten gelöscht werden. Der Unterschied wird besonders deutlich, wenn eine Äußerung mit einer Grammatik analysiert werden soll, die keine Regeln enthält. Der kontextfreie Ansatz liefert in diesem Fall keine Strukturbeschreibung, während die CDG-Analyse alle möglichen Strukturen liefert. Jede weitere Regel der CD-Grammatik *reduziert* die Anzahl der potentiellen Strukturen. In einer kontextfreien Grammatik dagegen werden erst durch neue Regeln weitere Strukturen ermöglicht. Das CDG-Parsingverfahren wird daher in der Literatur als *Disambiguierungsprozeß* angesehen, bzw. es wird vom *eliminativen* Parsing gesprochen.

CD-Grammatiken verwenden Dependenzstrukturen für die Beschreibung der Beziehungen zwischen Wortformen. Sie eignen sich besonders gut, weil die Anzahl der möglichen Unterordnungen einer Wortform nur linear in der Länge der Äußerung wächst. Dadurch kann jeder Wortform eine Variable zugewiesen werden, deren Wertebereich der Menge aller möglichen Unterordnungen entspricht. Diese Repräsentation ermöglicht eine Realisierung des Parsingverfahrens als *Constraint Satisfaction Problem* (CSP).

In Abbildung 4.2 sind alle möglichen Unterordnungsrelationen des Satzes (6) in einem Graphen dargestellt.[1] Dieser Graph wird im folgenden als *Dependenzgraph* einer Äußerung bezeichnet. Eine Dependenzstruktur ist ein Teil-

[1] Die Kanten mit zwei Pfeilen symbolisieren zwei einzelne Dependenzbeziehungen, wobei eine Kante vom dominierenden zum abhängigen Knoten führt.

graph, welcher dem Baumkriterium und den Regeln der Grammatik genügt (siehe Abbildung 4.1). Regeln sind dabei logische Ausdrücke auf den Kanten des Dependenzgraphens, d.h. auf den Elementen der zugehörigen Dependenzrelationen. In konkreten Implementationen werden aus Effizienzgründen maximal zwei Kanten miteinander verglichen, wodurch der Anwendungsbereich einer Regel, welche lokale Unterordnungen ausschließt, eingeschränkt wird.

Der Formalismus von Maruyama erlaubt die Verwendung von mehreren *Analyseebenen*[2]. Für jede dieser Ebenen beschreibt die Grammatik die Menge aller gültigen Dependenzstrukturen. Dabei können verschiedene Dependenzrelationen pro Ebene verwendet werden. Für die Beschreibung einer Semantikebene verwendet [Menzel 1995] u.a. die Relationen agens-von und patiens-von. Ein Vorteil des Ebenenkonzeptes ist die Möglichkeit, Robustheit durch Redundanz in der Modellierung der sprachlichen Kompetenz zu erreichen. Die Kopplung der ebenenspezifischen Wissensmodellierung wird erreicht, indem Regeln die Dependenzrelationen der einzelnen Ebenen miteinander in Beziehung setzen. Damit sind Modellierungen möglich, in denen jede weitestgehend autonome Ebene nur über wenige Regeln einen bidirektionalen Informationsaustausch zu anderen Ebenen realisieren kann. Insbesondere kann versucht werden, ein Modell der Interaktion von Syntax, Semantik und Pragmatik in dem Formalismus zu realisieren.[3]

4.1.3 Formale Definition einer CD-Grammatik

Der Formalismus der CD-Grammatik benötigt ein Lexikon, in dem jeder Lexikoneintrag nur Merkmalsstrukturen mit nicht ambigen Werten enthält. Ein *CD-Lexikon* ist ein Lexikon nach Definition 3.2.2, welches die möglichen Merkmalsstrukturen einschränkt.

Definition 4.1.1 (CD-Lexikon) *Die Menge aller Merkmalsstrukturen sei F. Ein CD-Lexikon L ist ein Lexikon $L \subseteq \Sigma \times F$ mit der Einschränkung, daß nur einzelne Symbole als Merkmalswerte verwendet werden dürfen:*

$$\forall (w, f) \in L : \forall m \in \mathrm{dom}(f) : |f(m)| = 1$$

Ein Lexikoneintrag wird im folgenden mit λ bezeichnet. Die Auswahl eines Lexikoneintrags $\lambda = (w, f)$ für eine Wortform w kann als eine Lesart r für w bezüglich eines CD-Lexikon L angesehen werden (vgl. Definition 3.2.4). Ein CD-Lexikon benötigt im allgemeinen mehr Merkmalsstrukturen als das Lexikon für die Fehlersimulation. Für die Wortform „der“ aus Abbildung 3.4 auf Seite 43 gibt es für ein CD-Lexikon insgesamt sechs Lesarten, welche durch hochgestellte Indizes unterschieden werden: $\mathsf{der}^1, \ldots, \mathsf{der}^6$.

Die Knoten in einer Dependenzstruktur repräsentieren genau eine Lesart der verwendeten Wortformen sowie die Position innerhalb der Äußerung. Im folgenden wird für diese Informationsstruktur der Begriff des *L-Knotens* verwendet.

[2] Maruyama und Harper sprechen von verschiedenen Rollen (engl. *roles*). Eine Rolle kann als eine spezifische Sicht auf die Beziehungen zwischen den Wortformen der Äußerung angesehen werden.

[3] Semantische Netze und Dependenzstrukturen weisen aufgrund ihres relationalen Basisprinzips viele Gemeinsamkeiten auf ([Menzel 2000]).

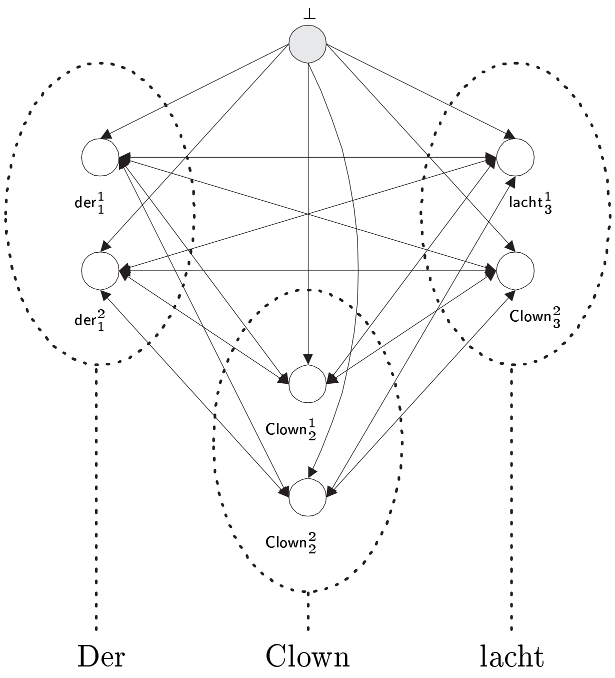

Abbildung 4.3: Dependenzgraph unter Berücksichtigung lexikalischer Mehrdeutigkeit.

Ein L-Knoten besteht aus einem Lexikoneintrag λ und einem Positionsindex. Die Position wird durch einen tiefergestellten Index notiert, z.B. der_1^3.

Definition 4.1.2 (L-Knoten) *Sei L ein CD-Lexikon und $\omega = w_1 \dots w_n \in \Sigma^*$ eine Äußerung. Ein L-Knoten v für die Wortform w_i ist ein Paar $v = ((w_i, f), i)$ mit $(w_i, f) \in L$. Die Menge aller L-Knoten für ω ist*

$$\mathcal{V}(\omega) = \{((w_i, f), i) | 1 \leq i \leq n \wedge (w_i, f) \in L\}.$$

Da die Regeln der Grammatik unabhängig von der vorliegenden Äußerung formuliert werden, wird die Menge aller *potentiellen* L-Knoten benötigt. Die unendliche Menge aller potentiellen L-Knoten ist

$$\mathcal{V} = \{(\lambda, i) | \exists \omega \in \Sigma^* : (\lambda, i) \in \mathcal{V}(\omega)\}.$$

In Abbildung 4.3 ist ein vereinfachter Dependenzgraph unter Berücksichtigung lexikalischer Mehrdeutigkeit dargestellt, wobei nur sechs von elf L-Knoten gezeichnet wurden.

Eine CD-Grammatik beinhaltet im wesentlichen eine Menge von Kantenbezeichnungen für jede Analyseebene und eine Menge von Regeln, die als unäre und binäre Bedingungen auf den Kanten der möglichen Dependenzgraphen formuliert werden. Zusätzlich zu den potentiellen L-Knoten wird noch ein Wurzelknoten eingeführt.

Definition 4.1.3 (CD-Grammatik) *Eine CD-Grammatik G ist ein Tupel $(L, \perp, \mathcal{L}, R, C)$ mit*

- L *: Ein CD-Lexikon.*
- $\perp$ *: Ein Symbol für den Wurzelknoten.*
- $R = \{r_1, \ldots, r_m\}$ *: Eine endliche Menge von Analyseebenen.*
- $\mathcal{L} = \bigcup_{1 \leq i \leq m} \{(r_i, l) \mid l \in L_i\}$ *: Eine endliche Menge von Bezeichnungen L_i pro Analyseebene r_i.*
- $C = \{c_1^u, \ldots, c_{n_u}^u, c_1^b, \ldots, c_{n_b}^b\}$ *: Eine endliche Menge von unären und binären Bedingungen. Sie werden über den möglichen Kanten $E = (\mathcal{V} \cup \{\perp\}) \times \mathcal{L} \times \mathcal{V}$ formuliert:*[4]
  - *Unäre Bedingungen sind $c_i^u \subseteq E$ mit $1 \leq i \leq n_u$.*
  - *Binäre Bedingungen sind $c_i^b \subseteq E \times E$ mit $1 \leq i \leq n_b$.*

Regeln werden durch logische Ausdrücke notiert mit den üblichen Junktoren (z.B. $\rightarrow, \neg, \wedge, \vee$) und Prädikaten (z.B. $=, \neq, <, \leq, >, \geq$). Folgende Funktionen ermöglichen den Zugriff auf den Informationsgehalt einer Kante bzw. den zugehörigen Knoten:

- Informationsgehalt einer Kante $\mathsf{X} = (v, (r, l), v')$:
  - $\mathsf{X.label}$ liefert die Beschriftung einer Kante, l.
  - $\mathsf{X.level}$ liefert die Analyseebene einer Kante, r.
  - $\mathsf{X} \uparrow$ bzw. $\mathsf{X} \downarrow$ liefern den oberen bzw. unteren Knoten einer Kante, v bzw. v'.
- Informationsgehalt eines L-Knotens $v = ((w, f), i)$ bzw. des Wurzelknotens $v = \perp$:
  - Das Prädikat root angewendet auf einen Knoten liefert genau dann wahr, wenn der Knoten die Wurzel kennzeichnet: $\mathsf{root}(v) \Leftrightarrow v = \perp$.
  - pos liefert die Position eines L-Knotens in der Äußerung, i.
  - word liefert die Wortform eines L-Knotens, w.
  - Jeder andere Bezeichner m liefert den Wert des Merkmals der zugehörigen Merkmalsstruktur eines L-Knotens, $f(m)$. Das Prädikat exists angewendet auf einen L-Knoten und ein Merkmal liefert genau dann wahr, wenn das Merkmal in der Merkmalsstruktur des Lexikoneintrags vorhanden ist, d.h. $m \in \mathrm{dom}(f)$.

Bei der Formulierung von Bedingungen ist darauf zu achten, daß keine Typverletzung bei den Zugriffsfunktionen auftritt: Der Wurzelknoten kennt keine Position, keine Wortform und keine Merkmalsstruktur.

[4] Eine Kante $(v, (r, l), v')$ verbindet im Dependenzgraphen der Ebene r den Knoten v mit dem Knoten v' und ist mit l bezeichnet. Erst wenn eine Äußerung ω vorliegt, sind die L-Knoten bekannt ($\mathcal{V}(\omega)$) und damit die Dependenzgraphen der Ebenen eindeutig bestimmt.

Beispiel

Die Unterordnung eines Artikels auf der Syntaxebene kann durch folgendes Prädikat ausgedrückt werden:

$$\begin{aligned}\mathsf{det{-}modifies{-}noun(X)} \quad \Leftrightarrow \quad & \mathsf{X.level} = \mathsf{SYN} \wedge \mathsf{X.label} = \mathsf{DET} \wedge \\ & \neg\mathsf{root}(\mathsf{X}\uparrow) \rightarrow \\ & \mathsf{X}\downarrow \mathsf{cat} = \mathsf{det} \wedge \mathsf{X}\uparrow \mathsf{cat} = \mathsf{n}.\end{aligned}$$

Die Bedingung, notiert als Menge, ist dann

$$c_1{}^u = \{e \in E | \mathsf{det{-}modifies{-}noun(e)}\}.$$

Die Forderung, daß ein Artikel vor einem attributiven Adjektiv steht, kann als binäre Bedingung formuliert werden:

$$\begin{aligned}\mathsf{det{-}precedes{-}adj(X, Y)} \quad \Leftrightarrow \quad & \mathsf{X.level} = \mathsf{SYN} \wedge \mathsf{Y.level} = \mathsf{SYN} \wedge \\ & \mathsf{X}\uparrow = \mathsf{Y}\uparrow \wedge \\ & \mathsf{X.label} = \mathsf{DET} \wedge \mathsf{Y.label} = \mathsf{ADJ} \rightarrow \\ & \mathsf{X}\downarrow \mathsf{pos} < \mathsf{Y}\downarrow \mathsf{pos}.\end{aligned}$$

Dabei wird von zwei verschiedenen Kanten ausgegangen:

$$c_1{}^b = \{(e, e') \in E \times E \; | \mathsf{det{-}precedes{-}adj(e, e')} \wedge e \neq e'\}.$$

4.1.4 Strukturbeschreibungen

Im folgenden wollen wir den Begriff der *Strukturbeschreibung* einer Äußerung für die Verwendung innerhalb des Formalismus der CD-Grammatik präzisieren. Im wesentlichen verstehen wir darunter eine Baumstruktur für jede Analyseebene. Allerdings fordern wir *nicht* unbedingt, daß alle Regeln der Grammatik erfüllt werden. Diejenigen Strukturbeschreibungen, die alle Regeln erfüllen, werden als *wohlgeformt* bezeichnet. Im Formalismus der CD-Grammatik werden folgende Forderungen an eine Strukturbeschreibung gestellt:

- Baumstruktur:
  Alle Kanten einer Analyseebene bilden eine Baumstruktur mit dem Wurzelknoten $\perp$.

- Lexikalische Eindeutigkeit:
  Für jede Wortform gibt es genau einen L-Knoten. Diese Forderung stellt sicher, daß einerseits alle Wortformen auch in der Strukturbeschreibung vorhanden sind und andererseits für alle Analyseebenen *dieselben* L-Knoten verwendet werden.

Folgende Definition formalisiert den Begriff der Strukturbeschreibung.

Definition 4.1.4 (Strukturbeschreibung) *Sei $\omega = w_1 \dots w_n \in \Sigma^*$ eine Äußerung und $G = (L, \perp, \mathcal{L}, R, C)$ mit $|R| = m$ eine CD-Grammatik. Eine Strukturbeschreibung für ω ist eine Menge von Bäumen $S = \{(V_\perp, E_i) | 1 \leq i \leq m\}$, wobei folgendes gilt:*

1. *Die Knotenmenge ist $V_\perp = V \cup \{\perp\}$ mit*

$$V = \{(\lambda_1, 1), \dots, (\lambda_n, n) | (\lambda_i, i) \in \mathcal{V}(\omega), 1 \leq i \leq n\}.$$

Jeder Wortform w_i wird genau ein L-Knoten (λ_i, i) zugeordnet.

2. *Die Kantenmenge ist*

$$E_i \subseteq \{(v, (r_i, l), v') \in V_\perp \times \mathcal{L} \times V\}.$$

Die Kanten innerhalb eines Dependenzbaumes dürfen nur Bezeichnungen der entsprechenden Analyseebene verwenden.

Im folgenden sei mit $E(S)$ die Menge aller Kanten einer Strukturbeschreibung S notiert, d.h.

$$E(S) = \bigcup_{(V,E) \in S} E.$$

Eine Kante bzw. ein Kantenpaar aus den Bäumen einer Strukturbeschreibung kann unäre bzw. binäre Regeln der Grammatik verletzen. Eine verletzte Bedingung wollen wir als *Konflikt* bezeichnen. Eine Kantenmenge ist *konfliktfrei*, falls keine Kante einen Konflikt verursacht.

Definition 4.1.5 (Konflikt) *Sei ω eine Äußerung und $G = (L, \perp, \mathcal{L}, R, C)$ eine CD-Grammatik. Sei E eine Kantenmenge mit $E \subseteq (\mathcal{V}(\omega) \cup \{\perp\}) \times \mathcal{L} \times \mathcal{V}(\omega)$. Die unäre Konfliktfunktion k_u und die binäre Konfliktfunktion k_b bzgl. E sind wie folgt definiert:*

$$k_u : E \rightarrow C \quad \textit{und} \quad k_b : E \times E \rightarrow C \quad \textit{mit}$$

$$\begin{array}{lcl} k_u(e) & = & \{{c_i}^u \in C | e \notin {c_i}^u\} \\ k_b(e, e') & = & \{{c_i}^b \in C | (e, e') \notin {c_i}^b \wedge e \neq e'\}. \end{array}$$

Ein Konflikt c ist ein Element aus der Menge aller Regelverletzungen (Konfliktmenge):

$$c \in (\bigcup_{e \in E} k_u(e) \cup \bigcup_{e, e' \in E} k_b(e, e')).$$

Eine Strukturbeschreibung S ist *wohlgeformt*, falls die Kanten der Strukturbeschreibung keinen Konflikt verursachen, d.h. die Konfliktmenge für $E(S)$ ist leer. Die formale Sprache einer CD-Grammatik ist die Menge der Äußerungen, für die eine wohlgeformte Strukturbeschreibung existiert. Unter einer *Dependenzanalyse* wird im folgenden die Menge aller wohlgeformten Strukturbeschreibungen verstanden.

4.1.5 Parsing von CD-Grammatiken

Die Aufgabe des CDG-Parsings ist die Ermittlung aller wohlgeformten Strukturbeschreibungen einer Äußerung. Dabei können im wesentlichen die Verfahren zum Lösen von *Constraint Satisfaction* Problemen verwendet werden. Im folgenden wird kurz formal beschrieben, wie aus einer Äußerung und einer CD-Grammatik eine Instanz des CSPs erzeugt werden kann.

Constraint Satisfaction Problem

Ein *Constraint Satisfaction* Problem besteht nach [Tsang 1993] aus einer Menge von Variablen, einer Menge von Wertebereichen (Domänen) für jede Variable, und einer Menge von Bedingungen (*Constraints*), welche auf den Werten der Variablen definiert werden. Formal ist ein CSP ein Tripel (Z, D, C_{CSP}) mit einer endlichen Menge von Variablen $Z = \{z_1, \ldots, z_n\}$, einer Menge von Wertebereichen $D = \{D_{z_1}, \ldots, D_{z_n}\}$ und einer Menge von Bedingungen[5]

$$C_{\text{CSP}} \subseteq \{c^{(i_1,\ldots,i_l)} | \quad c^{(i_1,\ldots,i_l)} \subseteq D_{z_{i_1}} \times \ldots \times D_{z_{i_l}}, \; 1 \leq i_j \leq n, 1 \leq j \leq l \leq n\}.$$

Gesucht ist eine Belegung der Variablen mit Werten aus ihren Wertebereichen, so daß alle Bedingungen erfüllt sind, d.h. eine Belegungsfunktion

$$\begin{array}{l} b : Z \to \bigcup_{1 \leq i \leq n} D_{z_i} \quad \text{mit } b(z_i) \in D_{z_i} \text{ und} \\ \forall c^{(i_1,\ldots,i_l)} \in C_{\text{CSP}} : (b(z_{i_1}), \ldots, b(z_{i_l})) \in c^{(i_1,\ldots,i_l)}. \end{array}$$

CDG-Parsing als CSP

Aus einer Äußerung ω und einer CD-Grammatik $G = (L, \bot, \mathcal{L}, R, C)$ läßt sich ein CSP wie folgt konstruieren: Für jede Analyseebene und für jeden L-Knoten wird eine Variable eingeführt. Die Menge der Variablen ist $Z = R \times \mathcal{V}(\omega)$. Die Wertebereiche der Variablen ergeben sich aus den möglichen Unterordnungsbeziehungen der Wortform innerhalb der entsprechenden Analyseebene, d.h. die Werte bestehen aus einer Beschriftung und einem dominierenden L-Knoten bzw. dem Wurzelknoten. Die Wertebereiche der Variablen sind
$D_{(r_i,v)} \subseteq L_i \times (\mathcal{V}(w) \cup \{\bot\})$ mit

$$\begin{array}{rl} D_{(r_i,(\lambda,k))} = & \{(l, (\lambda', k')) \quad | \quad k \neq k', (r_i, l) \in \mathcal{L}\} \cup \\ & \{(l, \bot) \quad | \quad (r_i, l) \in \mathcal{L}\}. \end{array}$$

Die unären und binären Bedingungen über den Werten der Variablen Z sind

$$\begin{array}{rl} C_{\text{CSP}} = & \{{c_1}^{(i)} | 1 \leq i \leq |Z|\} \cup \\ & \{{c_2}^{(i,j)} | 1 \leq i < j \leq |Z|\}. \end{array}$$

[5] Die Notation $c^{(i_1,\ldots,i_l)}$ besagt, daß die Bedingung c eine Stelligkeit von l hat, d.h. die Werte von l Variablen einschränkt. Die verwendeten Variablen sind z_{i_1} bis z_{i_l}.

Die Elemente ${c_1}^{(i)}$ und ${c_2}^{(i,j)}$ sind durch die Bedingungen der Grammatik bestimmt.[6] Sei $i \neq j$,$z_i = (r_p, v_p)$,$z_j = (r_q, v_q)$.

$$\begin{aligned} {c_1}^{(i)} &= \{(l_p, v'_p) | \forall_{1 \leq k \leq n_u} : \\ &\quad (v'_p, (r_p, l_p), v_p) \in {c_k}^u\} \\ {c_2}^{(i,j)} &= \{((l_p, v'_p), (l_q, v'_q)) | \forall_{1 \leq k \leq n_b} : \\ &\quad ((v'_p, (r_p, l_p), v_p), (v'_q, (r_q, l_q), v_q)) \in ({c_k}^b \cup {c_k}^{bT})\}. \end{aligned}$$

Gesucht ist eine partielle Belegungsfunktion der Variablen aus Z, die alle Bedingungen erfüllt. Die Belegungsfunktion muß partiell sein, da eine Lösung sich genau für eine Lesart, d.h für eine Variable pro Wortform und Analyseebene, entscheiden muß. Die Äußerung ω habe die Länge n. Die ausgewählten L-Knoten seien durch die Menge $\mathcal{V}'(\omega) = \{(\lambda_1, 1), \ldots, (\lambda_n, n)\} \subseteq \mathcal{V}(\omega)$ gegeben. Dieser Auswahlprozeß für eine Lesart von ω legt die Variablen Z' und die Wertebereiche D'_z für $z \in Z'$ fest. Die gesuchte Belegungsfunktion b' ist dann

$$b' : Z' \rightarrow \bigcup_{1 \leq l \leq |Z'|} D'_{z_l} \quad \text{mit } b'(z_i) \in D'_{z_i}.$$

Die Belegungsfunktion b', welche das CSP löst, stellt eine Analyse der Äußerung für eine CD-Grammatik dar. Wird noch gefordert, daß keine Zyklen im Dependenzgraphen vorliegen, liegt damit eine wohlgeformte Strukturbeschreibung vor. Die Abbildung 4.4 stellt das CSP für den Satz (6) „Der Clown lacht." als ungerichteten vollständigen Graphen dar.[7] Dieser Graph wird häufig als Constraintnetz oder Beschränkungsgraph bezeichnet. Markiert wurde diejenige Belegung, welche der Dependenzstruktur aus Abbildung 4.1 entspricht. Der Dependenzgraph aus Abbildung 4.2 ergibt sich aus allen Werten der Domänen, da jeder Wert einer Variablen einer Dependenzkante entspricht.

Lösungsverfahren

Eine Übersicht über Verfahren zum Lösen von *Constraint Satisfaction* Problemen findet sich z.B. in [Kumar 1992]. In der Arbeiten von [Maruyama 1990] und [Harper 1993] wird als Lösungsverfahren die Herstellung von Knoten- und Kantenkonsistenz verwendet. Der Zeitbedarf des Konsistenzverfahrens[8] angewendet auf eine CD-Grammatik liegt in der Größenordnung von $O(n^4)$, wobei n die Länge der Äußerung ist. Nimmt man z.B. an, daß jede Wortform im Mittel k Lesarten hat, ändert sich die Komplexitätsklasse zwar nicht ($O(k^4 n^4) = O(n^4)$), allerdings wächst die Anzahl der Möglichkeiten, eine Lesart für eine Wortform auszuwählen, exponentiell in n (k^n). In praktischen Fällen kann erst eine Reduktion der Variablen um einen konstanten Faktor ein akzeptables Zeitverhal-

---

[6] Sei $A \subseteq B \times B$ eine binäre Relation. Mit A^T sei die gespiegelte Relation definiert mit $A^T = \{(b_2, b_1) | (b_1, b_2) \in A\}$.

[7] Es wird hier nur eine Analyseebene, die Syntax, betrachtet. Außerdem liegt der Analyse die stark vereinfachte Annahme zugrunde, daß jede Wortform nur genau eine Lesart besitzt.

[8] Kantenkonsistenz kann im Falle eines vollständigen Beschränkungsgraphens mit e Variablen und a möglichen Werten in $O(e^2 \cdot a^2)$ Schritten hergestellt werden ([Schröder 1995]).

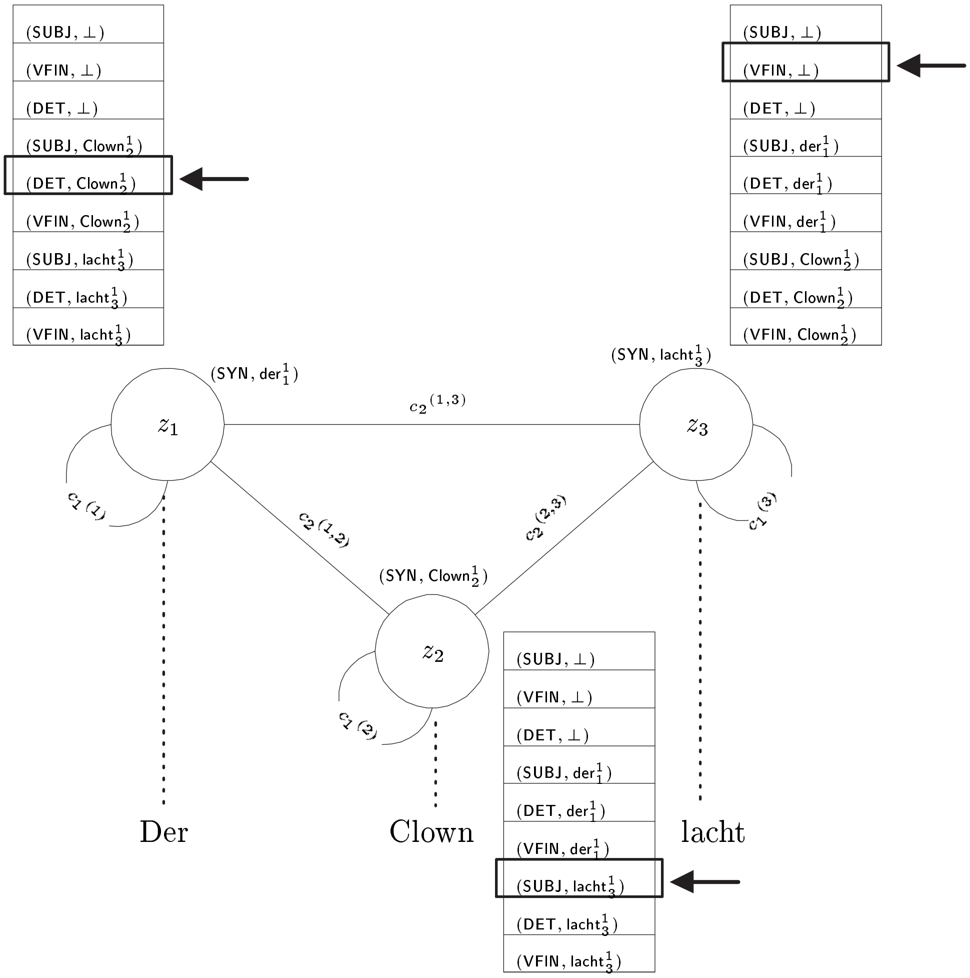

Abbildung 4.4: Initiales Constraintnetz.

ten sicherstellen. [9] Im allgemeinen reicht die Herstellung von Kantenkonsistenz jedoch nicht aus, um alle Lösungen zu finden, so daß eine Kombination aus Konsistenz- und Suchverfahren notwendig ist, wobei die Suche exponentiell viel Zeit benötigen kann ([Kumar 1992],[Schröder 1995]). Ein wichtiges Ziel ist es deshalb, viele Werte aus den Wertebereichen der Variablen durch Konsistenzmaßnahmen zu streichen, um den verbleibenden Suchraum möglichst klein zu gestalten.

[9] Das Problem der lexikalischen Mehrdeutigkeit einer CD-Grammatik wurde in [Foth 1999a] behandelt. Insbesondere sind nicht alle Lesarten einer Wortform für alle Analyseebenen relevant, d.h. die Regeln der Grammatik benutzen nicht alle Merkmale. Die Semantikebene z.B. benötigt keine Kasus-, Numerus- und Genusinformation, während z.B. die Syntaxebene keine Information über semantische Rollen benötigt. Die Folge ist, daß die Anzahl der Variablen des CSPs deutlich reduziert werden kann.

4.2 Bewertete CD-Grammatiken

Die Erweiterung des Formalismus der CD-Grammatik um *bewertete* Regeln wurde erstmals von [Menzel 1995] vorgeschlagen. Die Motivation resultiert aus dem Ergebnis, daß die binäre Natur der Regeln nicht ausreicht, um für die robuste Analyse natürlicher Sprache verwendet werden zu können. Ebenso haben praktische Versuche in [Harper 1995] gezeigt, daß viele Analysen nicht vollständig disambiguiert werden können. Dies kann meistens nicht durch die Hinzunahme weiterer Regeln gelöst werden, da dadurch eventuell andere Analysen nicht mehr möglich sind (Überspezifikation). Im Abschnitt 4.2.1 wird die Definition einer bewerteten CD-Grammatik angegeben.

In bewerteten CD-Grammatiken wird die Forderung, alle Regeln zu erfüllen, aufgegeben. Stattdessen wird die Menge aller potentiellen Strukturbeschreibungen mit Hilfe einer *Kostenfunktion* geordnet. In diesem Sinne ist es nicht unbedingt mehr gerechtfertigt, vom eliminativen Parsing zu sprechen, da keine Strukturen zurückgewiesen werden, sondern nur bestimmte Strukturen vor anderen *präferiert* werden. Das Verfahren wird deshalb auch als *präferenzbasiertes* Parsing bezeichnet. Gewöhnlich wird dann diejenige Struktur, die die geringsten Kosten verursacht, als die beste Strukturbeschreibung der Äußerung angesehen. Das Parsingproblem kann damit als ein allgemeines Optimierungsproblem betrachtet werden. Die Kostenfunktion wird im Abschnitt 4.2.2 eingeführt.

Im Laufe der Zeit sind verschiedene Lösungsverfahren für bewertete CD-Grammatiken vorgeschlagen worden, welche in Abschnitt 4.2.3 kurz diskutiert werden.

4.2.1 Bestrafung von Regelverletzungen

Die Definition einer CD-Grammatik wird dahingehend erweitert, daß jede Regel mit einer *Strafe* annotiert wird. Die Strafe wird erst wirksam, wenn eine Strukturbeschreibung Regeln der Grammatik verletzt.

Definition 4.2.1 (Bewertete CD-Grammatik) *Eine bewertete CD-Grammatik G ist ein Tupel $(L, \perp, \mathcal{L}, R, C, \Phi)$, wobei $(L, \perp, \mathcal{L}, R, C)$ eine CD-Grammatik und Φ eine Bewertungsfunktion ist, welche jeder Bedingung aus C eine Bewertung aus $[0 \ldots 1]$ zuordnet, d.h.*

$$\Phi : C \to [0 \ldots 1].$$

Der Wert $\Phi(c)$ gibt die Strafe für die Verletzung der Bedingung c an. Liegen mehrere Regelverletzungen vor, ergibt sich die Gesamtstrafe multiplikativ aus den Einzelstrafen. Die multiplikative Verknüpfung, die in Abschnitt 4.2.2 definiert wird, bestimmt implizit die Eigenschaften des Strafwertes $\Phi(c)$. Je kleiner der Wert, desto größer ist die Strafe. Der Wert Null steht für die maximale Strafe. Eine mit Null bewertete Bedingung kann nicht verletzt werden und wird als *harte* Bedingung bezeichnet. *Weiche* Bedingungen sind solche, für die $\Phi(c) > 0$ gilt. Eine mit Eins bewertete Bedingung kann ohne eine Strafe zu verursachen verletzt werden und ist damit irrelevant.

Beispiel

Modellierungstechniken für bewertete CD-Grammatiken finden sich in [Schröder 1999]. Als klassisches Beispiel für eine präferenzbasierte Regel wird häufig das Phänomen der Topikalisierung betrachtet: In dem Satz

(7) „Das Kind sucht seine Mutter."

kann sowohl „Kind" als auch „Mutter" das Subjekt des Satzes sein. Nur wenn zusätzliche Information verfügbar ist, kann die strukturelle Mehrdeutigkeit aufgelöst werden. Ist dies nicht der Fall, wird grundsätzlich die nicht topikalisierte Struktur bevorzugt. Folgende Regel modelliert diese Heuristik:

$$\begin{aligned}&\mathsf{X.level} = \mathsf{SYN} \wedge \mathsf{X.label} = \mathsf{SUBJ} \wedge \neg\mathsf{root}(\mathsf{X}\uparrow) \rightarrow \\ &\qquad \mathsf{X}\downarrow\mathsf{pos} < \mathsf{X}\uparrow\mathsf{pos}\end{aligned} \qquad ({c_2}^u)$$

Damit eine Struktur auch für fehlerhafte Sätze verfügbar ist, werden weiche Regeln verwendet, z.B. für eine verletzte Kongruenzforderung:

$$\begin{aligned}&\mathsf{X.level} = \mathsf{SYN} \wedge \mathsf{X.label} = \mathsf{DET} \wedge \neg\mathsf{root}(\mathsf{X}\uparrow) \rightarrow \\ &\qquad \mathsf{X}\downarrow\mathsf{case} = \mathsf{X}\uparrow\mathsf{case} \wedge \mathsf{X}\downarrow\mathsf{num} = \mathsf{X}\uparrow\mathsf{num}\wedge \\ &\qquad \mathsf{X}\downarrow\mathsf{gender} = \mathsf{X}\uparrow\mathsf{gender}\end{aligned} \qquad ({c_3}^u)$$

Eine Verletzung der Kongruenzforderung wird als schwerwiegender angesehen als die Verwendung von topikalisierten Konstituenten, was durch folgende Strafen modelliert wird:

Regel $c \in C$	Strafe $\Phi(c)$
${c_2}^u$	0.9
${c_3}^u$	0.02

Die Strafen bestimmen, welche Regeln im Fehlerfall wichtiger sind als andere, d.h. welche Regelverletzungen als schwerwiegender angesehen werden. In dem Satz

(8) „Den Abwehrspieler sucht die Schiedsrichterin."

kann entweder ${c_2}^u$ oder ${c_3}^u$ verletzt werden.[10] Da die Strafe für ${c_2}^u$ geringer ist, wird die topikalisierte Lesart bevorzugt. Andererseits wird in

(9) * „Die Stürmer sucht den Ball."

nur ${c_3}^u$ verletzt, da die alternative Lesart sowohl die Präferenz- als auch die Kongruenzregel verletzen würde.

[10] Für die Beispielsätze (8) und (9) werden weitere harte Regeln angenommen: Einerseits muß das Subjekt im Nominativ stehen und mit dem finiten Verb in Numerus und Person kongruieren, andererseits muß das direkte Objekt im Akkusativ stehen.

4.2.2 Kosten einer Strukturbeschreibung

Eine Strukturbeschreibung kann nach Definition 4.1.5 Konflikte verursachen, indem unäre bzw. binäre Bedingungen verletzt werden. Für bewertete CD-Grammatiken läßt sich ein Maß angeben, wodurch verschiedene Strukturen aufgrund der durch sie verursachten Konflikte miteinander verglichen werden können. Dieses Maß wollen wir als *Kosten* einer Strukturbeschreibung bezeichnen.

Definition 4.2.2 (Kosten) *Sei* $E_{\text{all}} = (\mathcal{V}(\omega) \cup \{\perp\}) \times \mathcal{L} \times \mathcal{V}(\omega)$ *die Menge aller Kanten aus den möglichen Dependenzgraphen für eine Äußerung* ω. *Die Bewertungsfunktion*[11] *einer Kantenmenge* $E \subseteq E_{\text{all}}$ *ist*

$$\Phi(E) = \prod_{e \in E} \prod_{c \in k_u(e)} \Phi(c) \quad \cdot \quad \prod_{(e,e') \in E \times E} \prod_{c \in k_b(e,e')} \Phi(c).$$

Als Kosten von E *bezeichnen wir die Funktion* $k : 2^{E_{\text{all}}} \to [0 \ldots 1]$ *mit*

$$k(E) = \begin{cases} 0, & \textit{wenn } E \textit{ konfliktfrei ist.} \\ 1 - \Phi(E) & \textit{sonst.} \end{cases}$$

Die Kostenfunktion hat folgende Eigenschaften:

- Wird keine Bedingung verletzt, so sind die Kosten minimal.
- Wird eine harte Bedingung verletzt, so sind die Kosten immer maximal.[12]
- Die Kosten einer Kantenmenge wachsen monoton mit der Anzahl der Kanten in dieser Menge,

$$E \subseteq E' \Rightarrow k(E) < k(E').$$

Insbesondere die letzte Eigenschaft erweist sich für die Realisierung eines Lösungsverfahrens als besonders vorteilhaft.

Eine Dependenzanalyse einer Äußerung besteht aus denjenigen Strukturbeschreibungen S, welche minimale Kosten verursachen, d.h.

$$S_{\text{all}} = \{S \quad | \quad \arg \min_{S} \; k(E(S))\}.$$

Allerdings werden Strukturbeschreibungen mit den maximalen Kosten eins grundsätzlich nicht zugelassen, d.h. es dürfen keine harten Bedingungen verletzt werden. Strukturbeschreibungen mit minimalen Kosten wollen wir als *optimale* Strukturbeschreibungen bezeichnen. Wird nur der Wert Null als Strafe verwendet, ist eine bewertete CD-Grammatik äquivalent zu der in Abschnitt 4.1 definierten CD-Grammatik, d.h. optimale und wohlgeformte Strukturbeschreibungen sind identisch.

[11] Eine ähnliche Darstellung findet sich in [Menzel 1998d].

[12] In der Arbeit von [Foth 1999b] wird eine verallgemeinerte Bewertungsfunktion vorgeschlagen, welche die Anzahl der verletzten harten Bedingungen berücksichtigt.

4.2.3 Parsing als Optimierungsproblem

Eine bewertete CD-Grammatik kann ganz analog zum Abschnitt 4.1.5 als CSP formalisiert werden. Der Unterschied liegt in der gesuchten Belegungsfunktion b'. Das klassische CSP fordert von der Belegungsfunktion, daß alle Bedingungen erfüllt werden. Im bewerteten Fall wird gefordert, daß es keine Belegungsfunktion b'' gibt, die geringere Kosten verursacht. Die Problemklasse wird auch als *Partial Constraint Satisfaction Problem* (PCSP, [Freuder 1992]) bezeichnet.

Konsistenzverfahren, die aufgrund der lokalen Natur der Bedingungen Werte aus den Wertebereichen ausschließen, können nur noch für harte Bedingungen verwendet werden. In [Menzel 1998a] wird eine Heuristik vorgeschlagen, die angibt, welche Werte aus welchen Domänen auch dann gelöscht werden dürfen, wenn die beteiligten Werte nicht vollständig unverträglich sind, d.h. die Werte verletzen weiche Bedingungen. Dieses Selektionsverfahren kann weder vollständig noch korrekt sein, ermöglicht allerdings eine Art Konsistenz für weiche Bedingungen.

Eine Übersicht über weitere Verfahren liefert [Menzel 1998d]. Insbesondere wird dort ein Suchverfahren (*branch and bound*) vorgestellt, welches aufgrund der Monotonie der Kostenfunktion große Teile des Suchraums ausschließen kann. Dieses Verfahren ist theoretisch sowohl vollständig als auch korrekt. Da allerdings exponentiell viel Zeit und Platz benötigt werden kann, wird in der praktischen Realisierung nur eine begrenzte Anzahl von alternativen Suchzuständen in einer *Suchagenda* gespeichert, die im Falle einer fehlgelaufenen Suche eventuell alle besucht werden müssen.

Da die Problemklasse des CSPs NP-vollständig ist, kann kaum erwartet werden, daß es ein vollständiges und korrektes Verfahren gibt, welches das Problem in polynomiell viel Zeit lösen kann. In [Foth 1999b] wird ein heuristisches Verfahren vorgestellt, welches auf dem Reparaturansatz basiert, d.h. ausgehend von einer initialen, suboptimalen Lösung, werden die Konfliktstellen sukzessiv durch die Wahl einer veränderten Belegungsfunktion beseitigt bzw. es wird versucht, die Kosten zu reduzieren. Dabei sind, im Unterschied zu anderen Optimierungsverfahren (z.B. *Hill Climbing*), auch Veränderungen möglich, welche eine schlechtere als die bisher beste Belegung erzwingen. Da die optimale Lösung in der Regel ebenfalls Konflikte aufweist, müssen spezielle Maßnahmen ergriffen werden, um eine Terminierung zu gewährleisten. Als erfolgreich hat sich eine Kombination aus der Tabu-Suche und der Ausnutzung von lokalen Verträglichkeitseigenschaften, ähnlich dem Selektionsverfahren, erwiesen. Das Verfahren ist zumindestens theoretisch nicht vollständig und korrekt, findet aber im Unterschied zu dem Suchverfahren *branch and bound* jederzeit eine suboptimale Lösung, die sich mit fortschreitender Laufzeit stetig verbessert. Es erfüllt die Anforderungen eines *anytime* Verfahrens ([Menzel 1994]). Die Ergebnisse in [Foth 1999b] belegen experimentell eine deutliche Effizienzverbesserung ab einer gewissen Problemgröße (ca. 200 Variablen im CSP). Es wird in 90 Prozent der Fälle eine korrekte Strukturbeschreibung ermittelt.

In dieser Arbeit wird das Suchverfahren *branch and bound* verwendet, da für das Anwendungsgebiet der Fehlerdiagnose eine korrekte Analyse notwendig ist. Allerdings müssen die Analysen der Sätze, welche die Grenzen der Suchagenda sprengen, grundsätzlich als fragwürdig angesehen werden. Obwohl in vielen Fällen trotzdem die richtige Analyse geliefert wird, ist das Risiko einer völlig unverständlichen Fehlererklärung zu groß.[13]

4.3 Fehlerdiagnose

Der Vorschlag, bewertete CD-Grammatiken für die Realisierung von Fremdsprachenlernsystemen zu verwenden, findet sich in [Menzel 1998b] und [Menzel 1998e]. Dort wird eine Klassifizierung der Grammatikregeln in drei Klassen vorgeschlagen, die im wesentlichen harten Regeln, Präferenzregeln und weichen Wohlgeformtheitsbedingungen entsprechen. Eine Nichtbeachtung einer Regel der letzteren Klasse wird dabei als Fehler interpretiert. Harte Regeln sind, obwohl theoretisch nicht notwendig, in der Praxis für ein akzeptables Zeitverhalten des Parsingverfahrens erforderlich. Sie reduzieren die in Frage kommenden Strukturbeschreibungen auf ein vernünftiges Maß, allerdings zu dem Preis, daß es möglicherweise fehlerhafte Eingabesätze geben kann, die nicht mehr analysiert werden können.[14] Präferenzregeln kompensieren das Problem zwischen Überspezifikation einerseits und fehlender Disambiguierungsinformation andererseits durch die Verwendung von weichen, jedoch nicht als Fehler interpretierten Bedingungen. In Abschnitt 4.3.1 wird der aus dieser Einteilung resultierende Begriff der Diagnose[15] definiert. Einige Beispiele für die Modellierung von Regeln werden im Abschnitt 4.3.2 behandelt. Die Generierung einer verbalen Fehlererklärung kann ähnliche Mechanismen wie das Verfahren der Fehlersimulation verwenden. Dies ist Gegenstand des Abschnittes 4.3.3.

4.3.1 Definition des Diagnosebegriffs

Für den Begriff der Diagnose im Formalismus der CD-Grammatik teilen wir die Menge der Regeln in zwei disjunkte Teilmengen auf, wobei die erste alle harten und präferenzbasierten und die zweite alle für die Diagnose relevanten weichen Regeln enthält. Weiterhin benötigen wir für die Differenzierung von verschiedenen Lesarten und für die Generierung von Fehlererklärungen die beteiligten Kanten in einem durch eine Regel verursachten Konflikt (siehe Definition 4.1.5). Diese Information entspricht im Verfahren der Fehlersimulation der Wertebelegung an den Ports.

[13] In der Implementation des integrativen Verfahrens wird z.Zt. noch keine Methode für die Überprüfung einer übergelaufenen Agenda zur Verfügung gestellt.

[14] Allerdings kann man leicht eine Grammatik so schreiben, daß alles akzeptiert wird und trotzdem harte Regeln verwendet werden. Dies gelingt, indem jede Wortform immer auch unter dem Wurzelknoten angeordnet werden kann. Diese Technik wird in Abschnitt 4.3.2 vorgestellt.

[15] Der formale Diagnosebegriff im CDG-Parsing unterscheidet sich von der Verwendung im Verfahren der Fehlersimulation.

Definition 4.3.1 (Diagnose einer Strukturbeschreibung)
Sei $C = C_{\text{pref}} \cup C_{\text{diag}}$ die Regelmenge einer bewerteten CD-Grammatik mit $C_{\text{pref}} \cap C_{\text{diag}} = \emptyset$. Sei S_{all} die Dependenzanalyse für eine Äußerung ω. Eine Diagnose für eine Strukturbeschreibung $S \in S_{\text{all}}$ ist

$$\begin{aligned}\text{diagnosis}(S) = \textstyle\bigcup_{e,e' \in E(S)} \quad & \{(c,e,e) | c \in k_u(e) \cap C_{\text{diag}}\} \cup \\ & \{(c,e,e') | c \in k_b(e,e') \cap C_{\text{diag}}\}.\end{aligned}$$

Aufbauend auf dieser Definition können wir festlegen, was eine *korrekte* Äußerung in einer bewerteten CD-Grammatik ist:

Definition 4.3.2 (Korrekte Äußerung) *Eine Äußerung ω ist bzgl. einer bewerteten CD-Grammatik genau dann korrekt, wenn es in der Dependenzanalyse S_{all} eine Strukturbeschreibung $S \in S_{\text{all}}$ gibt, für die* $\text{diagnosis}(S) = \emptyset$ *gilt.*

Eine Dependenzanalyse kann mehr als eine Diagnose besitzen, wobei die Anzahl der verletzten Regeln für jede Diagnose unterschiedlich sein kann. Dies resultiert einerseits aus der verwendeten Kostenfunktion (siehe Definition 4.2.2), andererseits aus der Partitionierung der Regelmenge. Daher ist auch für die Fehlerdiagnose mit CDG-Parsing eine ähnliche Fehlerselektionsstrategie wie für das Verfahren der Fehlersimulation notwendig.

4.3.2 Modellierung

Im folgenden werden einige Beispiele für die Modellierung ausgewählter linguistischer Phänomene diskutiert. Die Modellierungstechniken basieren auf [Schröder 1999].

Präferenzen

Nicht alle verletzten Bedingungen sind Fehler im Sinne der Grammatik. Sie bezeichnen vielmehr eine Präferenz der Unterordnung einer Wortform unter einer anderen. Die Präferenzregel ${c_3}^u$ für die Behandlung der Topikalisierung haben wir schon in Abschnitt 4.2.1 behandelt. Andere typische Präferenzbedingungen, die nicht zuletzt auch der Beschleunigung der Analyse dienen, bevorzugen Strukturen, in denen die untergeordnete Wortform möglichst dicht bei der modifizierten Wortform steht:

„Ein Artikel befindet sich *möglichst in der Nähe* des modifizierten Substantivs.“

$$\begin{aligned}&\mathsf{X.level} = \mathsf{SYN} \wedge \mathsf{X.label} = \mathsf{DET} \wedge \neg\mathsf{root}(\mathsf{X}\uparrow) \rightarrow \\ &\qquad \mathsf{abs}(\mathsf{X}\uparrow \mathsf{pos} - \mathsf{X}\downarrow \mathsf{pos}) = 1\end{aligned} \qquad ({c_4}^u)$$

Die Bewertungsfunktion muß dabei dynamische Strafen unterstützen, d.h. die Strafe kann erst berechnet werden, wenn die beteiligten Dependenzkanten bekannt sind. Diese Information liegt im Falle eines Konflikts aber immer vor, so daß die Definition der bewerteten CD-Grammatik leicht erweitert werden kann. Folgende Bewertungsfunktion wird verwendet:

$$\Phi({c_4}^u) = 0.9 + \frac{1}{10 \cdot \text{abs}(\mathsf{X}\uparrow \mathsf{pos} - \mathsf{X}\downarrow \mathsf{pos})}$$

Eine Verletzung der Regeln ${c_3}^u$ und ${c_4}^u$ wird nicht als ein Fehler interpretiert, sondern löst das Problem der Überspezifikation durch die Verwendung von verletzbaren Bedingungen. Zum Beispiel bewirkt die Regel ${c_4}^u$, daß in dem Satz

(10) * „Ein der Clown lacht.“

der näherstehende Artikel „der“ dem Substantiv „Clown“ untergeordnet wird.

Partielles Parsing

Der erste Artikel in Satz (10) sollte als überflüssig angesehen werden. Um für diese Phänomene überhaupt Strukturbeschreibungen generieren zu können, wird eine Regel benötigt, die beliebige Konstituenten unter ein ausgezeichnetes Element des Dependenzbaumes unterordnet. Das einzige Element, welches nicht auf der Oberfläche realisiert ist und daher am besten geeignet ist, ist der Wurzelknoten. Als Dependenzrelation für die Markierung von überflüssigen Konstituenten wird das Symbol IGNORE verwendet.[16]

„Jede Wortform kann als überflüssige Konstituente dem Wurzelknoten untergeordnet werden.“

$$\mathsf{X.level} = \mathsf{SYN} \wedge \mathsf{X.label} = \mathsf{IGNORE} \rightarrow \mathsf{root(X\uparrow)} \qquad ({c_5}^u)$$

„Überflüssige Konstituenten sind unerwünscht.“

$$\mathsf{X.level} = \mathsf{SYN} \rightarrow \neg(\mathsf{X.label} = \mathsf{IGNORE}) \qquad ({c_6}^u)$$

Die Regel ${c_5}^u$ kann als harte Regel verwendet werden, während ${c_6}^u$ als weiche Regel mit einer harten Bestrafung verwendet werden muß. Im Vergleich mit anderen Regelstrafen muß $\Phi({c_6}^u)$ wesentlich kleiner sein, da sonst aufgrund des multiplikativen Charakters der Kostenfunktion Worte mit grammatischen Fehlern fälschlicherweise als überflüssige Wortformen angesehen werden könnten.[17]

Regel $c \in C$	Strafe $\Phi(c)$
${c_5}^u$	0.0
${c_6}^u$	0.00001

Existenzforderungen

Existenzforderungen werden benötigt, weil einige Wortformen obligatorische Komplemente fordern. Ein Substantiv benötigt einen untergeordneten Artikel[18], eine Präpositionalgruppe benötigt eine Präposition, Verben benötigen häufig ein Subjekt, und je nach syntaktischer Valenz z.B. ein direktes oder ein indirektes Objekt. Für den Formalismus der CD-Grammatik stellen Existenzforderungen ein Problem dar, da jede Regel immer allquantifiziert ist. Der

[16] Die Modellierung von überflüssigen Konstituenten benötigt weitere Regeln. Für eine exakte Modellierung sei auf den Anhang A.3 auf Seite 106 verwiesen.

[17] Das Gesamtverhalten der Kombination von Regelverletzung und den zugehörigen Kosten ist schwer absehbar. Modellierungsfehler können nur in den seltensten Fällen vollständig ausgeschlossen werden.

[18] Substantive mit Nullartikel werden für das folgende Beispiel nicht betrachtet.

Modellierungstrick besteht darin, für jede Dependenzrelation $R(A, B)$ der Syntaxebene, in der die Wortform A obligatorisch ist, eine weitere Analyseebene einzuführen, in der die inverse Relation $R^{-1}(B', A')$ modelliert wird. Zum Beispiel verbindet die Kante mit der Kantenbeschriftung DET in der Syntaxebene einen Artikel mit einem Substantiv. Auf der Hilfsebene verbindet eine Kante DET^{-1} ein Substantiv mit einem Artikel. Wenn auf beiden Ebenen derselbe L-Knoten für das Substantiv verwendet wird, so muß auch derselbe L-Knoten für den Artikel verwendet werden. Die folgenden drei Regeln modellieren diese Technik.

„Substantive modifizieren Artikel."

$$\begin{aligned}&\mathsf{X.level} = \mathsf{SYN\text{-}DET} \wedge \neg\mathsf{root}(\mathsf{X}\uparrow) \rightarrow\\ &\quad \mathsf{X}\downarrow \mathsf{cat} = \mathsf{n} \wedge \mathsf{X}\uparrow \mathsf{cat} = \mathsf{det}\end{aligned} \qquad ({c_7}^u)$$

„Substantive dürfen nicht unter dem Wurzelknoten angeordnet werden."

$$\begin{aligned}&\mathsf{X.level} = \mathsf{SYN\text{-}DET} \rightarrow \mathsf{root}(\mathsf{X}\uparrow) \Leftrightarrow\\ &\quad \neg\mathsf{X}\downarrow \mathsf{cat} = \mathsf{n}\end{aligned} \qquad ({c_8}^u)$$

„Jede Substantiv-Artikel-Kante der Hilfsebene entspricht einer Artikel-Substantiv-Kante der Syntaxebene."

$$\begin{aligned}&\mathsf{X.level} = \mathsf{SYN} \wedge \mathsf{Y.level} = \mathsf{SYN\text{-}DET} \wedge \mathsf{X}\downarrow = \mathsf{Y}\uparrow \rightarrow\\ &\quad \mathsf{X.label} = \mathsf{DET} \wedge \mathsf{X}\uparrow = \mathsf{Y}\downarrow\end{aligned} \qquad ({c_2}^b)$$

Die Regel ${c_2}^b$ wird auch als *Mapping*-Regel bezeichnet, da sie die Kanten zweier Analyseebenen miteinander in Beziehung setzt. Um ein fehlendes Komplement diagnostizieren zu können, reicht es, nur die Regel ${c_8}^u$ mit einer von Null verschiedenen Strafe zu versehen.

Regel $c \in C$	Strafe $\Phi(c)$
${c_7}^u$	0.0
${c_8}^u$	0.01
${c_2}^b$	0.0

4.3.3 Fehlererklärung

Eine Dependenzanalyse in einer bewerteten CD-Grammatik kann mehrere optimale Strukturbeschreibungen liefern. Für jede dieser Beschreibungen ist eine Diagnose verfügbar. Eine Diagnose setzt sich aus den verletzten Regeln und den beteiligten Dependenzkanten zusammen.

Fehlerverbalisierung

Für die Generierung einer verbalen Erklärung kann z.B. ein musterbasierter Ansatz ähnlich dem der Fehlersimulation verwendet werden. Die allgemeine Form wäre z.B. $<$*Aktivierungsmuster*$>$ $\Rightarrow$ $<$*Textmuster*$>$. Als Aktivierungsmuster können logische Formeln verwendet werden, die auf einer Strukturbeschreibung und dem zugehörigen Diagnoseergebnis formuliert werden. Das Textmuster kann Übersetzungsfunktionen für die verwendeten Symbole, die sich z.B.

aus der Merkmalsstruktur der Wortformen ergeben, enthalten. Als einfaches Beispiel sei eine Verletzung der Kongruenzforderung ${c_3}^u$ mit dem zugewiesenen Bezeichner det-n-agr angenommen. Folgendes Erklärungsschema könnte dann verwendet werden:

$\mathsf{violated}(\mathsf{det\text{-}n\text{-}agr}, \mathsf{X}) \Rightarrow$ „Der Artikel $\mathsf{X} \downarrow$ stimmt nicht mit dem Substantiv $\mathsf{X} \uparrow$ im Kasus, Numerus oder Genus überein."

Hier wird schon deutlich, daß *zu viele* Wohlgeformtheitsforderungen in einer Regel kodiert wurden. Eine Aufspaltung der Regel ${c_3}^u$ in drei separate Regeln ist daher angebracht.

Auch für das CDG-Parsing können Diagnosen verdichtet werden. Das allgemeine Verdichtungsschema kann z.B. die Form <*Verdichtungsmuster*> ⇒ <*Komprimierter Ausdruck*> haben. Das Verdichtungsmuster wird analog zum Aktivierungsmuster als logischer Ausdruck realisiert. Das folgende Beispiel verdichtet mehrere Kongruenzverletzungen im Kasus, wobei ein ähnlicher Verdichtungsausdruck wie in der Fehlersimulation verwendet wurde.

$$\begin{aligned}&\mathsf{violated}(\mathsf{det\text{-}n\text{-}agr\text{-}case}, \mathsf{X}_1) \wedge \ldots \wedge \mathsf{violated}(\mathsf{det\text{-}n\text{-}agr\text{-}case}, \mathsf{X}_\mathsf{n}) \wedge\\ &\mathsf{X}_1 \uparrow = \mathsf{X}_2 \uparrow = \ldots = \mathsf{X}_\mathsf{n} \uparrow \Rightarrow\\ &\mathsf{and\text{-}up\text{-}agree}(\{\mathsf{X}_1, \ldots, \mathsf{X}_\mathsf{n}\}, \mathsf{X}_1, \mathsf{case}).\end{aligned}$$

Ein Erklärungsschema ist dann z.B.

$\mathsf{and\text{-}up\text{-}agree}(\{\mathsf{X}_1, \ldots, \mathsf{X}_\mathsf{n}\}, \mathsf{Y}, \mathsf{m}) \Rightarrow$ „$\mathsf{X}_1 \downarrow$ und … und $\mathsf{X}_\mathsf{n} \downarrow$ stimmen mit $\mathsf{Y} \uparrow$ nicht im m überein."

Dabei können natürlich nicht nur Kongruenzregeln verdichtet werden. In dem Satz

(11) * „Er legt."

können z.B. die verletzten Existenzforderungen aufgrund der syntaktischen Valenz des Verbs verdichtet werden. Auch für die Fehlerdiagnose mit CDG-Parsing gilt, daß es immer konkurrierende Verdichtungsmöglichkeiten geben kann.

Fehlerselektion

Die Aufgabe der Fehlerselektion ist es, einerseits eine Ordnung auf den verfügbaren Diagnosen aufzustellen und andererseits nur eine Teilmenge der optimalen Strukturbeschreibungen zu betrachten, d.h. eine Reduktion der Diagnosemenge vorzunehmen.

Es gibt im Unterschied zur Fehlersimulation eine einfache Möglichkeit, die Diagnosemenge zu reduzieren. Dies kann erreicht werden, indem alle weichen Regeln unterschiedliche Strafen aufweisen. Dadurch ist es unwahrscheinlich, daß verschiedene Regelverletzungen dieselben Kosten verursachen. Das Fehlen alternativer Diagnosen kann aber auch als Nachteil interpretiert werden: In dem Satz

(12) * „Das Buch liegt auf den Tisch."

wäre dann nur eine Diagnose verfügbar, wie z.B.

(E6) „Das Verb liegen fordert eine lokale Ortsangabe.“

Weitere Erklärungen werden ignoriert:

(E7) „Der Artikel *den* und das Substantiv *Tisch* kongruieren nicht im Kasus.“

(E8) „Der Artikel steht im Akkusativ, während das Substantiv im Dativ steht.“

(E9) „Der Artikel und das Substantiv stimmen nicht im Numerus überein.“

Werden allerdings gleiche Strafen für die betroffenen Regeln verwendet, wären alle alternativen Diagnosen verfügbar. Damit können prinzipiell alle Fehler, die das Verfahren der Fehlersimulation im Simulationsmodus der Regelfehlersimulation generieren kann, auch mit dem Verfahren des CDG-Parsings diagnostiziert werden.

Die Selektionsstrategien der Fehlersimulation können zum Teil auch für das CDG-Parsing verwendet werden. Auch das Vorliegen einer Strukturbeschreibung gestattet noch weitere Selektionskriterien. Im folgenden werden einige Kriterien vorgeschlagen.

- Die Diagnosen werden nach der Anzahl der verletzten Regeln geordnet.
- Einige Regelverletzungen werden vor anderen präferiert. Dieses Ordnungskriterium ist unabhängig von den Strafwerten, da gleiche Strafen erst alternative Diagnosen ermöglichen. Die Ordnung kann auf den Regelbezeichnern definiert werden.
- Die Verdichtungsmöglichkeiten, die Merkmalspräferenz, die Lesartenpräferenz und die Einbeziehung des Studentenmodells können analog zur Fehlersimulation als Kriterien übernommen werden (vgl. Abschnitt 3.3.7).
- Das Kriterium der syntaktischen Tiefe kann für vorliegende Strukturbeschreibungen wesentlich detaillierter formuliert werden, indem logische Formeln über den Bestandteilen der Strukturbeschreibungen eine Ordnung definieren. Als Beispiel können folgende Strukturpräferenzen verwendet werden:
  - Bevorzuge Strukturen, in denen das Subjekt vor dem finiten Verb steht.
  - Bevorzuge Strukturen, die keine Projektionsverletzungen aufweisen.

Kapitel 5

Integration von CDG-Parsing und Fehlersimulation

In diesem Kapitel wird ein integrativer Verarbeitungsansatz für natürlichsprachliche Äußerungen vorgestellt, welcher die Vorteile der Fehlersimulation und des CDG-Parsings zu kombinieren versucht. In Abschnitt 5.1 wird dieser Ansatz motiviert, indem die derzeitigen Probleme des CDG-Parsings genauer untersucht werden. Es wird erläutert, inwiefern ein integrativer Ansatz einige der Probleme lösen kann. In Abschnitt 5.2 wird ein sequentielles Verarbeitungsmodell vorgestellt. Dazu wird ein Formalismus benötigt, der es gestattet, aus den Strukturbeschreibungen des CDG-Parsings Fehlersimulationsmodelle zu konstruieren. In Abschnitt 5.3 wird der entwickelte Formalismus erläutert und anhand ausgewählter linguistischer Phänomene verdeutlicht. In Abschnitt 5.4 werden die Ergebnisse sowohl in Bezug auf das Zeitverhalten als auch auf die Diagnosequalität vorgestellt. Für die Evaluation des Verfahrens wird eine Beispielgrammatik und ein Testkorpus mit 84 korrekten und 81 fehlerhaften Sätzen verwendet. Das im Rahmen dieser Arbeit entwickelte Programm für die Validierung des integrativen Ansatzes wird in Abschnitt 5.5 behandelt.

5.1 Motivation

Das CDG-Parsing gestattet die Überprüfung von Merkmalsbelegungen, jedoch nicht deren Manipulation. Das Verfahren ist daher nicht in der Lage, Faktenfehler zu diagnostizieren. Ebensowenig können Korrekturvorschläge aufgrund der in Regelverletzungen beteiligten Kanten generiert werden. Das Parsing von bewerteten CD-Grammatiken leidet zudem an Effizienzproblemen, wenn komplexere linguistische Phänomene modelliert werden sollen. Aus Effizienzgründen dürfen maximal zwei Kanten in einer Regel miteinander in Beziehung gesetzt werden. Gerade aber diese Einschränkung erschwert die Modellierung von Beziehungen, in denen Konstituenten in der Strukturbeschreibung weit voneinander entfernt sind. Andere Grammatikformalismen verfügen meist über einen Mechanismus, Merkmale von einer Konstituente auf eine andere zu übertragen. Dieser Merkmalstransport, häufig durch die Verwendung des Unifikationsoperators realisiert, ist in CD-Grammatiken nicht möglich.

Ebene	Beschreibung	Anzahl der verwendeten Kantenbezeichnungen
SYN	Syntaktische Wohlgeformtheitsbedingungen und sortale Beschränkungen der Valenzfüller.	12
SYN–DET	Substantive benötigen einen Artikel.	1
SYN–PN	Präpositionen benötigen ein Substantiv.	1
SYN–SUBJ	Finite Verben benötigen ein Subjekt.	1
SYN–IOBJ	Bestimmte Verben benötigen ein indirektes Objekt.	1
SYN–DOBJ	Bestimmte Verben benötigen ein direktes Objekt.	1
SYN–PMOD	Bestimmte Verben benötigen eine Präpositionalgruppe.	1
SYN–VPAR	Das Hilfsverb „haben" benötigt ein Partizip.	1

Tabelle 5.1: Analyseebenen der Grammatiken G_1 bis G_3.

Für die folgende Diskussion werden drei bewertete CD-Grammatiken, G_1, G_2 und G_3, mit insgesamt acht Analyseebenen verwendet.[1] Sieben Ebenen sind dabei für die Modellierung von Obligatheitsforderungen zuständig (vgl. Abschnitt 4.3.2). Um die Grammatiken möglichst einfach zu gestalten, wurde auf eine Realisierung einer eigenständigen Semantikebene verzichtet. Es können damit hauptsächlich nur syntaktische Fehler in einer Äußerung diagnostiziert werden.[2] Die Tabelle 5.1 enthält eine Kurzbeschreibung der Ebenen sowie die Anzahl der verwendeten Kantenbezeichnungen. Alle drei Grammatiken verwenden die gleiche Teilmenge von Regeln. Nur die Grammatik G_3 besitzt acht zusätzliche Präferenzregeln, welche einige Lesarten der beteiligten Wortformen bevorzugen. Diese Regeln genügen folgendem Schema:[3]

$$\mathsf{X.level} = \mathsf{SYN} \wedge \mathsf{exists}(\mathsf{X} \downarrow < \textit{Merkmal} >) \rightarrow \mathsf{X} \downarrow < \textit{Merkmal} > \neq < \text{Wert} >$$

Durch die Verwendung verschiedener Regelstrafen werden z.B. Lesarten, die im Genus Femininum sind, bevorzugt. Insgesamt werden 92 (100) Regeln verwendet, davon sind 72 (80) unär und 20 binär. Sei C die Regelmenge einer

[1] Die verwendeten bewerteten CD-Grammatiken sind Erweiterungen der Grammatik aus dem Anhang A.3. Zusätzlich werden Regeln verwendet, die, sofern möglich, den Wohlgeformtheitsbedingungen der Fehlersimulation im Anhang A.4 entsprechen.

[2] Die sortalen Beschränkungen der Valenzfüller von Verben werden auf der Syntaxebene modelliert. Sie sind notwendig, um semantisch inadäquate Strukturbeschreibungen ausschließen zu können. Zum Beispiel sollte in dem Satz „Die Blume sucht das Kind." nur das Substantiv „Kind" als Subjekt angesehen werden können. Eine Diagnose von verletzten sortalen Beschränkungen, wie z.B. in „Die Blume sucht das Buch." ist auch möglich.

[3] Die Technik, bestimmte Lesarten zu bevorzugen, findet sich in [Schröder 1999].

bewerteten CD-Grammatik. Die für die Diagnose relevanten Regeln sind durch die Menge $C_{\text{diag}} \subset C$ gegeben.[4] Die Grammatiken G_1 bis G_3 unterscheiden sich durch die Verwendung unterschiedlicher Regelbewertungen $\Phi(c)$.

- Grammatik G_1: Es werden keine harten Regeln verwendet, d.h.

$$\forall c \in C : \Phi(c) > 0.$$

- Grammatik G_2: Für einige Regelbewertungen, insbesondere für morpho-syntaktische Wohlgeformtheitsbedingungen, werden *gleiche* Regelbewertungen verwendet. Insbesondere sollen alle Diagnosen, die das Verfahren der Fehlersimulation im Modus der Regelfehlersimulation generiert, verfügbar sein. Die morpho-syntaktischen Wohlgeformtheitsbedingungen sind durch die Menge $C_{\text{morph}} \subseteq C_{\text{diag}}$ gegeben. Für die Regelbewertungen ergibt sich:

$$\forall c, c' \in C_{\text{morph}} \wedge c \neq c' : \Phi(c) = \Phi(c').$$

- Grammatik G_3: Die Regelbewertungen sind so gewählt, daß in den meisten Fällen nur eine Dependenzstruktur für eine Äußerung generiert wird. Alle Regelbewertungen aus der Regelmenge C_{morph} sind verschieden:

$$\forall c, c' \in C_{\text{morph}} \wedge c \neq c' : \Phi(c) \neq \Phi(c')$$

Die benötigte Zeit für die Analyse des Satzes (13)

(13) * „Ich habe mich auf meinem kleinen Stuhl gesetzt.“

wird für alle drei Grammatiken gemessen. Dabei wird die Bestensuche *branch and bound* verwendet (siehe Abschnitt 4.2.3). Die Ergebnisse sind in Tabelle 5.2 zu sehen.[5]

Grammatik	Laufzeit
G_1	> 20 min
G_2	160 s
G_3	100 s

Tabelle 5.2: Laufzeitvergleich der Grammatiken G_1 bis G_3.

Die Analyse der Grammatik G_1 mußte aufgrund des hohen Ressourcenbedarfs abgebrochen werden. Ein Verzicht auf harte Regeln ist in der Praxis nicht möglich. Für die Anwendung in einem Sprachlernsystem ist selbst die Analysezeit der Grammatik G_3 noch zu groß. Bezogen auf die Diagnosequalität, gemessen an der verfügbaren Anzahl von Diagnosen, ist die Grammatik G_2 zu bevorzugen, da sie drei verschiedene Diagnosen liefert, während die Grammatik G_3 nur eine einzige liefert. Ob eine einzelne Diagnose den Fehler in jedem Fall adäquat erklären kann, ist jedoch anzuzweifeln. Möchte man eine ähnliche Diagnosequalität wie im Verfahren der Fehlersimulation für die Regelfehlersimulation erhalten, sollte die Grammatik G_2 verwendet werden.

[4] Im Anhang A.3 werden diese Regeln durch den Präfix CID−ERR des Regelbezeichners markiert.

[5] Die Testsätze wurden auf einer Sun Ultra 2 mit 300 Mhz analysiert.

5.1.1 Modellierungstechniken zur Suchraumreduktion

Betrachten wir eine CD-Grammatik mit m Analyseebenen und m_l Kantenbezeichnungen für jede Ebene l. Die zu analysierende Äußerung sei $\omega = w_1 \dots w_n$. Die Menge aller Lesarten der Wortform w_i wird mit $R(w_i)$ bezeichnet (vgl. Definition 3.2.4). Das CDG-Parsing konstruiert ein CSP mit $m \cdot \sum_{1 \le i \le n} |R(w_i)|$ Variablen. Für jede Wortform muß genau eine Lesart ausgewählt werden, wofür es insgesamt $\prod_{1 \le i \le n} |R(w_i)|$ Möglichkeiten gibt. Nachdem die Lesarten ausgewählt wurden, sind damit die Variablen der Analyseebenen festgelegt. Für jede Variable muß ein Wert aus dem Wertebereich bestimmt werden. Sei $r_{i,l}$ die Größe des Wertebereichs der zur i-ten Wortform und zur l-ten Analyseebene gehörenden Variable. Die Gesamtgröße $s(\omega)$ des Suchraums[6] ist

$$s(w_1 \dots w_n) = \prod_{1 \le i \le n} |R(w_i)| \cdot \prod_{1 \le l \le m} \prod_{1 \le i \le n} r_{i,l}.$$

Werden keine harten unären Regeln verwendet, gibt es $m_l \cdot n$ Werte für jeden Wertebereich der Variablen der l-ten Analyseebene.

Als Beispiel wird die Suchraumgröße für die Analyse des Satzes (13) unter der Verwendung der Grammatiken G_1 bis G_3 des vorherigen Abschnitts angegeben. Die Anzahl der Lesarten pro Wortform für den Satz (13) ist in Tabelle 5.3 dargestellt. Die initiale Größe des Suchraums ist

$$s(\omega) = \prod_{1 \le i \le n} |R(w_i)| \cdot \prod_{1 \le l \le m} (m_l \cdot n)^n = 5.7 \cdot 10^{69}.$$

Wortform	ich	habe	mich	auf	meinem	kleinen	Stuhl	gesetzt
Lesarten	1	2	2	2	2	44	3	1

Tabelle 5.3: Anzahl der Lesarten pro Wortform.

Es gibt verschiedene Techniken, den Suchraum einzuschränken. Die wichtigste ist sicherlich die Verwendung von harten unären Regeln. Dadurch können die Wertebereiche vor der eigentlichen Suche schon erheblich reduziert werden. Die Grammatiken G_2 und G_3 verwenden die gleichen harten Bedingungen. G_1 kann den Suchraum nicht weiter einschränken, da nur weiche Bedingungen verwendet werden. Die Tabelle 5.4 zeigt die Größe der Wertebereiche nach der Auswertung der harten Regeln für G_3. Der Suchraum reduziert sich um zahlreiche Größenordnungen:

$$s(\omega) = 1.5 \cdot 10^{11}.$$

Die Experimente des vorherigen Abschnittes haben allerdings gezeigt, daß die Analysezeit selbst unter ausgiebiger Verwendung von unären harten Regeln noch zu groß ist. Es ist daher meist notwendig, noch weitere Maßnahmen für die Suchraumreduktion in Erwägung zu ziehen. Eine Modellierungstechnik erlaubt die Reduktion der lexikalischen Mehrdeutigkeit. Dazu werden sogenannte *unterspezifizierte* Merkmale in der Wissensrepräsentation des Lexikons verwendet.

[6] Der Suchraum besteht aus der Menge aller möglichen Variablenbelegungen im CSP (Lösungstupel), wobei die verwendeten Variablen von der ausgewählten Lesart abhängig sind.

i	Wortform w_i	SYN $r_{i,1}$	SYN–DET $r_{i,2}$	SYN–PN $r_{i,3}$	SYN–SUBJ $r_{i,4}$	SYN–IOBJ $r_{i,5}$	SYN–DOBJ $r_{i,6}$	SYN–PMOD $r_{i,7}$	SYN–VPAR $r_{i,8}$
1	ich	6							
2	habe				4		4		2
3	mich	6							
4	auf	5		4					
5	meinem	4							
6	kleinen	4							
7	Stuhl	6	2						
8	gesetzt	2					4	2	
$\prod r_{i,l}$		34560	2	4	4	1	16	2	2

Tabelle 5.4: Größe der Wertebereiche nach der Auswertung unärer harter Bedingungen.

Ein unterspezifiziertes Merkmal enthält spezielle Werte, um mögliche Wertkombinationen zu realisieren. Die Merkmalsstruktur der Wortform „Stuhl“ kann z.B. so kodiert werden, daß nur eine Lesart notwendig ist. Dazu wird für das Kasusmerkmal der spezielle Wert nad verwendet, welcher alle drei Kasuslesarten, Nominativ, Dativ und Akkusativ, darstellt:

$$\mathsf{Stuhl} : [\mathsf{cat} :\{\mathsf{n}\}\ \ \mathsf{num} :\{\mathsf{sing}\}\ \ \mathsf{case} :\{\mathsf{nad}\}\ \ \mathsf{gender} :\{\mathsf{masc}\}].$$

Wird diese Technik konsequent verwendet, können erheblich weniger Lesarten berücksichtigt werden. Allerdings müssen die Grammatikregeln für alle unterspezifizierten Merkmale angepaßt werden. Zum Beispiel müssen die Kasuskongruenzregeln für Substantive den unterspezifizierten Wert berücksichtigen:

$$\begin{aligned}&\mathsf{X.level} = \mathsf{SYN} \wedge \mathsf{X.label} = \mathsf{DET} \wedge \neg\mathsf{root}(\mathsf{X}\uparrow) \rightarrow \\ &\quad(\neg\mathsf{X}\downarrow \mathsf{case} = \mathsf{gen} \wedge \mathsf{X}\uparrow \mathsf{case} = \mathsf{nad}) \vee (\mathsf{X}\downarrow \mathsf{case} = \mathsf{X}\uparrow \mathsf{case}).\end{aligned}$$

Da alle Sonderfälle in den Regeln berücksichtigt werden müssen, können sich leicht Modellierungsfehler einschleichen. Ein Problem dieser Technik liegt darin, daß Regeln mit unterspezifizierten Merkmalen nicht mehr transitiv sind.

5.1.2 Transitivität und Merkmalstransport

In Satz (13) liegt eine transitive Bedingungskette z.B. für die Kongruenzforderung im Kasus vor: Die Präposition bestimmt den Kasus des Substantivs und der Artikel und das Adjektiv kongruieren mit dem Substantiv. Werden unterspezifizierte Merkmale verwendet, kann die Transitivität nicht garantiert werden: Der Kasusfehler in (13) wird nicht erkannt. Es muß zusätzlich eine linguistisch wenig sinnvolle Kongruenzforderung zwischen dem Artikel und der Präposition hergestellt werden. Dafür ist schon eine binäre Regel erforderlich. Stehen mehr als zwei Dependenzkanten in einer transitiven Beziehung, kann der transitive Abschluß nicht mehr modelliert werden.

Werden dagegen keine unterspezifizierten Merkmale verwendet, können transitive Regeln formuliert werden, z.B. indem die Gleichheitsrelation verwendet

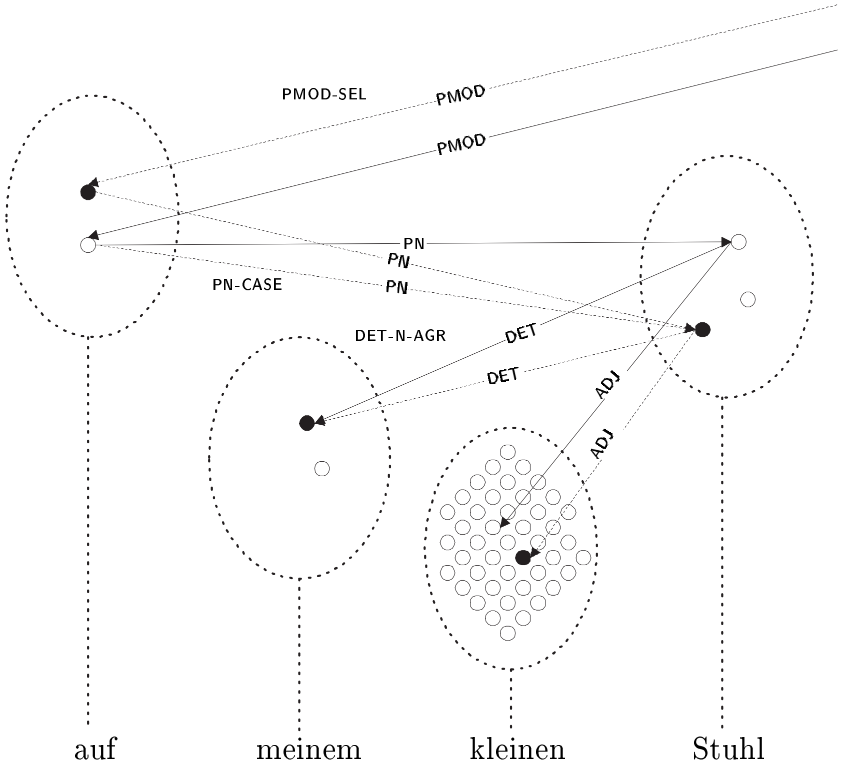

Abbildung 5.1: Herstellung von Transitivität in CD-Grammatiken.

wird. Abbildung 5.1 zeigt eine Teilstruktur der Äußerung unter Verwendung der Grammatik G_2.[7] Die drei verfügbaren Diagnosen (Selektionsforderung des Verbs, Rektionsforderung der Präposition und fehlende Kasuskongruenz zwischen Possessivpronomen und Substantiv) werden durch die Auswahl verschiedener Lesarten ermöglicht.

Dieser Mechanismus kann in der Praxis zum Transport von Merkmalen verwendet werden. Allerdings gelingt dies nur dann, wenn die Wortformen innerhalb der Bedingungskette die zu transportierenden Merkmale realisieren. In Abbildung 5.2 ist die Dependenzstruktur des Satzes (13) dargestellt. Das Reflexivpronomen muß mit dem Subjekt in Numerus und Person übereinstimmen. Die Kongruenz kann jetzt nicht mehr durch die Auswahl von kongruierenden Lesarten realisiert werden, da einerseits das Reflexivpronomen in keiner direkten Beziehung zum Subjekt steht und da andererseits das direkt verbundene Partizip die Merkmale Person und Numerus nicht realisiert. Eine binäre Regel könnte zwar die Kongruenz mit dem finiten Verb fordern, jedoch reicht ein Kongruenzfehler zwischen finitem Verb und Subjekt, um ein fehlerhaftes Reflexivpronomen zu verdecken, wie z.B. in

(14) * „Ich hat sich auf meinen kleinen Stuhl gesetzt."

In Modalverbkonstruktionen sind selbst binäre Regeln nicht mehr ausreichend:

[7] Schwarze Kreise symbolisieren Lesarten im Dativ, während weiße Kreise Lesarten im Akkusativ darstellen. Die drei Diagnosen des CDG-Parsings stehen an den beteiligten Kanten.

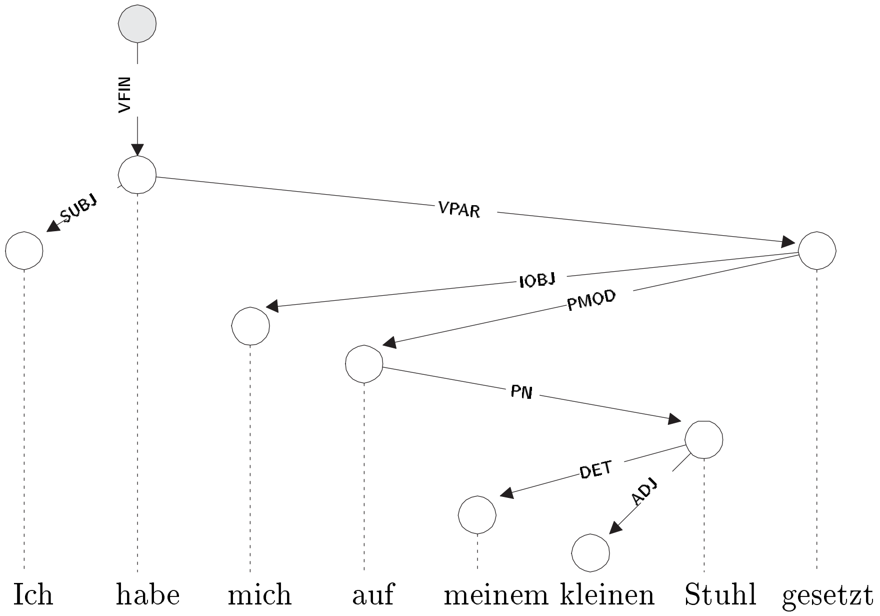

Abbildung 5.2: Merkmalstransport für die Modellierung von Reflexivpronomen.

(15) * „Ich soll sich auf meinen kleinen Stuhl gesetzt haben.“

Das Reflexivpronomen ist über vier Kanten mit dem Subjekt verbunden. Eine binäre Regel kann maximal das Hilfsverb „haben“ im Infinitiv referenzieren.

5.1.3 Beschränkung auf binäre Regeln

Die Regeln im CD-Grammatik-Formalismus sind aus Effizienzgründen auf maximal binäre Beziehungen zwischen Dependenzkanten beschränkt. Jeder Suchzustand des Suchraums wird bewertet, indem die Kostenfunktion für die aktuelle Wertebelegung der Variablen berechnet wird. Der Formalismus stellt sicher, daß jede Regel in konstanter Zeit c ausgewertet werden kann. Die Berechnung der Kostenfunktion für n Variablen und für n_q Regeln der Stelligkeit q benötigt $c \cdot n_q\binom{n}{q}$ Schritte. Die Auswertung einer ternären Regel wächst damit in der Größenordnung von $O(n^3)$ in der Anzahl der Variablen. Dieser erhöhte Rechenaufwand würde die Performanz des Verfahrens deutlich verschlechtern.[8]

Die Verwendung von maximal binären Regeln stellt eine Modellierungsgrenze da: Insbesondere logisch zusammengehörige Konstituenten, die in einer Strukturbeschreibung weit voneinander entfernt sind, können nicht direkt in Beziehung gesetzt werden. Wohlgeformtheitsbedingungen z.B für Reflexivpronomen, Possessivpronomen oder Relativpronomen können durch binäre Regeln allenfalls approximiert werden.

Ähnliche Einschränkungen der Modellierung entstehen, wenn mehrere Analyseebenen verwendet werden: Um die Ergebnisse einer Ebene in einer ande-

[8]Der Satz (13) besitzt allein auf der Syntaxebene 57 Variablen. Eine binäre Regel benötigt $c \cdot 1596$ Schritte, während eine ternäre schon $c \cdot 29260$ Schritte benötigen würde.

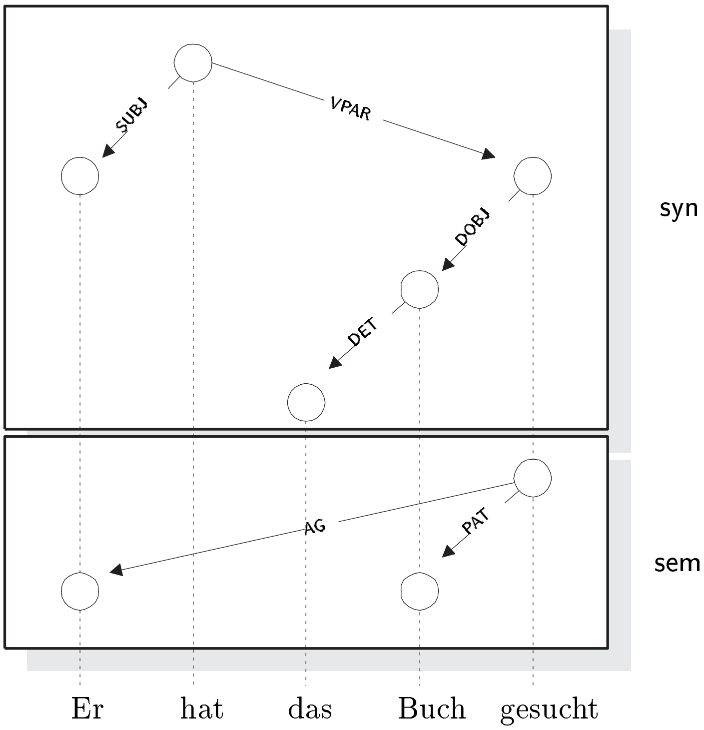

Abbildung 5.3: Beziehungen zwischen syntaktischen und semantischen Strukturen.

ren Ebene benutzen zu können, sind Regeln erforderlich, die die Dependenzkanten der jeweiligen Ebenen miteinander in Beziehung setzen. Allerdings ist der Anwendungsbereich dieser sogenannten *Mapping*-Regeln noch stärker eingeschränkt, da jeweils nur eine Kante pro Ebene referenziert werden kann. In [Menzel 2000] werden Beispiele für auf Dependenzrelationen basierenden semantischen Strukturen aufgeführt, in denen die zugehörige syntaktische Struktur einfache Formen einer Strukturtransformation benötigt. Die semantische Modellierung basiert auf semantischen Rollen (Agens, Patiens, Instrument, etc.), wobei der Valenzrahmen eines Verbs für jede Rolle eine selektionale Beschränkung für den Valenzfüller definiert. Für analytische Verbformen, bei denen das finite Verb nicht mehr Träger der Verbvalenzen ist, wird schon eine ternäre Abbildungsregel benötigt, d.h. es müssen drei Kanten, eine auf der Semantikebene und zwei auf der Syntaxebene, verglichen werden. Um z.B. in Abbildung 5.3 eine Identität zwischen dem Agens und dem Subjekt zu fordern, müssen drei zusammenhängende Kanten in Beziehung gesetzt werden (AG, VPAR, SUBJ).

5.1.4 Potential des integrativen Ansatzes

Das Verfahren der Fehlersimulation aus Kapitel 3 weist im Vergleich zum CDG-Parsing einige komplementäre Eigenschaften auf: Es gestattet die effiziente Realisierung von transitiven Bedingungsketten durch den Wertaktualisierungsoperator. Es können Faktenfehler diagnostiziert und Korrekturvorschläge generiert werden. Außerdem ist es nicht nötig, unterspezifizierte Merkmale zu verwenden, da disjunktive Lesarten durch die Mengenrepräsentation dargestellt werden können. Allerdings muß für das Verfahren die Struktur einer Äußerung als bekannt vorausgesetzt werden. Weiterhin können nur wenige linguistische Phänomene, im wesentlichen nur Kongruenzforderungen, modelliert werden.

Eine einfache Kopplung beider Verfahren kann einige Defizite kompensieren: Das CDG-Parsing bietet die Möglichkeit, auch für fehlerhafte Äußerungen eine Strukturbeschreibung generieren zu können. Diese Beschreibung dient als Ausgangslage für die Formulierung von merkmalsbasierten Wohlgeformtheitsbedingungen, die anschließend mit dem Verfahren der Fehlersimulation analysiert werden. Die Grammatik kann in zwei relevante Teilbereiche zerlegt werden. Für die Analyse dieser Teilbereiche trägt jeweils das entsprechende Verfahren die Verantwortung. Aufgrund der stark eingeschränkten Modellierungsmöglichkeiten der Fehlersimulation sind dies inbesondere Beziehungen, die auf morphologischen Merkmalen basieren, wie z.B. Kasus, Genus und Numerus. Gerade diese Merkmale sind wesentlich verantwortlich für die hohe lexikalische Mehrdeutigkeit, wodurch eine starke Entlastung für die CDG-Analyse zu erwarten ist. Weiterhin kann die Diagnosequalität der Fehlersimulation ausgenutzt werden. Auf der anderen Seite können für strukturelle Fehler, z.B. fehlende Komplemente oder überflüssige Wortformen, die verletzten Bedingungen der CD-Grammatik als Fehlererklärung herangezogen werden.

5.2 Verarbeitungsmodell

Der Vorschlag einer sequentiellen Kopplung beider Verfahren findet sich in [Menzel 1998c]. Für die Realisierung wird eine fortlaufend weiterentwickelte Implementation des CDG-Parsingverfahrens verwendet. Dieses System ist ausführlich in [Schröder 1998] beschrieben. Für die Fehlersimulation wird auf die Implementation aus [Stockfleth 1999] zurückgegriffen.

Sowohl das CDG-Parsing als auch die Fehlersimulation benötigen ein Lexikon. Die Repräsentation lexikalischen Wissens unterscheidet sich jedoch für beide Systeme. Im integrativen Ansatz wird die Lexikonrepräsentation der Fehlersimulation, wie sie im Abschnitt 3.2.1 vorgestellt wurde, verwendet. Das Lexikon des CDG-Systems wird dynamisch aus dem Lexikon der Fehlersimulation, der bewerteten CD-Grammatik und der vorliegenden Äußerung konstruiert. Dieser Konvertierungsprozeß muß nicht alle Merkmale des Lexikons berücksichtigen, sondern nur diejenigen, welche in den Grammatikregeln der CD-Grammatik verwendet werden. Dies kann festgestellt werden, indem die Zugriffsoperationen $\uparrow$ und $\downarrow$ in den Regeln analysiert werden. Dadurch wird sichergestellt, daß nur die für die Grammatik relevante Information während des Parsings berücksichtigt werden muß.

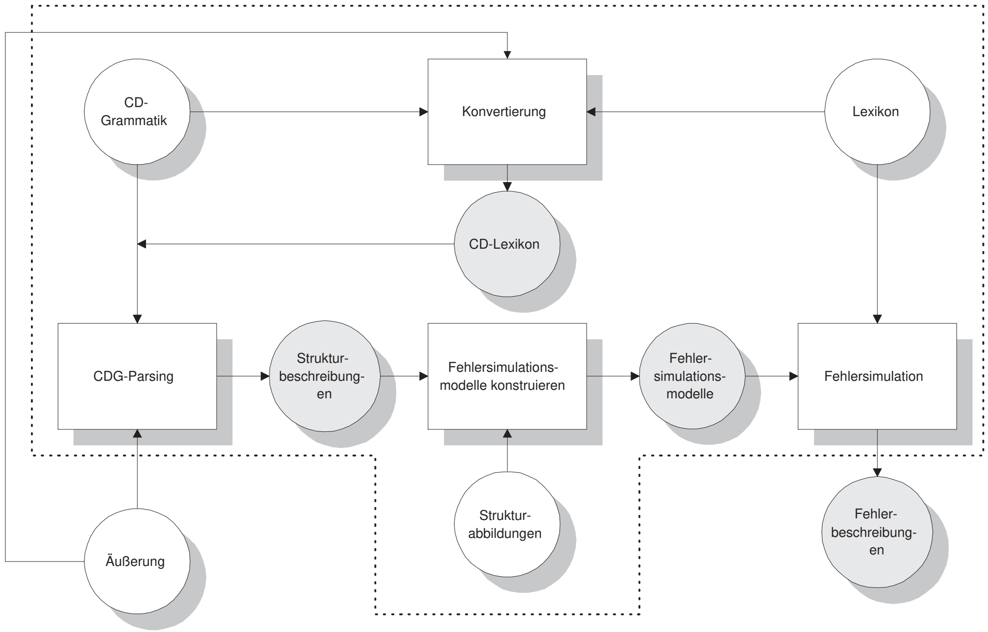

Abbildung 5.4: Systemarchitektur des integrativen Ansatzes

Die Systemarchitektur des integrativen Verfahrens ist in Abbildung 5.4 dargestellt.[9] Realisiert wurde eine Diagnosekomponente, deren Aufgabe es ist, aus einer Äußerung eine Menge von *Fehlerbeschreibungen* zu generieren. Der Begriff der Fehlerbeschreibung bezeichnet dabei die gesamte Information, die nach der Analyse der Äußerung vorhanden ist. Dazu gehört eine optimale Strukturbeschreibung und die Menge aller Diagnosen der Fehlersimulation für diese Strukturbeschreibung. Die Menge der verletzten Regeln der CD-Grammatik und die beteiligten Kanten ergeben sich nach Definition 4.3.1 direkt aus der optimalen Strukturbeschreibung. Folgende Schritte bestimmen den Ablauf der Analyse im integrativen Verfahren:

1. Aus dem Lexikon, der CD-Grammatik und der vorliegenden Äußerung wird ein CD-Lexikon nach Definition 4.1.1 erzeugt.

2. Das CDG-Parsingverfahren ermittelt die Dependenzanalyse der Äußerung. Das Ergebnis ist eine Menge von optimalen Strukturbeschreibungen.

3. Für jede Strukturbeschreibung wird ein Fehlersimulationsmodell konstruiert. Die Konstruktionsvorschriften werden dabei durch Strukturabbildungen angegeben, welche im Abschnitt 5.3 behandelt werden.

4. Jedes Fehlersimulationsmodell wird simuliert. Das Ergebnis ist eine Menge von Fehlerbeschreibungen, d.h. eine Menge von Diagnosen für jede Strukturbeschreibung. Dabei werden allerdings nur diejenigen Fehlerbeschreibungen verwendet, deren Diagnosen eine minimale Anzahl von verletzten Komponenten aufweisen.

[9]Die weißen Kreise in der Abbildung stellen die für die Analyse benötigten Ressourcen dar, während die grauen Kreise dynamisch erzeugte Ressourcen repräsentieren.

Der vierte Verarbeitungsschritt realisiert eine Disambiguierungsstrategie für alternative Strukturbeschreibungen. Er ist vor allem notwendig, wenn die CD-Grammatik keine Kongruenzregeln enthält, z.B. wenn nur eine Kasusmarkierung die syntaktische Funktion einer Konstituente festlegt. In dem Satz

(16) „Ihren Mann sucht meine Frau."

wird entweder das Substantiv „Mann" oder das Substantiv „Frau" als Subjekt des Satzes angesehen. Das CDG-Parsing liefert zwei Strukturbeschreibungen, für die jeweils ein Fehlersimulationsmodell konstruiert und anschließend simuliert. Da die Subjektlesart von „ihren Mann" die Nominativforderung für das Subjekt verletzt, während die alternative Lesart fehlerfrei ist, wird erstere nicht weiter berücksichtigt.

Für die Modellierung grammatikalischen Wissens kommen im wesentlichen zwei Möglichkeiten in Betracht:

- Strikte Trennung der Aufgabenbereiche: Die CD-Grammatik und die Bedingungen der Fehlersimulation modellieren unterschiedliches sprachliches Wissen.
- Überlappende Modellierung: Die CD-Grammatik modelliert mehr oder weniger vollständig die Gesamtgrammatik. Die Regeln der Fehlersimulation sind zum Teil schon Bestandteil der CD-Grammatik, ermöglichen jedoch einige Defizite des CDG-Parsings zu kompensieren.

Für die erste Modellierungstechnik ist eine Effizienzverbesserung zu erwarten, da nur wenige Merkmale für die Grammatik relevant sind. Die Grammatik verzichtet z.B. auf morphologische Merkmale wie Kasus, Genus, Numerus und Person. Die fehlenden Wohlgeformtheitsbedingungen werden im Fehlersimulationsmodell repräsentiert. Dies hat zur Folge, daß einige Präferenzregeln nicht mehr formuliert werden dürfen: Eine Unterordnung könnte fälschlicherweise nur präferiert werden, weil eine konfliktverursachende Regel im Fehlersimulationsmodell enthalten und nicht mehr Bestandteil der CD-Grammatik ist. Insbesondere die Präferenz, daß das Subjekt im Satz meistens vor dem finiten Verb steht, kann nicht verwendet werden, da z.B. die CDG-Analyse in dem Satz

(17) „Dem Kind gibt der Mann den Ball."

immer die Konstituente „dem Kind" als Subjekt ansehen würde.

Die zweite Modellierungstechnik hingegen unterliegt nicht dieser Einschränkung. Allerdings sollte die CD-Grammatik so formuliert werden, daß möglichst nur eine Strukturbeschreibung ermittelt wird, z.B. indem immer verschiedene Regelstrafen verwendet werden. Ist dies nicht der Fall, würde die Fehlersimulation für alternative Strukturen des CDG-Parsings nahezu identische Diagnosen liefern.[10] Der Nachteil dieser Technik ist der hohe Zeitbedarf des CDG-Parsingverfahrens.

[10]Dies liegt daran, daß die Fehlersimulation nicht die Merkmalsstrukturen der verwendeten L-Knoten verwendet, um die Merkmalsstrukturen der Modellvariablen festzulegen. Diese ergeben sich einzig aus den möglichen Lesarten der Äußerung.

5.3 Strukturabbildungen

Die Aufgabe der Strukturabbildung ist es, aus einer optimalen Strukturbeschreibung einer bewerteten CD-Grammatik ein Fehlersimulationsmodell zu konstruieren. Eine Strukturbeschreibung nach Definition 4.1.4 ist eine Menge von Dependenzbäumen, bestehend aus einem Wurzelknoten und den L-Knoten der zu analysierenden Äußerung sowie den mit einer Bezeichnung versehenen Dependenzkanten. Ein Fehlersimulationsmodell nach Definition 3.2.7 enthält eine Menge von einfachen Wohlgeformtheitsbedingungen, wobei nur restrict- und agree-Prädikate erlaubt sind. Die Bedingungen der Fehlersimulation werden auf Modellvariablen formuliert, die als Platzhalter für die Wortformen der Äußerung dienen.

Um aus einer optimalen Strukturbeschreibung ein Fehlersimulationsmodell konstruieren zu können, müssen sowohl die Modellvariablen als auch die Prädikate des Modells bestimmt werden. Die Modellvariablen können einfach aus der Äußerung konstruiert werden, indem für jede Wortform eine eindeutige Modellvariable erzeugt wird.

Die zu verwendenden Wohlgeformtheitsbedingungen der Fehlersimulation sind abhängig von der Strukturbeschreibung. Deren Eigenschaften definieren die Prädikate des Fehlersimulationsmodells. Unter einer Strukturabbildung wird im folgenden eine Vorschrift verstanden, die festlegt, welche Prädikate instanziiert werden sollen, wenn die Struktur bestimmten Eigenschaften genügt. Diese Vorschrift hat die Form

$$< \mathit{Struktureigenschaft} > \quad \rightarrow \quad < \mathit{Prädikate\ der\ Fehlersimulation} > .$$

Die Struktureigenschaft wird durch einen logischen Ausdruck auf den Dependenzbäumen definiert. Um die Implementation möglichst einfach und effizient zu gestalten, werden nur konjunktiv verknüpfte Formeln verwendet. Es werden folgende Formeln unterstützt:

- Ausdrücke, um einen Merkmalswert eines L-Knotens zu überprüfen:

  $$\mathsf{value}(\mathit{L\text{-}Knoten}, \mathit{Merkmal}, \mathit{Vergleichswert}).$$

  Die Formel ist genau dann wahr, wenn der Wert des Merkmals des L-Knotens eine Teilmenge des Vergleichswert ist.

- Ausdrücke, um eine Kante einer Analyseebene mit einer bestimmten Bezeichnung zu referenzieren:

  $$\mathsf{modifies}(\mathit{Knoten}, \mathit{Kantenbezeichnung}, \mathit{Knoten}, \mathit{Analyseebene}).$$

  Die Formel $\mathsf{modifies}(x, l, y, i)$ ist genau dann wahr, wenn es eine Kante im Dependenzbaum der Analyseebene i gibt, welche den Knoten y mit dem Knoten x verbindet und mit l bezeichnet ist.

- Ausdrücke, um festzustellen, ob es *keine* Kante gibt, die einen bestimmten Knoten modifiziert:

  $$\mathsf{no\text{-}modifier}(Kantenbezeichnung, Knoten, Analyseebene).$$

  Die Formel $\mathsf{no\text{-}modifier}(l, y, i)$ ist genau dann wahr, wenn der Dependenzbaum der Ebene i keine Kante mit der Bezeichnung l enthält, welche den Knoten y mit einem anderen Knoten verbindet.

Die rechte Seite der Abbildungsvorschrift enthält eine Konjunktion von restrict- und agree-Prädikaten. Als Argumente dürfen allerdings keine Modellvariablen, sondern nur L-Knoten verwendet werden. Weiterhin können nur Variablen für die Knoten eines Dependenzbaumes verwendet werden. Die folgende Abbildungsvorschrift modelliert z.B. die Kasusforderung des Verbs für ein direktes Objekt:

$$\mathsf{modifies}(x, \mathsf{DOBJ}, y, \mathsf{syn}) \rightarrow \mathsf{agree}(y, x, \mathsf{dobj\text{-}case}, \mathsf{case}) \qquad (a_1)$$

Dabei sind x und y Variablen, die für Knoten im Dependenzbaum stehen. Gibt es Knoten, die die linke Seite der Abbildungsvorschrift erfüllen, werden die Prädikate auf der rechten Seite instanziiert und in das Fehlersimulationsmodell eingefügt. Die Instanziierung erfordert eine Ermittlung der Modellvariablen aus den Knoten. Dazu wird einfach der Positionsindex des L-Knotens verwendet.

5.3.1 Konstruktion eines Fehlersimulationsmodells

Sei $G = (L', \perp, \mathcal{L}, R, C)$ eine bewertete CD-Grammatik mit $m = |R|$ Analyseebenen nach Definition 4.2.1. Sei $S = \{(V_\perp, E_i) | 1 \leq i \leq m\}$ eine optimale Strukturbeschreibung einer Äußerung $\omega = w_1 \dots w_n$ bezüglich G. Eine Strukturbeschreibung nach Definition 4.1.4 enthält für jede Analyseebene einen Dependenzbaum mit den Knoten $V_\perp = V \cup \{\perp\}$ und den Kanten $E_i \subseteq V_\perp \times \mathcal{L} \times V$, wobei $\perp$ den Wurzelknoten und V die L-Knoten von ω repräsentieren. Ein L-Knoten $((w_j, f), j)$ enthält die Wortform w_j, eine Merkmalsstruktur f und die Position j in ω. Die Formeln auf einer Strukturbeschreibung S haben folgende Semantik:

$$\mathsf{value}(x, m, s) \Leftrightarrow x \in V \wedge x = ((w, f), j) \wedge m \in \mathrm{dom}(f) \wedge f(m) \subseteq s. \quad (5.1)$$

$$\mathsf{modifies}(x, l, y, r_i) \Leftrightarrow (y, (r_i, l), x) \in E_i. \quad (5.2)$$

$$\mathsf{no-modifier}(l, x, r_i) \Leftrightarrow \neg \exists y \in V : (x, (r_i, l), y) \in E_i. \quad (5.3)$$

Eine Strukturabbildung kann als Paar (Q, T) mit einer Menge von Formeln Q auf der Strukturbeschreibung und einer Menge von Prädikaten T der Fehlersimulation dargestellt werden. Die Menge aller verwendeten Strukturabbildungen sei U. Aus einem Lexikon L nach Definition 3.2.2, der vorliegenden Äußerung ω und den Regeln der CD-Grammatik wird das CD-Lexikon L' dynamisch konstruiert. Das implementierte Konstruktionsverfahren eines Fehlersimulationsmodell $\mathcal{M} = (L, K, P)$ mit den zu bestimmenden Modellvariablen K und den

Prädikaten P nach Definition 3.2.7 ist in Algorithmus 1 angegeben. Die Auswertung der Formelmenge Q verwendet den Algorithmus 2, welcher auf einer Vereinfachung der Auswertungsstrategie von Prolog ([Clocksin 1994]) basiert: Für jede Formel in dem konjunktiven Ausdruck werden nacheinander aller Variablenbindungen bestimmt, die den Ausdruck erfüllen. Die Auswertung jeder einzelnen Formel, im Algorithmus 2 durch den Aufruf der Funktion eval repräsentiert, verwendet die Semantik der Abbildungsprädikate (5.1) - (5.3), um die erfüllenden Variablenbindungen zu bestimmen.

Algorithmus 1 Konstruktion eines Fehlersimulationsmodells $\mathcal{M} = (L, K, P)$ für eine Strukturbeschreibung S aus einer Menge von Strukturabbildungen U.

```
// Für jede Wortform der Äußerung wird ein Symbol
// für eine Modellvariable eingeführt.
K = {V_i | 1 ≤ i ≤ n}
while U ist nicht leer do
  Entnehme eine Strukturabbildung (Q, T) aus U.
  // Bestimme alle Variablenbindungen,
  // die alle Formeln in Q für S erfüllen (Algorithmus 2).
  B = evalFormulaSet(Q, S)
  // Bestimme für jede Variablenbindung b ∈ B die Modellvariablen in T,
  // instanziiere alle Prädikate aus T und
  // füge sie in das Fehlersimulationsmodell ein.
  for all b ∈ B do
    P = P ∪ {T_[b]}
    // Der Ausdruck T_[b] steht für eine Menge von Formeln,
    // in denen jedes freie Vorkommen der Variablen aus b
    // durch den Wert der Variablen ersetzt ist.
    // Ist der Wert einer Variable ein L-Knoten ((w, f), i),
    // wird er ersetzt durch die Modellvariable V_i.
  end for
end while
```

Für die Realisierung des Konstruktionsverfahrens ist die Bestimmung aller Variablenbindungen, die eine Formelmenge erfüllen, die aufwendigste Aufgabe. Am einfachsten wäre es sicherlich eine Logikprogrammiersprache, wie z.B. Prolog, zu verwenden. Da jede Prolog-Regel immer als eine Implikation $q_1 \wedge \ldots \wedge q_n \rightarrow p$ dargestellt werden kann, können die Formeln auf einer Strukturbeschreibung direkt als Regel formuliert werden, ohne eine Implementation für die Auswertung realisieren zu müssen. Weiterhin können mit nur geringem Aufwand mächtigere Formeln auf einer Strukturbeschreibung zur Verfügung gestellt werden.

Gegen die Verwendung von Prolog spricht allerdings, daß das System für die Analyse von bewerteten CD-Grammatiken in der Programmiersprache C ([Kernighan 1990]) und das System für die Fehlersimulation in C++ ([Stroustrup 1992]) implementiert sind. Daher wäre es erforderlich, die Repräsentation einer Strukturbeschreibung in Prolog-Klauseln zu transformieren, diese anschließend von einem Prolog-Programm einzulesen, die Strukturab-

Algorithmus 2 Berechnet alle Variablenbindungen B, die eine Formelmenge Q für eine Strukturbeschreibung S erfüllen.

```
B = ∅
Z = {(Q, ∅)}
while Z ist nicht leer do
  Entnehme einen Zustand (Q', b) aus Z.
  if Q' ist leer then
    B = B ∪ {b} // Die Variablenbindung b erfüllt Q.
  else
    Entnehme eine Formel q aus Q'.
    // Die Funktion eval berechnet alle mit b verträglichen
    // Variablenbindungen, die q erfüllen.
    // Sie verwendet die Semantik der Abbildungsprädikate (5.1)-(5.3).
    B' = eval(q, b, S)
    for all b' ∈ B' do
      Füge neuen Zustand (Q', b') in Z ein.
    end for
  end if
end while
```

bildungen auszuwerten und das konstruierte Fehlersimulationsmodell wieder zurück nach C++ zu konvertieren.

Die funktionale Komponente des Systems wurde in C++ realisiert, wobei eine Einbindung des CDG-Parsings als C-Bibliothek ohne Schwierigkeiten möglich war. Da nur funktionale Aspekte berücksichtigt sind, sollte eine Portierung auf verschiedene Plattformen möglich sein.[11]

5.3.2 Modellierung linguistischer Phänomene

Die Kongruenz zwischen einem Artikel und einem Substantiv in Kasus, Numerus und Genus kann z.B. durch folgende Strukturabbildung realisiert werden:

$$\begin{aligned}&\mathsf{modifies}(x, \mathsf{DET}, y, \mathsf{syn}) \rightarrow\\ &\quad \mathsf{agree}(x, y, \mathsf{case}) \wedge\\ &\quad \mathsf{agree}(x, y, \mathsf{gender}) \wedge\\ &\quad \mathsf{agree}(x, y, \mathsf{num})\end{aligned} \qquad (a_2)$$

Für alle Kanten im Dependenzbaum der Syntaxebene, die mit DET bezeichnet sind, werden drei agree-Prädikate in das Fehlersimulationsmodell aufgenommen. Die Kongruenzforderung zwischen Adjektiv und Substantiv kann analog formuliert werden. Die Kasusforderung einer Präposition ist leicht zu realisieren:

$$\mathsf{modifies}(x, \mathsf{PN}, y, \mathsf{syn}) \rightarrow \mathsf{agree}(y, x, \mathsf{pcase}, \mathsf{case}) \qquad (a_3)$$

Wird eine Präpositionalgruppe als obligatorisches Komplement eines Verbs benötigt, bestimmt das Verb die selektionalen Eigenschaften der Ergänzung

[11] In der derzeitigen Version des Systems werden die Betriebssysteme Linux und Sun Solaris unterstützt.

(lokal, direktional):

$$\begin{aligned}&\mathsf{modifies}(x, \mathsf{PMOD}, y) \wedge \mathsf{value}(y, \mathsf{cat}, \{\mathsf{v\text{-}fin}, \mathsf{v\text{-}par}, \mathsf{v\text{-}inf}\}) \rightarrow \\ &\qquad \mathsf{agree}(!x, !y, \mathsf{sel})\end{aligned} \qquad (a_4)$$

Der Satz

(18) * „Das Buch liegt auf den kleinen Tisch.“

liefert vier Regelfehlerdiagnosen (Selektionsforderung des Verbs, Kasusforderung der Präposition, Kasuskongruenz bzw. Numeruskongruenz zwischen Artikel und Substantiv) und drei Faktenfehlerdiagnosen mit zwei unterscheidbaren Korrekturen („auf dem kleinen Tisch“ und „auf den kleinen Tischen“).

Das Zusammenspiel von Artikel und Adjektiv in einer Substantivgruppe erfordert in einer CD-Grammatik eine binäre Regel. Der Artikel bestimmt, welcher Flexionsklasse das Adjektiv angehört. Im integrativen Ansatz genügt folgende Abbildung:

$$\begin{aligned}&\mathsf{modifies}(x, \mathsf{DET}, z, \mathsf{syn}) \wedge \mathsf{modifies}(y, \mathsf{ADJ}, z, \mathsf{syn}) \rightarrow \\ &\qquad \mathsf{agree}(!x, y, \mathsf{atype\text{-}sel}, \mathsf{atype})\end{aligned} \qquad (a_5)$$

Dabei kann die Selektionsforderung des Artikels im Modus der Faktenfehlersimulation nicht angezweifelt werden. Dadurch wird z.B. in

(19) * „Ein kleine Mann lacht.“

ein Korrekturvorschlag nur für das Adjektiv („kleiner“) generiert.

Die Verwendung der no-modifier-Formel ist z.B. dann nötig, wenn ein Adjektiv ohne Artikel in einer Substantivgruppe verwendet wird. Die CD-Grammatik liefert nur dann keinen Fehler, wenn das Substantiv eine alleinstehende Verwendung erlaubt. Ist der Wert des Merkmals nodet des Substantivs gleich t, muß das Adjektiv der starken Flexionsklasse angehören:

$$\begin{aligned}&\mathsf{modifies}(x, \mathsf{ADJ}, y, \mathsf{syn}) \wedge \mathsf{no\text{-}modifier}(\mathsf{DET}, y, \mathsf{syn}) \wedge \\ &\mathsf{value}(y, \mathsf{nodet}, \{\mathsf{t}\}) \rightarrow \\ &\qquad \mathsf{restrict}(x, \mathsf{atype}, \{\mathsf{strong}\})\end{aligned} \qquad (a_6)$$

Die Diagnose des Satzes

(20) * „Kleinen Männer lachen.“

enthält u.a. das verletzte restrict-Prädikat. Die Faktenfehlersimulation liefert den Korrekturvorschlag „Kleine Männer lachen.“[12]

Die Forderung, daß das Reflexivpronomen eines reflexiven Verbs mit dem Subjekt in Numerus und Person kongruiert, kann wie folgt modelliert werden:

$$\begin{aligned}&\mathsf{value}(x, \mathsf{cat}, \{\mathsf{refl\text{-}pron}\}) \wedge \mathsf{modifies}(x, l, y, \mathsf{syn}) \wedge \\ &\mathsf{modifies}(z, \mathsf{SUBJ}, y, \mathsf{syn}) \rightarrow \\ &\qquad \mathsf{agree}(x, !z, \mathsf{pers}) \wedge \mathsf{agree}(x, !z, \mathsf{num})\end{aligned} \qquad (a_7)$$

Diese Vorschrift findet einen Kongruenzfehler in der Person in

[12] Es wird alternativ eine Kasusverletzung des Adjektivs mit dem Substantiv diagnostiziert. Dem liegt die Annahme zugrunde, daß das Adjektiv im Dativ steht, z.B. in „Wir geben den kleinen Männer das Buch.“ Diese Diagnose sollte jedoch dispräferiert werden, da dem Adjektiv auf der Oberfläche die Kasusunterscheidung nicht anzusehen ist.

(21) * „Ich freue dich über das Bild.“

sowie einen Korrekturvorschlag der Faktenfehlerdiagnose („mich“). Die Modellvariable für das Subjekt wird in beiden Prädikaten geschützt, um eine Korrektur des Subjekts zu verhindern („Du freust dich über das Bild.“). Um einen Fehler wie z.B. in

(22) * „Ich habe dich über das Bild gefreut.“

entdecken zu können, reicht die Abbildung (a_7) nicht aus. Folgende Abbildungsvorschrift, deren Formulierung als Regel einer CD-Grammatik drei Kanten in Beziehung setzen müßte, löst das Problem:

$$\begin{array}{l} \mathsf{value}(x, \mathsf{cat}, \{\mathsf{refl\text{-}pron}\}) \wedge \mathsf{modifies}(x, l_1, y_1, \mathsf{syn}) \wedge \\ \mathsf{no\text{-}modifier}(\mathsf{SUBJ}, y_1, \mathsf{syn}) \wedge \\ \mathsf{modifies}(y_1, l_2, y, \mathsf{syn}) \wedge \mathsf{modifies}(z, \mathsf{SUBJ}, y, \mathsf{syn}) \rightarrow \\ \qquad \mathsf{agree}(x, !z, \mathsf{pers}) \wedge \mathsf{agree}(x, !z, \mathsf{num}) \end{array} \qquad (a_8)$$

Die Formel no-modifier garantiert, daß immer das am tiefsten eingebettete Subjekt verwendet wird, so daß kein Fehler in dem Satz

(23) „Du glaubst, ich freue mich über das Bild.“

diagnostiziert wird. Diese Forderung kann in CD-Grammatiken nicht korrekt modelliert werden.

Die Verwendung verschiedener Analyseebenen kann für die Modellierung von Possessivpronomen verwendet werden. Wenn in einer Referenzebene ein Possessivpronomen auf seinen Antezedenten zeigt, fordert das Pronomen eine Übereinstimmung mit dem Antezedenten in Genus, Numerus und Person. Da aber ein Possessivpronomen die Funktion eines Artikels hat und damit die Abbildungsvorschrift (a_2) angewendet wird, werden spezielle Merkmale für die Kongruenz mit dem Antezedenten verwendet:

$$\begin{array}{l} \mathsf{modifies}(x, \mathsf{ANT}, y, \mathsf{ref}) \wedge \mathsf{value}(x, \mathsf{cat}, \{\mathsf{poss\text{-}pron}\}) \rightarrow \\ \qquad \mathsf{agree}(x, y, \mathsf{sgender}, \mathsf{gender}) \wedge \\ \qquad \mathsf{agree}(x, y, \mathsf{spers}, \mathsf{pers}) \wedge \\ \qquad \mathsf{agree}(x, y, \mathsf{snum}, \mathsf{num}) \end{array} \qquad (a_9)$$

Die Information, welcher Antezedent zu einem Pronomen gehört, ist natürlich abhängig vom Kontext einer Übung. In den für die Evaluation des Verfahrens verwendeten Grammatiken G_4 und G_5 wird vereinfacht immer eine Kongruenz mit dem Subjekt gefordert, falls das Possessivpronomen nicht das Subjekt modifiziert. Es wird keine Referenzebene verwendet. Der Satz

(24) * „Meine Frau gibt ihm seine Bild.“

liefert zwei Regelfehler- und drei Faktenfehlerdiagnosen. Einerseits kongruiert das Possessivpronomen „seine“ nicht im Genus oder im Numerus mit dem Substantiv „Bild“, andererseits muß der referenzierte Antezedent im Genus maskulin oder neutrum sein. Es gibt zwei unterscheidbare Korrekturen („ihr Bild“ oder „ihre Bilder“).

5.4 Evaluation

Die Leistungsfähigkeit des integrativen Verfahrens wird anhand von zwei Grammatiken G_4 und G_5 untersucht, die den beiden Modellierungstechniken aus Abschnitt 5.2 entsprechen:

1. Grammatik G_4: Die Wohlgeformtheitsbedingungen, die durch Strukturabbildungen realisiert werden, sind nicht Bestandteil der bewerteten CD-Grammatik. Insbesondere kennt die Grammatik keine Regeln, die die Merkmale Kasus, Numerus, Genus und Person verwenden. Diese Grammatik befindet sich im Anhang A.

2. Grammatik G_5 : Es wird die Grammatik G_3 und die Strukturabbildungen von G_4 verwendet. Die Wohlgeformtheitsbedingungen der Strukturabbildungen sind, sofern möglich, Bestandteil der bewerteten CD-Grammatik.

Im Unterschied zur Grammatik G_4 enthält die Grammatik G_5 zusätzliche Präferenzregeln. Bestimmte Lesarten der Wortformen und Strukturen, in denen das Subjekt vor dem finiten Verb steht, werden in der Grammatik G_5 bevorzugt.

Für die Untersuchung werden 84 korrekte und 81 fehlerhafte Sätze verwendet (siehe Anhang A.5). Die Sätze sind manuell nach folgenden Gesichtspunkten erstellt worden:

- Es sollen für möglichst alle Phänomene der linguistischen Modellierung adäquate Strukturbeschreibungen erstellt werden.

- Es sollen möglichst viele Fehlerkonstellationen durch die Testsätze abgedeckt werden können.

Folgende linguistischen Phänomene werden von den korrekten Testsätzen abgedeckt:

- Syntaktischer und semantischer Valenzrahmen der Verben:
  Verben können im Präsens und im Perfekt stehen. Alle Komplemente können topikalisiert werden. Als Komplemente werden direkte und indirekte Objekte sowie Präpositionalgruppen unterstützt. Folgende Verben werden modelliert: freuen, geben, haben, lachen, laufen, legen, liegen, sehen, sein, suchen.

- Substantiv- und Präpositionalgruppen :
  Verwendung von Adjektiven, bestimmter und unbestimmter Artikel, Nullartikel, Possessiv- und Reflexivpronomen, Genitivergänzungen, adverbiale Ergänzungen (Ort und Umstand).

Die folgenden Fehlerkonstellationen sollen anhand der fehlerhaften Testsätze überprüft werden:

- Fehlende obligatorische Konstituenten:
  Fehlende Komplemente des Verbs können Subjekt, direktes oder indirektes Objekt, Reflexivpronomen oder eine Ortsangabe sein. In einer Substantiv- oder Präpositionalgruppe können Artikel oder Präpositionen fehlen. Weiterhin kann ein finites Verb fehlen.

- Überflüssige Konstituenten.

- Anordnung:
  Untersucht werden fehlerhafte Anordnungen in einer Substantiv- oder Präpositionalgruppe sowie fehlerhafte Stellungen des finiten Verbs oder des Partizips.

- Semantische Sortenverletzungen.

- Kongruenzfehler, Konjugation von Verben, reflexive Verben, Personal- und Reflexivpronomen und Rektion.

Die Strukturbeschreibungen der korrekten Sätze entsprechen für beide Grammatiken den Erwartungen. Für die Grammatik G_4 werden allerdings in einigen Fällen mehr optimale Strukturbeschreibungen als in G_5 geliefert. Das liegt an der fehlenden Präferenzregel, welche das Subjekt im Vorfeld des Satzes bevorzugt.[13]

Für alle fehlerhaften Sätze werden identische Diagnosen für dieselben Strukturbeschreibungen geliefert, allerdings generiert die Grammatik G_4 in einigen Fällen mehrere Strukturen. Die Strukturbeschreibungen stimmen bis auf eine Ausnahme mit den Erwartungen überein. In dem Satz

(25) * „Er legt den Ball den Tisch."

wird die Konstituente „den Tisch" als Genitivergänzung des Substantivs „Ball" angesehen. Besser wäre es, diese Konstituente durch eine Unterordnung unter die Wurzel als überflüssig zu kennzeichnen. Das CDG-Parsing liefert eine Regelverletzung für eine fehlende obligatorische Präpositionalgruppe. Die Möglichkeit, den Fehler durch eine fehlende Präposition in der Konstituente „den Tisch" zu erklären, ist mit der diagnostizierten Struktur kaum möglich. Außerdem liefert die Fehlersimulation drei Diagnosen, die die Kasusverletzung betreffen, und korrigiert die Phrase zu „des Tisches". Der Grund für die ungünstige Struktur liegt darin, daß die Unterordnung unter die Wurzel stärker bestraft wird als alle anderen Fehler. Das Problem kann gelöst werden, wenn die semantische Analyse eine Genitivanbindung so stark dispräferiert, daß eine Unterordnung unter die Wurzel die günstigste Lösung darstellt.

Ein Vorteil der Grammatik G_5 liegt darin, daß für die Konstruktion eines Fehlersimulationsmodells meist nur eine optimale Strukturbeschreibung berücksichtigt werden muß, wodurch einerseits der Aufwand der Fehlersimulation reduziert wird und andererseits zusätzliche Maßnahmen für die Präferenz von Strukturen nicht mehr nötig sind. Ob die Grammatik G_5 tatsächlich in einem Sprachlernsystem verwendet werden kann, hängt nicht zuletzt von der Effizienz des Analyseverfahrens ab.

---

[13] Die topologischen Felder im Deutschen werden z.B. in [Grewendorf 1996] behandelt. In [Schröder 1999] wird beschrieben, wie im Formalismus der CD-Grammatik topologische Felder modelliert werden können.

Für das CDG-Parsing wächst die Analysezeit einer Äußerung mit der Größe des Suchraums. Abbildung 5.5 zeigt diese Beziehung anhand der 165 Testsätze.[14] Der Suchraum wurde mit der Formel aus Abschnitt 5.1.1 berechnet, wobei die initiale Auswertung von unären harten Regeln berücksichtigt wurde. Das Experiment wurde mit der Grammatik G_5 durchgeführt. Die sehr starken Schwankungen der Analysezeit, insbesondere bei Sätzen mit großem Suchraum, hängen mit dem verwendeten Suchverfahren zusammen: Im besten Fall kann die Lösung in linearer Zeit, im schlechtesten Fall in exponentieller Zeit gefunden werden.

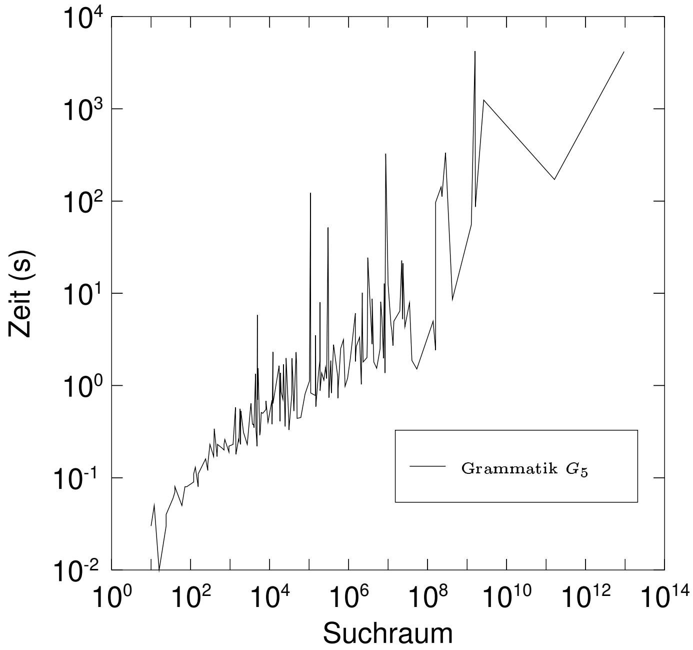

Abbildung 5.5: Beziehung zwischen Suchraumgröße und Analysezeit.

Es ist zu erwarten, daß sich aufgrund der reduzierten lexikalischen Mehrdeutigkeit der Grammatik G_4 eine Effizienzverbesserung ergibt. Allerdings muß in Betracht gezogen werden, daß die Fehlersimulation jetzt den Aufwand für die lexikalische Disambiguierung trägt. Abbildung 5.6 vergleicht die Größe des Suchraums der Grammatik G_5 mit der Grammatik G_4. Die Testsätze sind nach der benötigten Analysezeit für G_5 geordnet. Die Reduktion des Suchraums aufgrund der Reduktion der lexikalischen Mehrdeutigkeit ist deutlich zu erkennen.

In Abbildung 5.7 wird die Analysezeit beider Grammatiken miteinander verglichen. Es ist dabei zu erkennen, daß die benötigte Zeit der Grammatik G_4, welche keine morphologischen Merkmale verwendet, gerade für schwierige Sätze deutlich unter der Analysezeit der Grammatik G_5 liegt.

[14] In den folgenden Abbildungen werden für die Analysezeiten und die Anzahl der Zustände im Suchraum logarithmische Achseneinteilungen verwendet.

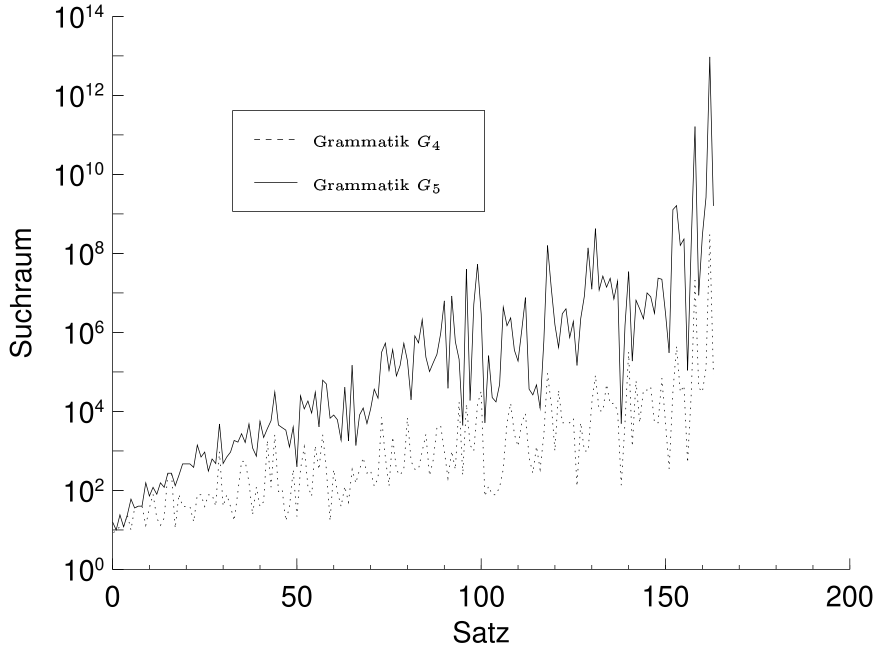

Abbildung 5.6: Vergleich der Suchraumgröße.

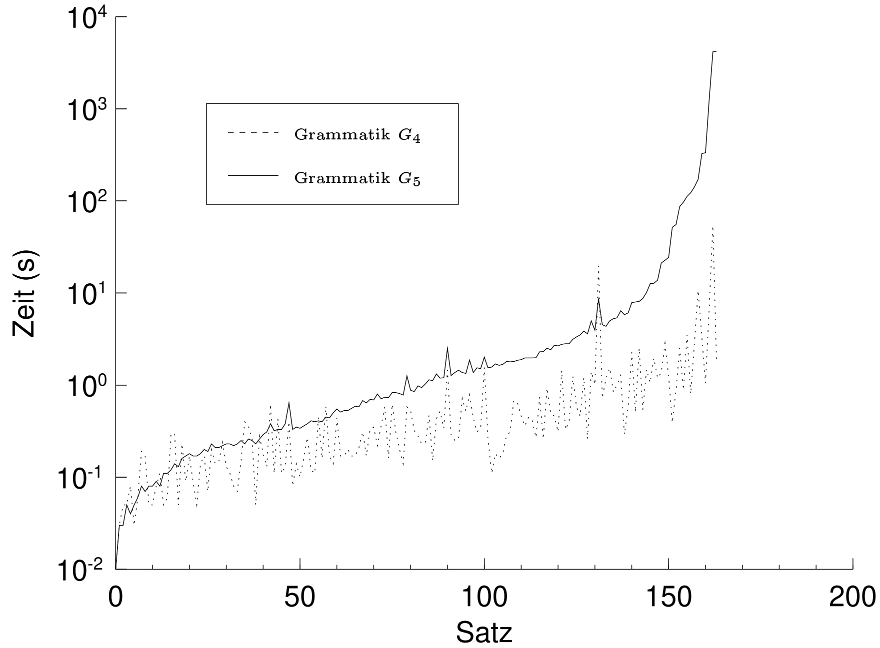

Abbildung 5.7: Vergleich der Analysezeiten.

Die Mengenrepräsentation von Merkmalswerten im Lexikon der Fehlersimulation ist kompakter als die Darstellung im CDG-Parsingverfahren. Dadurch sollte die Fehlersimulation wesentlich weniger Lesarten berücksichtigen müssen. Allerdings werden vom CDG-Parsing für die Grammatik G_4 meist mehrere optimale Strukturbeschreibungen geliefert, die alle analysiert werden müssen.[15] Die Abbildung 5.8 vergleicht die Analysezeiten der Fehlersimulation.

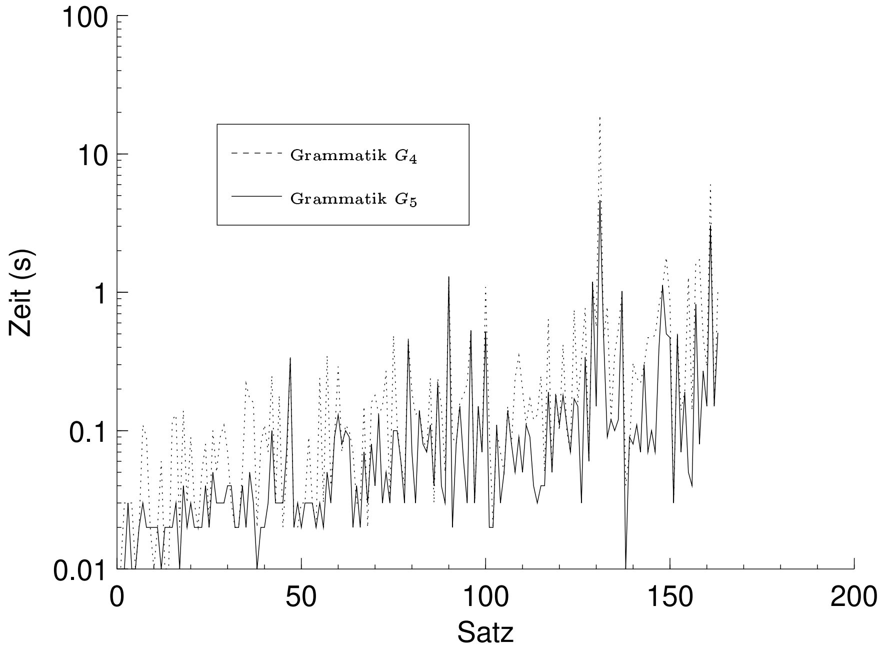

Abbildung 5.8: Vergleich der Analysezeiten der Fehlersimulation.

Da die Grammatik G_4 aufgrund fehlender Disambiguierungsinformation mehrere Strukturen simulieren muß, benötigt die Analyse auch mehr Zeit. Allerdings können die meisten Sätze innerhalb einer Sekunde verarbeitet werden. Die Sätze, in denen die Fehlersimulation mehrere Sekunden benötigt, sind Sätze mit einer großen Anzahl von Lesarten und mehreren verletzten Komponenten, wie z.B. in

(26) * „Ein kleine Mann sieht der Berg mit den Fernglas."

Die Grammatik G_4 (G_5) benötigt 6 (3) Sekunden für die Fehlersimulation, wobei die minimale Diagnose drei verletzte Komponenten enthält. Für das CDG-Parsing kehren sich die Verhältnisse um: Die Grammatik G_4 benötigt 1 Sekunde, während die Grammatik G_5 20 Minuten benötigt. Solange keine effizienteren Verfahren, die ein ausreichendes Maß an Korrektheit garantieren, verfügbar sind, scheint eine Verwendung der Grammatik G_5 in Sprachlernsystemen nicht sinnvoll zu sein. Die Ursache für die schlechte Performanz kann allerdings auch an einer nicht adäquaten Modellierung von Regeln liegen, wobei allerdings nicht klar ist, welche Richtlinien ein Grammatikschreiber zu erfüllen hat, um eine

[15]Im Durchschnitt werden 1.8 Strukturbeschreibungen für jeden Testsatz generiert.

effiziente Verarbeitung ermöglichen zu können. Die Berücksichtigung des Analyseverfahrens für den Entwurf einer Grammatik ist sicherlich keine günstige Voraussetzung für die Akzeptanz des Formalismus.

5.5 Ein Entwicklungswerkzeug

Um die Ergebnisse des integrativen Verfahrens, insbesondere die Adäquatheit der Strukturbeschreibungen und die Qualität der Diagnosen, beurteilen zu können, wurde das Programm *Grammar Diagnosis* entwickelt, welches die Ergebnisse der Analyse in grafischer Darstellung zur Verfügung stellt. Das Programm ist als ein Entwicklungswerkzeug zu verstehen. Es gestattet die visuelle Validierung einer entwickelten Grammatik im integrativen Verfahren.

Das System wurde in der Programmiersprache Java ([Flanagan 1998]) realisiert. Java bietet ein hohes Maß an Portabilität, wobei insbesondere grafische Benutzeroberflächen auf den meisten verfügbaren Betriebssystemen ohne Programmänderung lauffähig sind. Weiterhin kann die Unterstützung von Java durch zahlreiche Bibliotheken, die Bestandteil der Java-Entwicklungsumgebung sind, als vorbildlich angesehen werden. Insbesondere ist ab der Version 2 von Java eine Klassenbibliothek für die Erstellung grafischer Benutzeroberflächen fester Bestandteil.[16]

Die funktionale Komponente des integrativen Verfahrens ist in C++ als dynamische Bibliothek implementiert. Die Einbindung von C- und C++-Bibliotheken in Java erfolgt über das *Java Native Interface* ([Gordon 1998]) und erlaubt sowohl Java-Programmen, Funktionen aus einer C- bzw. C++-Bibliothek aufzurufen als auch innerhalb des C- bzw. C++-Programmcodes Java-Objekte von beliebigen Klassen zu instanziieren und deren Methoden aufzurufen.

Der Funktionalität des integrativen Verfahrens wird dem Entwicklungswerkzeug über eine Klasse zu Verfügung gestellt, die die Schnittstelle zwischen C++- und Java-Programm realisiert. Die Methoden gestatten eine einfache Benutzung des Systems, ohne sich mit den Details der Implementierung auseinander setzen zu müssen. Insbesondere für zukünftige Entwicklungen, die das vorgestellte Verfahren in einem Sprachlernsystem verwenden möchten, ist eine einfache Schnittstelle mit einer überschaubaren Menge von Methoden von Vorteil.[17] Eine vereinfachte Beschreibung dieser Schnittstelle befindet sich im Anhang B.

In Abbildung 5.9 ist ein Bildschirmausdruck des implementierten Entwicklungswerkzeugs zu sehen. Nachdem im Eingabefeld ein Satz eingegeben bzw. ein Testsatz ausgewählt wurde, wird die Äußerung analysiert. Das Ergebnis ist eine Menge von Fehlerbeschreibungen, deren Bestandteile in der linken Auswahlliste angezeigt werden. Dies sind die verletzten Regeln der CD-Grammatik in der optimalen Strukturbeschreibung sowie die minimalen Diagnosen des konstruierten Fehlersimulationsmodells. In der Auswahlbox über dieser Liste kann die Analyseebene des zu zeichnenden Dependenzbaums ausgewählt werden. Wird

[16] Die Klassenbibliothek hat den Namen *Swing*. Sie wird z.B. in [Eckstein 1998] beschrieben.

[17] In [Alpert 1998] wird eine technische Realisierung eines auf Java basierenden und über das Internet verfügbaren Lernsystems am Beispiel eines Mathetutors beschrieben. Die vorgestellte Architektur kann für den Entwurf eines web-basierten Fremdsprachenlernsystems verwendet werden.

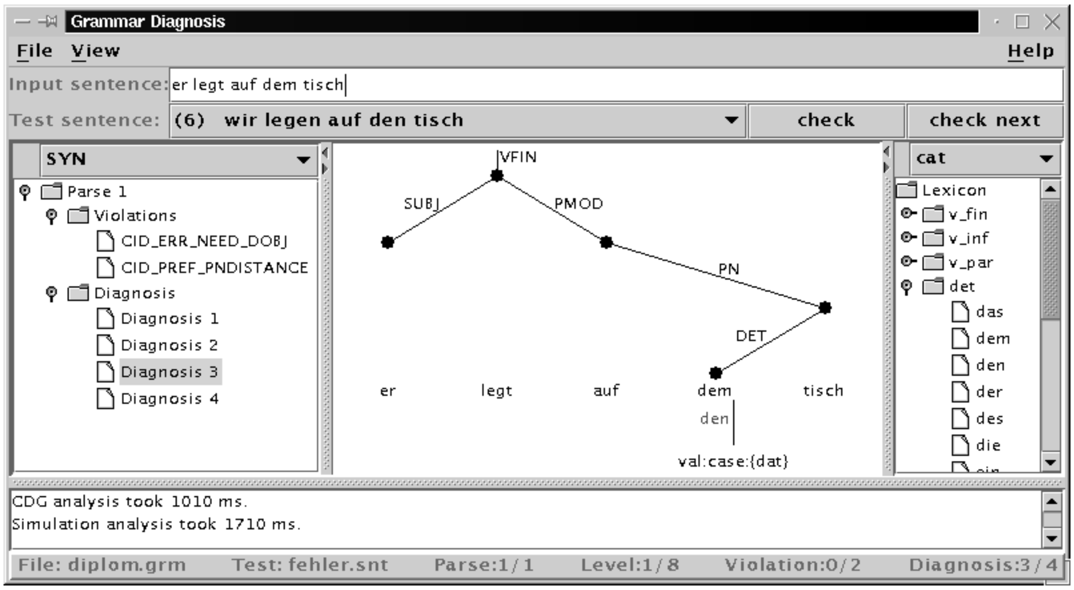

Abbildung 5.9: Programm *Grammar Diagnosis*

eine Regelverletzung in der Liste selektiert, werden die beteiligten Kanten in der Baumstruktur grafisch hervorgehoben. Die minimalen Diagnosen, die häufig aus mehreren als fehlerhaft angenommenen Komponenten bestehen, werden numeriert dargestellt. Wenn eine minimale Diagnose selektiert wurde, werden die fehlerhaften Komponenten visualisiert, indem für jede Komponente eine Kante zwischen den beteiligten Wortformen gezeichnet wird. Diese Kante wird mit einer abgekürzten Bezeichnung der Komponente beschriftet. Der Korrekturvorschlag einer Faktenfehlerdiagnose wird unter der zu korrigierenden Wortform dargestellt. Die Merkmalsstrukturen der Wortformen für eine minimale Diagnose, die sich aus dem Ergebnis der Fehlersimulation ergeben, können durch einen Mausklick auf eine Wortform angezeigt werden. Die Merkmalsstrukturen vor der Simulation können im Lexikon betrachtet werden.

Bei der Darstellung ist zu beachten, daß die Liste der Regelverletzungen und die Liste der Diagnosen unterschiedlich interpretiert werden: Die erstere enthält alle verletzten Regeln der CD-Grammatik, wird also als Konjunktion interpretiert, während die zweite alle alternativen Sichten auf einen Fehler im Verfahren der Fehlersimulation darstellt. Im Verfahren des CDG-Parsings werden sowohl für verschiedene Lesarten als auch für unterschiedliche Mengen von Regelverletzungen weitere Strukturbeschreibungen erzeugt, obwohl die strukturellen Beziehungen zwischen den Wortformen durchaus identisch sein können. In vielen Fällen sind allerdings mehrere Strukturbeschreibungen unerwünscht, so daß deren Existenz auf Modellierungsfehler in der Grammatik hindeuten kann.

Kapitel 6

Fazit

In diesem Kapitel werden die Ergebnisse der vorliegenden Arbeit zusammengefaßt, die noch bestehenden Probleme diskutiert und ein Ausblick auf zukünftige Arbeiten gegeben.

6.1 Ergebnisse

Das Ziel, eine Kopplung des CDG-Parsings mit der Fehlersimulation zu realisieren, konnte erreicht werden. Die Diagnosequalität der Fehlersimulation kann für die Verarbeitung von freien Äußerungen ausgenutzt werden, indem aus einer optimalen Strukturbeschreibung ein Fehlersimulationsmodell konstruiert wird. Die Konstruktionsvorschriften werden durch den vorgeschlagenen Formalismus der Strukturabbildungen realisiert. Insbesondere sind auch dann verschiedene Fehlererklärungen verfügbar, wenn bewertete CD-Grammatiken so formuliert werden, daß möglichst nur genau eine optimale Strukturbeschreibung als Lösung angeboten wird. Folgende Leistungen sind für die im Rahmen der Fehlersimulation modellierbaren linguistischen Phänomene verfügbar:

- Es können verschiedene Fehlererklärungen für eine fehlerhafte Äußerung generiert werden.
  - Verschiedene Kombinationen von Regelverletzungen sind diagnostizierbar.
  - Unterschiedliche Lesarten werden berücksichtigt.
  - Die Merkmalsstrukturen der verwendeten Wortformen können für eine Verbalisierung des Fehlers verwendet werden.
  - Es sind zwei Sichtweisen auf den Fehler möglich: Fakten- und Regelfehler.
  - Aus einer Faktenfehlerdiagnose können Korrekturvorschläge generiert werden.
- Transitive Bedingungsketten können modelliert und effizient verarbeitet werden.

- Die Strukturabbildungen erlauben die Modellierung von Wohlgeformtheitsbedingungen für weit entfernte, voneinander abhängige Konstituenten. Dies wurde am Beispiel von Reflexivpronomen demonstriert.[1] Damit kann der fehlende Transportmechanismus und die Beschränkung auf maximal binäre Regeln des CDG-Parsings zum Teil kompensiert werden.[2]

Die Versuche mit der meist vollständig disambiguierenden Grammatik G_5 konnten die Leistungsmerkmale der Fehlersimulation leider nur unter erheblichen Effizienzproblemen ausnutzen. Der Grund für die schlechte Performanz liegt an der Anzahl der Variablen im *Constraint Satisfaction* Problem, da jede Wortform durch die Verwendung von morpho-syntaktischen Merkmalen sehr viele Lesarten besitzen kann. Interessanterweise war es für die betrachteten Phänomene möglich, eine CD-Grammatik ohne morpho-syntaktische Merkmale zu entwickeln (Grammatik G_4 im Anhang A), die trotzdem noch in der Lage ist, sowohl strukturelle Fehler zu entdecken als auch ein ausreichendes Maß an Strukturreduzierung zu leisten, so daß die nachgeschaltete Fehlersimulation nur unwesentlich mehr Zeit benötigt als im Falle einer vollständig disambiguierenden CD-Grammatik. Dabei konnte für schwer analysierbare Sätze eine erhebliche Effizienzverbesserung beobachtet werden.

Als Ergebnis dieser Arbeit liegt eine Implementation des integrativen Verfahrens vor. Die Komplexität des Gesamtsystems ist beträchtlich. Es besteht insgesamt aus 44000 Zeilen Programmcode. Davon sind allein 50 Prozent für das CDG-Parsingverfahren verantwortlich, welches als Beistellung in Form einer C-Bibliothek verwendet werden konnte. Auf die Fehlersimulation und die Realisierung der Schnittstelle entfallen 40 Prozent des Gesamtsystems. Der Anteil des Entwicklungswerkzeugs ist mit nur 10 Prozent relativ bescheiden.

6.2 Probleme

Die Entwicklung einer CD-Grammatik, die einerseits keine morpho-syntaktischen Merkmale verwendet und andererseits möglichst wenige optimale Strukturbeschreibungen liefert, hat sich als nicht ganz einfach erwiesen. Die Schwierigkeiten entstehen u.a. dadurch, daß Strukturen nur deshalb bevorzugt werden könnten, weil die konfliktverursachende Regel nicht mehr in der CD-Grammatik, sondern im Fehlersimulationsmodell enthalten ist. Aus diesem Grund können z.B. einige Präferenzregeln nicht verwendet werden:

- Strukturen, in denen das Subjekt vor dem finiten Verb steht, können nicht mehr bevorzugt werden.

- Die Bestrafung von Projektionsverletzungen kann nur eingeschränkt verwendet werden. Zum Beispiel würde in dem Satz

[1] In Abschnitt 5.3.2 modellieren die Strukturabbildungen (a_7) und (a_8) die Kongruenzforderungen zwischen Subjekt und Reflexivpronomen.

[2] Eine Regel, die die Identität von L-Knoten fordert, kann in der Fehlersimulation nicht modelliert werden. Insbesondere Bedingungen, die Beziehungen zwischen semantischen und syntaktischen Strukturen herstellen, benötigen solche Forderungen (vgl. Abschnitt 5.1.3).

(27) „Dem Mann hat das Kind ein Buch gegeben.“

das Substantiv „Mann“ als Subjekt angesehen werden, da die Annahme, „Kind“ wäre das Subjekt, zu einer Projektionsverletzung führt.

Es sind sowohl Anordnungsregeln und Anordnungspräferenzen als auch semantische Regeln für die sortalen Beschränkungen von Valenzfüllern dringend erforderlich, um die Anzahl der optimalen Strukturbeschreibungen zu reduzieren. Ob ein Verzicht auf Merkmale, die maßgeblich für die lexikalische Mehrdeutigkeit verantwortlich sind, auch für die Modellierung von weiteren linguistischen Phänomenen möglich ist, bleibt ungeklärt. Die Vorteile einer effizienteren Strukturanalyse können schnell verloren gehen, wenn zu viele optimale Strukturbeschreibungen für die Fehlersimulation verbleiben.

Obwohl das Zeitverhalten für die Testsätze im vernünftigen Rahmen bleibt, muß erwähnt werden, daß die Fehlersimulation für Äußerungen mit vielen Lesarten sehr viel Rechenzeit benötigen kann. Zum Beispiel sind Äußerungen mit vier oder mehr Adjektiven aufgrund der lexikalischen Mehrdeutigkeit kaum noch zu verarbeiten. Für das CDG-Parsing können auch dann Effizienzprobleme auftreten, wenn jede Wortform nur eine Lesart besitzt. Zum Beispiel führt die strukturelle Mehrdeutigkeit von Präpositionalgruppen meist zu erheblichen Effizienzproblemen. Daher sollten Lernsysteme, die das integrative Verfahren verwenden wollen, nur Äußerungen mit einer vorher bekannten, maximalen Analysezeit verarbeiten. Während die Analysezeit der Fehlersimulation über die lexikalische Mehrdeutigkeit abgeschätzt werden kann, ist für das CDG-Parsing mit dem verwendeten Suchverfahren *branch and bound* keine derartige Abschätzung möglich. Es kann versucht werden, die Größe des Suchraums zu verwenden. Allerdings sind für verschiedene Äußerungen bei gleicher Suchraumgröße erhebliche Schwankungen zu erwarten.

Die verwendeten Testsätze erfüllen in erster Linie den Zweck, die Funktionalität des Verfahrens und die Korrektheit und Leistungsfähigkeit der Grammatik zu überprüfen. Insbesondere sollen Fehler in ungrammatischen Äußerungen, die bestimmten Fehlerklassen zuzuordnen sind, identifiziert werden können. Die Beurteilung der Diagnosequalität erfolgte anhand der ermittelten Regelverletzungen der CD-Grammatik und der minimalen Diagnosen der Fehlersimulation. Folgende Gesichtspunkte bleiben dabei unberücksichtigt:

- Die Testsätze sind in keiner Weise repräsentativ in Bezug auf die tatsächlich von einem Schüler im Unterricht oder in einem prototypischen Lernsystem produzierten Äußerungen.

- Die symbolische Repräsentation der Regelverletzungen und der minimalen Diagnosen ist sicher nicht ausreichend, um den Nutzen für den Schüler beurteilen zu können.

- Der Vorteil, verschiedene Fehlererklärungen für eine ungrammatische Äußerung generieren zu können, kann erst dann zuverlässiger beurteilt werden, wenn ein maschinelles Auswahlverfahren für Diagnosen verfügbar ist.

- Das Verfahren der Fehlersimulation kann u.U. sehr viele minimale Diagnosen liefern. Es muß festgelegt werden, nach welchen Entscheidungskriterien bestimmte Diagnosen bevorzugt und andere vollständig ignoriert werden sollen.

Die Modellierung einer Grammatik für das integrative Verfahren ist weitgehend antizipationsfrei. In der entwickelten CD-Grammatik werden allerdings einige Regeln verwendet, die den Fehlerfall explizit formulieren:

- Eigennamen werden gewöhnlich ohne Artikel verwendet.
- Wenn in einer Äußerung das finite Verb fehlt, jedoch das zweite Partizip vorhanden ist, soll eine vernünftige Analyse möglich sein. In dem Satz

  (28) * „Ich gegeben ihm das Buch“

  wird diagnostiziert, daß der Schüler nicht die finite Form des Verbs verwendet hat.

In beider Fällen werden fehlerhafte Unterordnungen der beteiligten Wortformen erlaubt. Weitere weiche Regeln bestrafen den Fehlerfall. Insbesondere die Fehlerklasse, die sich durch eine Verwechslung von Wortarten ergibt, kann nicht ohne eine Antizipation des Fehlerfalls modelliert werden.

6.3 Ausblick

Für das integrative Verfahren sind einige Erweiterungen möglich:

- Es werden Verfahren für die Fehlerverbalisierung benötigt, die eine vorzugsweise deklarative Beschreibung der Fehlerselektions- und Reduktionsstrategie sowie des Verbalisierungswissens erlauben.
- Für einen Einsatz des Verfahrens in einem Fremdsprachenlernsystem sind Erweiterungen bezüglich des Umfangs der Wissensrepräsentation erforderlich. Insbesondere ist es wünschenswert, inhaltliche Fehler diagnostizieren zu können. Dafür ist neben einer umfangreicheren semantischen Modellierung eine Repräsentation des Weltmodells erforderlich. Die binären Relationen zwischen Wortformen im CD-Formalismus begrenzen die Möglichkeiten einer Domänenmodellierung. Hier ist u.U. eine Kopplung mit einer geeigneten Wissensrepräsentationssprache notwendig. Insbesondere wäre eine Pragmatikmodellierung interessant, die eine in Abhängigkeit der Schülerantwort *veränderliche* Wissensbasis für den Diagnoseprozeß verwenden kann.
- Die Diagnosequalität der Fehlersimulation kann durch zusätzliche Modellierungskonstrukte verbessert werden.
- Um die Effizienz des CDG-Parsingverfahren zu verbessern, können andere Lösungsverfahren untersucht werden, die eventuell auch eine bessere Abschätzung der Analysezeit ermöglichen.

- Für das Verfahren der Fehlersimulation kann versucht werden, die Berechnung der minimalen Diagnosen für mehrere Strukturbeschreibungen und die Analyse von Äußerungen mit vielen Lesarten effizienter zu gestalten.

Im folgenden werden Vorschläge zur Verbesserung der Diagnosequalität und der Effizienz des integrativen Verfahrens vorgestellt.

Verbesserung der Diagnosequalität

Das Verfahren der Fehlersimulation kann um Prädikate für die Formulierung von Anordnungsbeziehungen erweitert werden. Zum Beispiel kann ein Prädikat $\mathsf{precedes}(A, B)$ eingeführt werden, welches besagt, daß die Position der Modellvariablen A kleiner ist als die Position von B. Bisher sind vergleichbare Anordnungsregeln nur in der CD-Grammatik enthalten. Da diese Regeln allerdings wichtig sind, um möglichst wenige optimale Strukturbeschreibungen zu erhalten, ist ein vollständiger Verzicht auf Anordnungsregeln in der CD-Grammatik nicht sinnvoll. Daher ist eine redundante Wissensmodellierung vorzuziehen.

Der Vorteil von Anordnungsprädikaten in der Fehlersimulation ist, daß es damit möglich wird, einen Korrekturvorschlag durch *Umordnung* der Wortformen zu ermitteln. Dazu ist im Modus der Faktenfehlersimulation die Normal- und Fehlerfallsemantik des precedes-Prädikats anzugeben. Die notwendigen Formeln sind in [Menzel 1992a] beschrieben. Es wäre interessant, zu untersuchen, inwieweit eine Korrektur von Stellungsfehlern in freien Äußerungen möglich ist. Außerdem stellt sich die Frage, ob eine CD-Grammatik z.B. Stellungsfehler des finiten Verbs oder des zweiten Partizips derart durch Regelverletzungen beschreiben kann, daß eine angemessene Verbalisierung möglich ist. Die Einschränkung auf maximal binäre Regeln erfordert meist eine zusätzliche Analyseebene für die Behandlung von topologischen Feldern. In der Fehlersimulation können beliebig viele Wortformen in der Struktur miteinander in Beziehung gesetzt werden.

Eine andere Erweiterung, die ebenfalls die Generierung eines Korrekturvorschlags betrifft, wurde schon im Zusammenhang mit dem Schutzmechanismus von Merkmalen im Modus der Faktenfehlersimulation diskutiert (vgl. Abschnitt 3.3.6). Insbesondere wurde für das Genusmerkmal eines Substantivs ein zweistufiger Simulationsprozeß vorgeschlagen. Erst wenn ein mit einem speziellen Schutzoperator markiertes Prädikat, z.B. $\mathsf{agree}(*\mathsf{N}, \mathsf{DET}, \mathsf{gender})$, verletzt und ein Korrekturvorschlag vom Schüler gewünscht wird, wird der Schutzmechanismus in einer erneuerten Faktenfehlerdiagnose aktiviert.

Effizienzverbesserung

Für die robuste Verarbeitung von freien Äußerungen können im CDG-Parsing Lösungsverfahren verwendet werden, die auf Kosten der Korrektheit und der Vollständigkeit eine effizientere Analyse ermöglichen. Als Lösungsverfahren kann z.B. das transformationsbasierte Verfahren aus [Foth 2000] verwendet werden. Die Experimente in [Foth 1999b] zeigen deutliche Effizienzverbesserungen. Es bleibt zu prüfen, ob das Lösungsverfahren z.B. für die betrachteten Testsätze eine qualitativ vergleichbare Analyse ermöglicht. Die vorliegende Implementation benötigt eine Erweiterung, die die Auswahl eines Lösungsverfahren gestattet.

Die Fehlersimulation muß für die Grammatik G_4 meist mehrere Modelle simulieren. Für alle Modelle werden nacheinander die minimalen Diagnosen bestimmt. Erst danach kann entschieden werden, welche Modelle tatsächlich die geringsten fehlerhaften Komponenten aufweisen. Eine (pseudo-)parallele Verarbeitung, die auf Kosten eines höheren Speicherbedarfs geht, könnte die Analyse einzelner Modelle für eine ausgewählte Lesart frühzeitig abbrechen, wenn es für die momentan betrachtete Variante definitiv keine bessere als die bisher beste Diagnose gibt.

Ein vermutlich deutlicherer Effizienzvorteil könnte erreicht werden, wenn nicht alle möglichen Lesarten der Äußerung simuliert werden müßten. Die Betrachtung aller Lesarten ist aber für die Vollständigkeit des Verfahrens notwendig. Es wurde zwar versucht, die Unabhängigkeit von Teilnetzen im Komponentennetz systematisch auszunutzen. Die Berechnung der Diagnose eines Teilmodells war auch in erheblich weniger Zeit möglich, aber das Zusammensetzen der Gesamtlösung aus den Teillösungen hat so viel Zeit benötigt, daß die gesamte Analysedauer sogar größer war als zuvor. Je nachdem, in welchem Rahmen das integrative Verfahren verwendet werden soll, sind weitere Untersuchungen zur Effizienzverbesserung notwendig.

Anhang A

Repräsentation sprachlichen Wissens

Im folgenden wird die formale Syntax der entwickelten Wissensbeschreibungssprache angegeben. Eine Beschreibung der Definition von Merkmalen, Merkmalsstrukturen und dem Lexikon findet sich in [Stockfleth 1999]. Der Formalismus wurde um die Indizierung von Lexikoneinträgen für die Korrekturgenerierung und um Konstrukte für die strukturabhängige Generierung von Fehlersimulationsmodellen erweitert.[1] Der folgende Abschnitt beschreibt Teilaspekte der Repräsentation des verwendeten Lexikons. Danach wird die verwendete CD-Grammatik vollständig angegeben. Die formale Syntax für die Beschreibung von bewerteten CD-Grammatiken sowie eine Erläuterung der Notation kann in [Schröder 1998] nachgeschlagen werden. Anschließend werden die verwendeten Strukturabbildungsregeln erläutert. Der letzte Abschnitt gibt die für die Evaluation des integrativen Ansatzes verwendeten Testsätze an.

A.1 Formale Syntax der Wissensbeschreibung

Kommentare werden mit der Zeichenfolge `//` eingeleitet und erstrecken sich auf die gesamte Zeile. Bezeichner (<id>) beginnen mit einem Zeichen aus der Menge {`a-z`, `A-Z`}. Danach können beliebig viele Zeichen aus der Menge {`a-z`, `A-Z`, `0-9`, `-`, `_` } folgen. Zeichenketten (<string>) beginnen und enden mit dem Anführungszeichen (`"`). Innerhalb der Anführungszeichen können beliebige, darstellbare Zeichen folgen. Das leere Wort wird mit <empty> bezeichnet.

Wissensbeschreibung

knowlegde-base	::= *declaration declaration-list*
declaration-list	::= <empty> | *declaration declaration-list*
declaration	::= `include` <string> |
	slot-declaration |

[1] Die Definition von statischen Fehlersimulationsmodellen, die Definition von Übungen und die Konstrukte für die Fehlerverbalisierung nach [Stockfleth 1999] werden nicht mehr unterstützt.

		template-declaration |
		lexicon-declaration |
		correction-declaration |
		mapping-declaration

Merkmale

slot-declaration	::=	`SLOTS` *slot-value-list* `END`
slot-value-list	::=	<empty> | *slot-value slot-value-list*
slot-value	::=	*slot* `:` *value*
value	::=	`{ }` | `{` *atom-seq* `}`
atom-seq	::=	*atom* | *atom* `,` *atom-seq*
slot	::=	<id>
atom	::=	<id>

Merkmalsstrukturen

f-struct	::=	`[` *slot-value-list* `]`
template-declaration	::=	`TEMPLATE` *template-list* `END`
template-list	::=	<empty> | *template template-list*
template	::=	*f-struct-name* `:` *f-struct*
f-struct-name	::=	<id>

Lexikon

lexicon-declaration	::=	`LEXICON` *wordform-list* `END`
wordform-list	::=	<empty> | *wordform wordform-list*
wordform	::=	*word* `:` *f-struct-seq*
f-struct-seq	::=	*ext-f-struct* | *ext-f-struct* `,` *f-struct-seq*
ext-f-struct	::=	*f-struct* | *f-struct-name* | *ext-f-struct* `+` *ext-f-struct*
word	::=	<id>

Korrekturen

correction-declaration	::=	`CORRECTION` *correction-list* `END`
correction-list	::=	<empty> | *correction correction-list*
correction	::=	<id> `:` *word-seq*
word-seq	::=	*value*

Strukturabbildungen

mapping-declaration	::=	`CDG-MAPPING` *mapping-stmt-list* `END`
mapping-stmt-list	::=	<empty> | *mapping-stmt mapping-stmt-list*
mapping-stmt	::=	*use-grammar-decl* |
		default-level-decl |
		mapping-rules-decl

```
use-grammar-decl     ::= USE-GRAMMAR <string>

default-level-decl   ::= DEFAULT-LEVEL level

mapping-rules-decl   ::= MAP-RULES mapping-rule-list END
mapping-rule-list    ::= <empty> | mapping-rule mapping-rule-list
mapping-rule         ::= predicate-list => component-list

predicate-list       ::= <empty> | predicate predicate-list
predicate            ::= pred-modifies |
                         pred-value |
                         pred-no-modifier
pred-modifies        ::= modifies ( node , ext-label , node opt-level )
pred-value           ::= value ( node , slot , ext-value )
pred-no-modifier     ::= no-modifier ( node , ext-label , node opt-level )

component-list       ::= <empty> | ext-component component-list
ext-component        ::= component | ! component
component            ::= restrict ( ext-variable , slot , value ) |
                         agree ( ext-variable , ext-variable , slot ) |
                         agree ( ext-variable , ext-variable , slot , slot )
opt-level            ::= <empty> | , level
ext-label            ::= variable | label
ext-variable         ::= variable | ! variable
node                 ::= <id>
variable             ::= <id>
label                ::= <string>
constraint           ::= <string>
level                ::= <string>
```

A.2 Lexikon

Dieser Abschnitt beschreibt einen Ausschnitt des verwendeten Lexikons. Die Modellierung verwendet grammatikalisches Wissen aus [Eisenberg 1998], insbesondere die Klassifizierung von Substantiven und Adjektiven.

A.2.1 Relevante Merkmale der CD-Grammatik

Die entwickelte CD-Grammatik verwendet keine morpho-syntaktischen Merkmale, da in der gewählten Modellierung die Fehlersimulation für die Realisierung der zugehörigen Phänomene verantwortlich ist. Folgende Merkmale sind für die CD-Grammatik relevant:

```
SLOTS
   cat:{v_fin,v_inf,v_par,det,adj,n,
        pers_pron,refl_pron,poss_pron,name,prep }
   arg1:{IOBJ,DOBJ,PMOD,VINF,VPAR}
```

```
   arg2:{IOBJ,DOBJ,PMOD,VINF,VPAR}
   sem:{animate,inanimate}
   sort0:{animate,inanimate}
   sort1:{animate,inanimate}
   sort2:{animate,inanimate}
   nodet:{f,t}
END
```

Das Merkmal `cat` definiert die unterstützten Wortarten. Es werden dabei drei verschiedene Wortarten für unterschiedliche Formen des Verbs angenommen, `v_fin` für finite Verben, `v_inf` für Infinitive und `v_par` für das zweite Partizip. Es werden Personalpronomen (`pers_pron`), Reflexivpronomen (`refl_pron`) und Possessivpronomen (`poss_pron`) unterstützt. Weiterhin werden Substantive (`n`), Adjektive (`adj`), Präpositionen (`prep`) und Eigennamen (`name`) verwendet.

Die Merkmale `arg1` und `arg2` modellieren obligatorische Ergänzungen von Verben. Der Wert dieser Merkmale bestimmt die syntaktische Funktion (Kantenbeschriftung in der CD-Grammatik) der Komplemente, wobei für jedes Verb implizit ein Komplement für das Subjekt gefordert wird.

Die Semantik wird nur insoweit unterstützt, als daß die selektionalen Restriktionen der Argumente eines Verbs modelliert werden. Dabei wird nur zwischen belebten und unbelebten Komplementen unterschieden. Das Merkmal `sort0` enthält die Restriktion für das Subjekt, die Merkmale `sort1` und `sort2` entsprechen den Restriktionen der Argumente `arg1` und `arg2`. Das Merkmal `sem` wird von den möglichen Komplementen realisiert (Substantive, Pronomen und Namen).

Das Merkmal `nodet` markiert Substantive, die ohne Artikel verwendet werden können.

Der Lexikoneintrag für die Wortformen des Satzes „ich gebe ihm das buch“ ohne Berücksichtigung der morphologischen Merkmale sieht dann wie folgt aus:

```
LEXICON
   ich  : [cat:{pers_pron} sem:{animate}]
   gebe : [cat:{v_fin} sort0:{animate}
           arg1:{IOBJ} sort1:{animate}
           arg2:{DOBJ} sort2:{inanimate}]
   ihm  : [cat:{pers_pron} sem:{animate}]
   das  : [cat:{det}]
   buch : [cat:{n} sem:{inanimate} nodet:{f}]
END
```

A.2.2 Relevante Merkmale der Fehlersimulation

Die Fehlersimulation benötigt zusätzlich folgende morpho-syntaktischen Merkmale:

```
SLOTS
   num:{sing,plur}
```

```
   snum:{sing,plur}
   case:{nom,gen,dat,acc}
   dobj_case:{nom,gen,dat,acc}
   gender:{masc,fem,neutr}
   sgender:{masc,fem,neutr}
   pers:{1,2,3}
   spers:{1,2,3}
   atype:{weak,mixed,strong}
   atype_sel:{weak,mixed,strong}
   pcase:{dat,acc,gen}
   sel:{loc,dir}
   konj:{sein,haben}
   ref : {r1,r2,r3,r4}
   ref_comp : {r1,r2,r3,r4}
   reflexiv: {0,DOBJ}
END
```

Neben den klassischen Merkmalen für Kasus, Numerus, Genus und Person (`case`,`num`,`gender`,`pers`) gibt es noch spezielle Merkmale für Possessivpronomen, um die Kongruenz mit dem Antezedenten im Numerus, Genus und in der Person überprüfen zu können (`snum`,`sgender`,`spers`).

Der Kasusrahmen von Verben wird für die Ergänzung durch ein direktes Objekt mit dem Merkmal `dobj_case` modelliert. Nur für das Verb „sein“ ist der Wert Nominativ, ansonsten immer Akkusativ. Die Kasusforderung für das Subjekt und das indirektes Objekt sind in den Abbildungsregeln kodiert. Das Merkmal `konj` gibt die Konjugationsform eines Verbs an.

Die Merkmale `pcase` und `sel` definieren die Eigenschaften von Präpositionen, unterschiedliche Kasusforderungen für lokale und direktionale Ortsangaben zu besitzen. Die Wortform „auf“ wird wie folgt repräsentiert:

```
LEXICON
   auf : [cat:{prep} sel:{loc} pcase:{dat}],
         [cat:{prep} sel:{dir} pcase:{acc}]
END
```

Die erlaubten Kombinationen von Artikeln und Adjektiven werden mit den Merkmalen `atype` und `atype_sel` modelliert. Die Adjektive werden in drei Klassen eingeteilt (schwach, gemischt, stark). Ein Artikel definiert mit dem Wert des Merkmals `atype_sel`, welche Adjektivklassen mit ihm verträglich sind.

Das Merkmal `reflexiv` markiert reflexive Verben. Als Komplement kommt nur ein direktes Objekt in Frage.

Die Merkmale `ref` und `ref_comp` werden für die Forderung verwendet, daß ein Pronomen als Komplement eines Verbs nur dann Personalpronomen sein kann, wenn es sich nicht auf das Subjekt bezieht. Ansonsten soll das Pronomen immer als Reflexivpronomen angesehen werden. Diese Mehrdeutigkeit von Personal- und Reflexivpronomen ist nur für die erste und zweite Person der

Pronomen vorhanden. In dem Satz „ich wasche dich“ ist „dich“ Personalpronomen, wohingegen in „du wäscht dich“ es als Reflexivpronomen angesehen werden muß. Technisch handelt es sich um ein Personalpronomen, wenn das Merkmal `ref` für das Pronomen in Subjektposition und das Merkmal `ref_comp` gemeinsame Werte haben.

Der Beispielsatz wird wie folgt repräsentiert:

```
LEXICON
   ich  : [cat:{pers_pron} sem:{animate}
           pers:{1} num:{sing} case:{nom} ref:{r1}]
   gebe : [cat:{v_fin} sort0:{animate}
           arg1:{IOBJ} sort1:{animate}
           arg2:{DOBJ} sort2:{inanimate} dobj_case:{acc}
           pers:{1} num:{sing} reflexiv:{0}]
   ihm  : [cat:{pers_pron} sem:{animate}
           pers:{3} num:{sing} gender:{masc,neutr} case:{dat}]
   das  : [cat:{det} num:{sing} gender:{neutr} case:{nom,acc}
           atype_sel:{weak}]
   buch : [cat:{n} sem:{inanimate} nodet:{f}
           pers:{3} gender:{neutr} case:{nom,dat,acc} num:{sing}]
END
```

A.3 CD-Grammatik

Dieser Abschnitt beschreibt die verwendete Beispielgrammatik.

A.3.1 Hierarchien

Die folgenden Hierarchien werden verwendet, um Wortarten zu gruppieren. Alle Wortarten unter dem Element `head` werden zu einer Menge zusammengefaßt, die in bestimmten Regeln als Stellvertreter für die einzelnen Elemente verwendet wird. Die Elemente können als potentielle Köpfe einer Phrase mit dem Namen der Hierarchie interpretiert werden.

```
np->top(head(n,pers_pron,refl_pron,name),
        rest(v_fin,v_inf,v_par,det,adj,poss_pron,prep));
det->top(head(det,poss_pron),
         rest(v_fin,v_inf,v_par,adj,n,pers_pron,refl_pron,name,
              prep));
vp->top(head(v_par,v_inf,v_fin),
        rest(n,pers_pron,refl_pron,name,det,adj,poss_pron,prep));
pp->top(head(prep),
        rest(n,pers_pron,refl_pron,name,det,adj,poss_pron,v_fin,
             v_inf,v_par));
adj->top(head(adj),
         rest(n,pers_pron,refl_pron,name,det,prep,poss_pron,
              v_fin,v_inf,v_par));
```

```
pmod->top(head(n,pers_pron,refl_pron,name,v_par,v_inf,v_fin),
          rest(det,adj,prep,poss_pron));
n->top(head(n),
       rest(prep,pers_pron,refl_pron,name,det,adj,poss_pron,
            v_fin,v_inf,v_par));
vfin->top(head(v_fin),
          rest(prep,pers_pron,refl_pron,name,det,adj,poss_pron,
               n,v_inf,v_par));
```

Die Hierarchie `ignore` definiert alle Wortarten, welche mit der Bezeichnung `IGNORE` der Wurzel untergeordnet werden können. Diese mögliche Unterordnung wird allerdings hart bestraft, um vollständige Analysen zu bevorzugen.

```
ignore->top(head(prep,pers_pron,refl_pron,name,det,adj,poss_pron,n),
            rest(v_fin,v_inf,v_par));
```

Die Hierarchie `vcomp` definiert alle Wortarten, welche als Komplemente eines finiten Verbs für die Forderung der Verbzweitstellung verwendet werden. Insbesondere sind dies Kanten mit der Beschriftung `IOBJ`, `DOBJ` und `PMOD`.

```
vcomp->top(head(n,pers_pron,refl_pron,name,prep),
           rest(v_fin,v_inf,v_par,adj,det,poss_pron));
```

Die Hierarchie `singles` definiert alle Kantenbezeichnungen, die nicht mehrfach verwendet werden dürfen, um einen gemeinsamen übergeordneten Knoten zu modifizieren. Zum Beispiel darf ein Substantiv nur von einem Artikel modifiziert werden.

```
singles->top(head(DET,PN,SUBJ,GMOD,IOBJ,DOBJ,VFIN,VINF,VPAR),
             rest(PMOD,ADJ,IGNORE));
```

Die Hierarchie `root` definiert alle Kantenbezeichnungen, die für eine Kante verwendet werden dürfen, welche den Wurzelknoten modifiziert.

```
root->top(head(VFIN,IGNORE),
          rest(VPAR,VINF,IOBJ,DOBJ,GMOD,SUBJ,PN,DET,PMOD,ADJ));
```

A.3.2 Analyseebenen

Es werden acht Analyseebenen für die Beschreibung der Syntax verwendet. Eine Ebenendefinition enthält die möglichen Kantenbeschriftungen der Ebene. Die Ebene `SYN` enthält allquantifizierte Regeln für die Beschreibung möglicher Unterordnungen. Die folgenden Ebenen sind technisch motivierte Hilfsebenen, um existenzquantifizierte Regeln modellieren zu können. Die Ebenen `SYN_SUBJ`, `SYN_IOBJ`, `SYN_DOBJ`, `SYN_VPAR` und `SYN_PMOD` modellieren die obligatorischen Ergänzungen des syntaktischen Valenzrahmens des Verbs. Die Ebene `SYN_DET` modelliert die Forderung, daß ein Substantiv nicht ohne Artikel verwendet werden darf. Die Ebene `SYN_PN` fordert die Existenz eines Komplements für eine Präposition.

```
SYN # VFIN, VPAR, VINF, SUBJ, IOBJ, DOBJ, PMOD, GMOD, DET, ADJ,
      PN, IGNORE;
SYN_SUBJ # IGNORE;
SYN_IOBJ # IGNORE;
SYN_DOBJ # IGNORE;
SYN_PMOD # IGNORE;
SYN_VPAR # IGNORE;
SYN_PN # IGNORE;
SYN_DET # IGNORE;
```

A.3.3 Regeln

Folgende Präfixe für die Namen von Bedingungen werden verwendet:

Präfix	Verwendungszweck
`CID_HARD`	Reduzierung des Suchraums.
`CID_PREF`	Präferenz von bestimmten Unterordnungen.
`CID_ERR`	Verarbeitung von nicht wohlgeformten Strukturen.

Harte Bedingungen dienen dazu, die möglichen Strukturbeschreibungen zu reduzieren, um eine effiziente Analyse zu ermöglichen. Präferenzen sind verletzbare Regeln, die Unterordnungen bestrafen, obwohl sie wohlgeformte Strukturen beschreiben. Konkurrieren verletzte Regeln mit anderen, werden diejenigen bevorzugt, die minimale Kosten verursachen. So kann z.B. modelliert werden, daß Präpositionalphrasen bevorzugt Verben modifizieren, indem die Unterordnung unter einer Substantivgruppe leicht bestraft wird. Liegt für diese Präferenz negative Evidenz vor, z.B. aufgrund von schwerwiegenden semantischen Verletzungen, wird automatisch die Unterordnung unter die Substantivgruppe bevorzugt, wenn dadurch die Gesamtkosten kleiner werden. Die Regeln, deren Verletzung nicht wohlgeformte Strukturen beschreiben, werden wesentlich härter bestraft als Präferenzen, damit eine konkurrierende Situation immer zu Gunsten der Präferenzen entschieden wird. Wird eine derartige Regel verletzt, kann sie als Erklärungsgrundlage für den Fehler herangezogen werden.

Unterordnung einer Wortform unter die Wurzel

Die syntaktische Funktion eines finiten Verbs erfordert die Unterordnung eines Verbs unter die Wurzel.

```
{X:SYN} : CID_HARD_VFINLABEL : syn : 0.0 :
    X.label=VFIN -> root(X^id) & subsumes(vp,head,X@cat);
```

Alle in der Hierarchie `ignore` definierten Wortarten dürfen auch unter der Wurzel stehen, um partielle Strukturbeschreibungen zu ermöglichen. Die Kantenbezeichnung ist dabei immer `IGNORE`.

```
{X:SYN} : CID_HARD_IGNORELABEL : syn : 0.0 :
    X.label=IGNORE -> root(X^id) & subsumes(ignore,head,X@cat);
```

Es dürfen nur die Beschriftungen der Hierarchie `root` für die Unterordnung unter den Wurzelknoten verwendet werden.

```
{X:SYN} : CID_HARD_ROOT : syn : 0.0 :
    root(X^id) -> subsumes(root, head, X.label);
```

Einschränkung der möglichen Wortarten in bestimmten syntaktischen Funktionen

Die folgende Tabelle definiert die zugelassenen Wortarten für syntaktische Funktionen, die nicht durch lexikalische Merkmale lizenziert werden. Die zugelassenen Wortformen werden durch Hierarchien definiert. Dabei werden auch syntaktisch fehlerhafte Unterordnungen erlaubt, die durch spezielle Regeln bestraft werden.

Beschriftung	Modifikator	Modifikant
DET	det	np
ADJ	adj	np
SUBJ	np	vp
GMOD	n	n
PN	pp	np
PMOD	pp	pmod

```
{X:SYN} : CID_HARD_DETLABEL : syn : 0.0 :
   X.label=DET -> ~root(X^id) &
                  subsumes(np,head,X^cat) &
                  subsumes(det,head,X@cat);
{X:SYN} : CID_HARD_ADJLABEL : syn : 0.0 :
   X.label=ADJ -> ~root(X^id) &
                  subsumes(np,head,X^cat) &
                  subsumes(adj,head,X@cat);
{X:SYN} : CID_HARD_SUBJLABEL : syn : 0.0 :
   X.label=SUBJ -> ~root(X^id) &
                  subsumes(vp,head,X^cat) &
                  subsumes(np,head,X@cat);
{X:SYN} : CID_HARD_GMODLABEL : syn : 0.0 :
   X.label=GMOD -> ~root(X^id) &
                  subsumes(n,head,X^cat) &
                  subsumes(n,head,X@cat);
{X:SYN} : CID_HARD_PNLABEL : syn : 0.0 :
   X.label=PN -> ~root(X^id) &
                  subsumes(pp,head,X^cat) &
                  subsumes(np,head,X@cat);
{X:SYN} : CID_HARD_PMODLABEL : syn : 0.0 :
   X.label=PMOD -> ~root(X^id) &
                  subsumes(pmod,head,X^cat) &
                  subsumes(pp,head,X@cat);
```

Präpositionen dürfen keine Hilfsverben modifizieren.

```
{X:SYN,Y:SYN} : CID_HARD_PMODNOAUXVERB : syn : 0.0 :
   X.label=PMOD & X^id=Y^id -> Y.label!=VPAR & Y.label!=VFIN;
```

Es dürfen nur verschiedene Komplemente für ein Verb verwendet werden.

```
{X:SYN,Y:SYN} : CID_HARD_SAMEVPCOMPL : syn : 0.0 :
   ~root(X^id) & X^id=Y^id & subsumes(vp,head,X^cat) &
   (X.label=SUBJ | X.label=PMOD | X.label=DOBJ | X.label=IOBJ) ->
      X.label != Y.label;
```

Bestimmte Beschriftungen werden durch lexikalische Merkmale lizenziert, insbesondere durch die Merkmale `arg1` und `arg2` eines Verbs. Der Index deutet dabei die Nummer des Komplements an. Die Werte dieser Merkmale sind Kantenbeschriftungen, d.h. syntaktische Funktionen, die als Komplemente in Frage kommen. Die folgende Tabelle definiert für einen Merkmalswert des ersten oder zweiten Arguments die zugelassenen Wortarten, welche durch Hierarchien repräsentiert werden. Lizenzgeber sind diejenigen Wortformen, die die Merkmale `arg1` bzw. `arg2` realisieren.

Beschriftung	Modifikator
DOBJ	np
IOBJ	np
VPAR	vp
VINF	vp

```
{X:SYN} : CID_HARD_DOBJLABEL : syn : 0.0 :
   X.label = DOBJ -> ~root(X^id) &
                  ( ( exists(X^arg1) &
                      X^arg1=DOBJ &
                      subsumes(np,head,X@cat) ) |
                    ( exists(X^arg2) &
                      X^arg2=DOBJ &
                      subsumes(np,head,X@cat) ) );
{X:SYN} : CID_HARD_IOBJLABEL : syn : 0.0 :
   X.label = IOBJ -> ~root(X^id) &
                  ( ( exists(X^arg1) &
                      X^arg1=IOBJ &
                      subsumes(np,head,X@cat) ) |
                    ( exists(X^arg2) &
                      X^arg2=IOBJ &
                      subsumes(np,head,X@cat) ) );
{X:SYN} : CID_HARD_VPARLABEL : syn : 0.0 :
   X.label = VPAR -> ~root(X^id) &
                  ( ( exists(X^arg1) &
                      X^arg1=VPAR &
                      subsumes(vp,head,X@cat) ) |
                    ( exists(X^arg2) &
                      X^arg2=VPAR &
                      subsumes(vp,head,X@cat) ) );
{X:SYN} : CID_HARD_VINFLABEL : syn : 0.0 :
   X.label = VINF -> ~root(X^id) &
```

```
                    ( ( exists(X^arg1) &
                        X^arg1=VINF &
                        subsumes(vp,head,X@cat) ) |
                      ( exists(X^arg2) &
                        X^arg2=VINF &
                        subsumes(vp,head,X@cat) ) );
```

Eindeutigkeit von syntaktischen Funktionen

Bestimmte syntaktische Funktionen müssen eindeutig sein, d.h. eine Wortform darf nur einmal in dieser Funktion modifiziert werden. Die Hierarchie `singles` definiert die eindeutigen Beschriftungen.

```
{X:SYN,Y:SYN} : CID_HARD_SINGLELABEL : syn : 0.0 :
    subsumes(singles,head,X.label) & X^id=Y^id ->
             X.label != Y.label;
```

Zyklenverbot

Die Verwendung von direkten Zyklen, d.h. die gegenseitige Modifikation zweier Wortformen, kann durch eine harte Bedingung realisiert werden. Zyklen über mehr als zwei Kanten müssen allerdings durch einen speziellen Test des Parsingverfahrens ausgeschlossen werden.

```
{X:SYN,Y:SYN} : CID_HARD_NOCYCLE : syn : 0.0 :
    X@id=Y^id -> ~ X^id=Y@id;
```

Semantik

Es wird nur eine sehr einfache semantische Modellierung verwendet: Jedes Komplement eines Verbs unterliegt u.U. selektionalen Restriktionen. Weder semantische Rollen (Agens, Patiens, etc.) noch selektionale Restriktionen anderer Wortarten werden unterstützt. Ebenso wird auf eine relativ aufwendige Modellierung der Semantik als eigenständige, relativ autonome Analyseebene verzichtet. Die Semantik verfolgt hier nur ein Ziel: Für Verben, deren Komplemente bestimmten selektiven Restriktionen genügen müssen, kann die Anzahl der Strukturbeschreibungen reduziert werden.

```
{X:SYN} : CID_ERR_SORT0 : syn : 0.005 :
   ~root(X^id) & X.label = SUBJ & exists(X^sort0) &
    exists(X@sem) -> X^sort0 = X@sem;
{X:SYN,Y:SYN} : CID_ERR_SORT0_VPAR : syn : 0.005 :
   ~root(X^id) & X^id=Y^id & X.label = SUBJ &
    exists(Y@sort0) &
    exists(X@sem) -> Y@sort0 = X@sem;
{X:SYN} : CID_ERR_SORT1 : syn : 0.005 :
   ~root(X^id) & X.label = X^arg1 & exists(X^sort1) &
    exists(X@sem) -> X^sort1 = X@sem;
{X:SYN} : CID_ERR_SORT2 : syn : 0.005 :
```

```
~root(X^id) & X.label = X^arg2 & exists(X^sort2) &
 exists(X@sem) -> X^sort2 = X@sem;
```

Existenzforderungen

Für die Modellierung von Existenzforderungen (obligatorische Ergänzungen) werden Hilfsebenen verwendet. Die Bäume der Hilfsebenen repräsentieren inverse Relationen bezüglich einer syntaktischen Funktion der Ebene SYN. Zum Beispiel modifiziert ein Substantiv einen Artikel in der Hilfsebene SYN_DET, während in der Ebene SYN ein Artikel ein Substantiv modifiziert. Die obligatorische Wortform wird dabei immer direkt unter der Wurzel angeordnet. Existiert sie nicht, wird die fordernde Wortform unter Strafe ebenfalls unter der Wurzel angeordnet. Die folgende Tabelle stellt die Unterordnungen für nicht auf lexikalische Merkmale basierenden Obligatheitsforderungen zusammen. Eine Bedingung realisiert dabei den Informationsaustausch mit der SYN-Ebene, so daß auf der Hilfsebene die entsprechende syntaktische Funktion der SYN-Ebene dargestellt wird (sogenannte *Mapping* Bedingung).

Ebene	Modifikator	Modifikant
SYN_DET	n	det
SYN_SUBJ	vfin	np
SYN_PN	pp	np

```
{X:SYN_DET} : CID_HARD_NEED_DET : syn : 0.0 :
   ~root(X^id) -> subsumes(n,head,X@cat) &
                  subsumes(det,head,X^cat);
{X:SYN_DET} : CID_ERR_NEED_DET : syn : 0.002 :
    root(X^id) <-> (exists(X@nodet) & X@nodet=t) |
                    ~(subsumes(n,head,X@cat));
{X:SYN, Y:SYN_DET} : CID_HARD_NEED_SYNCDET : syn : 0.0 :
    X@id=Y^id -> X.label=DET & X^id=Y@id;

{X:SYN_SUBJ} : CID_HARD_NEED_SUBJ : syn : 0.0 :
   ~root(X^id) -> subsumes(vfin,head,X@cat) &
                  subsumes(np,head,X^cat);
{X:SYN_SUBJ} : CID_ERR_NEED_SUBJ : syn : 0.002 :
    root(X^id) <-> ~(subsumes(vfin,head,X@cat));
{X:SYN, Y:SYN_SUBJ} : CID_HARD_NEED_SYNCSUBJ : syn : 0.0 :
    X@id=Y^id -> X.label=SUBJ & X^id=Y@id;

{X:SYN_PN} : CID_HARD_NEED_PN : syn : 0.0 :
   ~root(X^id) -> subsumes(pp,head,X@cat) &
                  subsumes(np,head,X^cat);
{X:SYN_PN} : CID_ERR_NEED_PN : syn : 0.0015 :
    root(X^id) <-> ~(subsumes(pp,head,X@cat));
{X:SYN, Y:SYN_PN} : CID_HARD_NEED_SYNCPN : syn : 0.0 :
    X@id=Y^id -> X.label=PN & X^id=Y@id;
```

Die folgende Tabelle stellt die Unterordnungen für auf lexikalische Merkmale basierenden Obligatheitsforderungen zusammen. Die modifizierende Wortform ist diejenige, welche die Merkmale `arg1` bzw. `arg2` realisiert.

Ebene	Modifikant
SYN_IOBJ	np
SYN_DOBJ	np
SYN_PMOD	pp
SYN_VPAR	vp

```
{X:SYN_IOBJ} : CID_HARD_NEED_IOBJ : syn : 0.0 :
   ~root(X^id) -> ( (exists(X@arg1) & X@arg1=IOBJ) |
                    (exists(X@arg2) & X@arg2=IOBJ) )  &
                  subsumes(np,head,X^cat);
{X:SYN_IOBJ} : CID_ERR_NEED_IOBJ : syn : 0.002 :
    root(X^id) <-> (~exists(X@arg1) & ~exists(X@arg2)) |
                   (exists(X@arg1) & ~exists(X@arg2) & X@arg1!=IOBJ) |
                   (exists(X@arg2) & ~exists(X@arg1) & X@arg2!=IOBJ) |
                   (exists(X@arg1) & exists(X@arg2) & X@arg1!=IOBJ &
                    X@arg2!=IOBJ);
{X:SYN, Y:SYN_IOBJ} : CID_HARD_NEED_SYNCIOBJ : syn : 0.0 :
    X@id=Y^id -> X^id=Y@id & X.label=IOBJ &
           ((exists(Y@arg1) & Y@arg1=IOBJ) |
            (exists(Y@arg2) & Y@arg2=IOBJ));

{X:SYN_DOBJ} : CID_HARD_NEED_DOBJ : syn : 0.0 :
   ~root(X^id) -> ( (exists(X@arg1) & X@arg1=DOBJ) |
                    (exists(X@arg2) & X@arg2=DOBJ) )  &
                  subsumes(np,head,X^cat);
{X:SYN_DOBJ} : CID_ERR_NEED_DOBJ : syn : 0.002 :
    root(X^id) <-> (~exists(X@arg1) & ~exists(X@arg2)) |
                   (exists(X@arg1) & ~exists(X@arg2) & X@arg1!=DOBJ) |
                   (exists(X@arg2) & ~exists(X@arg1) & X@arg2!=DOBJ) |
                   (exists(X@arg1) & exists(X@arg2) & X@arg1!=DOBJ &
                    X@arg2!=DOBJ);
{X:SYN, Y:SYN_DOBJ} : CID_HARD_NEED_SYNCDOBJ : syn : 0.0 :
    X@id=Y^id -> X^id=Y@id & X.label=DOBJ &
           ((exists(Y@arg1) & Y@arg1=DOBJ) |
            (exists(Y@arg2) & Y@arg2=DOBJ));

{X:SYN_PMOD} : CID_HARD_NEED_PMOD : syn : 0.0 :
   ~root(X^id) -> ( (exists(X@arg1) & X@arg1=PMOD) |
                    (exists(X@arg2) & X@arg2=PMOD) )  &
                  subsumes(pp,head,X^cat);
{X:SYN_PMOD} : CID_ERR_NEED_PMOD : syn : 0.0015 :
    root(X^id) <-> (~exists(X@arg1) & ~exists(X@arg2)) |
                   (exists(X@arg1) & ~exists(X@arg2) & X@arg1!=PMOD) |
                   (exists(X@arg2) & ~exists(X@arg1) & X@arg2!=PMOD) |
```

```
                   (exists(X@arg1) & exists(X@arg2) & X@arg1!=PMOD &
                    X@arg2!=PMOD);
{X:SYN, Y:SYN_PMOD} : CID_HARD_NEED_SYNCPMOD : syn : 0.0 :
    X@id=Y^id -> X^id=Y@id & X.label=PMOD &
           ((exists(Y@arg1) & Y@arg1=PMOD) |
            (exists(Y@arg2) & Y@arg2=PMOD));

{X:SYN_VPAR} : CID_HARD_NEED_VPAR : syn : 0.0 :
   ~root(X^id) -> ( (exists(X@arg1) & X@arg1=VPAR) |
                    (exists(X@arg2) & X@arg2=VPAR) )  &
                  subsumes(vp,head,X^cat);
{X:SYN_VPAR} : CID_ERR_NEED_VPAR : syn : 0.0015 :
    root(X^id) <-> (~exists(X@arg1) & ~exists(X@arg2)) |
                   (exists(X@arg1) & ~exists(X@arg2) & X@arg1!=VPAR) |
                   (exists(X@arg2) & ~exists(X@arg1) & X@arg2!=VPAR) |
                   (exists(X@arg1) & exists(X@arg2) & X@arg1!=VPAR &
                    X@arg2!=VPAR);
{X:SYN, Y:SYN_VPAR} : CID_HARD_NEED_SYNCVPAR : syn : 0.0 :
    X@id=Y^id -> X^id=Y@id & X.label=VPAR &
           ((exists(Y@arg1) & Y@arg1=VPAR) |
            (exists(Y@arg2) & Y@arg2=VPAR));
```

Fehlerhafte Wortart in einer Unterordnung

Die harten Bedingungen erlauben syntaktisch fehlerhafte Unterordnungen. Die folgende Tabelle definiert die zu bestrafenden Unterordnungen. Der Modifikant bzw. Modifikator bezeichnet dabei die Wortart.

Beschriftung	Modifikant	Modifikator	Strafe
VPAR		v_par	0.0011
VFIN		v_fin	0.0012
VINF		v_inf	0.0013
DET	n		0.0014
SUBJ	v_fin		0.0015

```
{X:SYN} : CID_ERR_MODIFIER_VPAR : syn : 0.0011 :
    X.label = VPAR -> X@cat=v_par;
{X:SYN} : CID_ERR_MODIFIER_VFIN : syn : 0.0012 :
    X.label = VFIN -> X@cat=v_fin;
{X:SYN} : CID_ERR_MODIFIER_VINF : syn : 0.0013 :
    X.label = VINF -> X@cat=v_inf;
{X:SYN} : CID_ERR_MODIFIEE_DET : syn : 0.0014 :
    X.label = DET & ~root(X^id) -> X^cat=n;
{X:SYN} : CID_ERR_MODIFIEE_SUBJ : syn : 0.0015 :
    X.label = SUBJ & ~root(X^id) -> X^cat=v_fin;
```

Die Verwendung der Beschriftung IGNORE wird stark bestraft, damit eine vollständige Analyse bevorzugt wird.

```
{X:SYN} : CID_ERR_NO_IGNORE : syn : 0.00001 :
    X.label != IGNORE;
```

Fehlerhafte Anordnungen

Folgende Tabelle definiert die korrekten Anordnungen für bestimmte Beschriftungen. Das Anordnungssymbol ← (→) bedeutet, daß der modifizierende Knoten links (rechts) vom Modifikanten steht. Die Regel für die Genitivanordnung ist hart, um möglichst viele potentielle Genitivergänzungen auszuschließen. Da die CD-Grammatik keine Kasusinformation enthält, könnte jede Substantivgruppe eine Genitivergänzung darstellen.

Beschriftung	Anordnung	Strafe
PN	→	0.0051
VPAR	→	0.0052
GMOD	→	0.0
DET	←	0.0054
ADJ	←	0.0055

```
{X:SYN} : CID_ERR_PRECPN_MODIFIEE : syn : 0.0051 :
    X.label=PN & ~root(X^id) -> distance(X^id,X@id) > 0;
{X:SYN} : CID_ERR_PRECVPAR_MODIFIEE : syn : 0.0052 :
    X.label=VPAR & ~root(X^id) -> distance(X^id,X@id) > 0;
{X:SYN} : CID_ERR_PRECGMOD_MODIFIEE : syn : 0.0 :
    X.label=GMOD & ~root(X^id) -> distance(X^id,X@id) > 0;
{X:SYN} : CID_ERR_PRECDET_MODIFIER : syn : 0.0054 :
   X.label=DET & ~root(X^id) -> distance(X@id,X^id) > 0;
{X:SYN} : CID_ERR_PRECADJ_MODIFIER : syn : 0.0055 :
   X.label=ADJ & ~root(X^id) -> distance(X@id,X^id) > 0;
```

Folgende binäre Bedingung fordert, daß ein Artikel vor einem attributiven Adjektiv steht, falls beide dieselbe Wortform modifizieren.

```
{X:SYN,Y:SYN} : CID_ERR_PREC_DET_ADJ : syn : 0.0056 :
   X^id=Y^id & X.label = DET & Y.label = ADJ ->
      distance(X@id, Y@id) > 0;
```

Eine Präposition steht immer ganz links in einer Präpositionalphrase.

```
{X:SYN,Y:SYN} : CID_ERR_PREPFORANY : syn : 0.0057 :
      X@id=Y^id & X.label = PN & distance(X^id,X@id)>0
      -> distance(X^id, Y@id) > 0;
```

Modifiziert eine Präpositionalphrase ein Substantiv, so darf sie niemals links vom Substantiv stehen. Die Regel ist hart, um die strukturellen Möglichkeiten von Präpositionalphrasen einzuschränken. Es wird also eine Anbindung an das Verb oder an den Wurzelknoten erwogen.

```
{X:SYN} : CID_HARD_PREC_PMOD_np : syn : 0.0 :
       ~root(X^id) & X.label = PMOD & subsumes(np,head,X^cat)
      -> distance(X@id, X^id) < 0;
```

Folgende Bedingungen stellen Forderungen an die Position des finiten Verbs bzw. an das Partizip. Das finite Verb steht nicht an erster Stelle, das Subjekt steht nicht hinter dem zweiten Partizip, Das zweite Partizip steht nach den Komplementen und das finite Verb steht in der zweiten Position.

```
{X:SYN} : CID_ERR_NOTFIRST_VFIN : syn : 0.0058 :
      X.label = VFIN -> ~start(X@id);
{X:SYN,Y:SYN} : CID_ERR_LAST_VPAR : syn : 0.0059 :
     ~root(Y^id) & X.label=VPAR & X^id=Y^id ->
        distance(Y@id,X@id)>0;
{X:SYN,Y:SYN} : CID_ERR_BEHIND_COMPL_VPAR : syn : 0.0060 :
     ~root(Y^id) & X.label=VPAR & X@id=Y^id &
     (X@arg1=Y.label | X@arg2=Y.label) ->
          distance(Y@id,X@id)>0;
{X:SYN,Y:SYN} : CID_ERR_VERBSECOND : syn : 0.0061 :
    ~root(X^id) & X^id=Y^id & X^cat=v_fin &
    subsumes(vcomp,head,X@cat) & subsumes(vcomp,head,Y@cat) &
    distance(X@id,X^id)>0 ->
    distance(Y@id,X^id)<0;
```

Eine Genitivergänzung darf keine Wortformen überspannen.

```
{X:SYN,Y:SYN} : CID_HARD_GMODPROJ : syn : 0.0 :
   X.label=GMOD & ~root(X^id) & ~root(Y^id) & X^id=Y@id ->
      distance(Y^id,X^id)>0 |
      distance(Y^id,X@id)<0;
```

Präferenzen

Die Anbindung einer Genitivergänzung wird gegenüber anderen Analysen dispräferiert, da die CD-Grammatik keine Kasusinformation enthält und dadurch die notwendige Kasusforderung nicht realisiert werden kann (hierfür ist die Fehlersimulation zuständig).

```
{X:SYN} : CID_PREF_NO_GMOD : syn : 0.9 :
    X.label != GMOD;
```

Die folgenden Bedingungen bestrafen Anordnungen, deren Modifikator eine Entfernung größer als eins (bzw. zwei) zum Modifikanten aufweist. Die Strafen sind dabei abhängig von den betrachteten L-Knoten, insbesondere von der Position der Knoten.

```
{X:SYN} : CID_PREF_DETDISTANCE : syn :
             [0.91+[0.1/[abs(distance(X^id,X@id))]]] :
      ~root(X^id) & X.label = DET ->
          abs(distance(X@id,X^id)) = 1;
{X:SYN} : CID_PREF_ADJDISTANCE : syn :
             [0.92+[0.1/[abs(distance(X^id,X@id))]]] :
       ~root(X^id) & X.label = ADJ ->
```

```
          abs(distance(X@id,X^id)) = 1;
{X:SYN} : CID_PREF_PNDISTANCE : syn :
             [0.93+[0.1/[abs(distance(X^id,X@id))]]] :
       ~root(X^id) & X.label = PN ->
            abs(distance(X@id,X^id)) = 2;
{X:SYN} : CID_PREF_GMODDISTANCE : syn :
            [0.89+[0.1/[abs(distance(X^id,X@id))]]] :
       ~root(X^id) & X.label = GMOD ->
            abs(distance(X@id,X^id)) = 2;
{X:SYN} : CID_PREF_PMODDISTANCE : syn :
            [0.87+[0.1/[abs(distance(X^id,X@id))]]] :
       ~root(X^id) & X.label = PMOD & subsumes(np, head,X^cat) ->
            abs(distance(X@id,X^id)) = 1;
```

Projektive Strukturen sollten gegenüber nicht projektiven Strukturen bevorzugt werden. Da die CD-Grammatik keine morphologischen Merkmale kennt, würde eine allgemeine Projektionsregel u.U. *fehlerhafte* Strukturen bevorzugen, weil z.B. Kongruenzfehler erst von der Fehlersimulation entdeckt werden. Die Projektionsforderung wird daher nur für die Anbindung von Präpositionalphrasen formuliert.

```
{X:SYN,Y:SYN} : CID_PREF_PROJPMOD : syn : 0.88 :
      (X.label=PMOD|Y.label=PMOD) & X@id=Y^id ->
      ~ ((distance(Y@id, X^id) > 0 &
          distance(X^id, Y^id) > 0) |
         (distance(Y^id, X^id) > 0 &
          distance(X^id, Y@id) > 0));
```

Eine Präposition modifiziert meist kein Personal- oder Reflexivpronomen.

```
{X:SYN} : CID_PREF_PMODNOPRONOUN : syn : 0.9 :
   X.label=PMOD & ~root(X^id) -> X^cat!=pers_pron & X^cat!=refl_pron;
```

Hilfskonstruktionen

Die folgenden Bedingungen sind rein technischer Natur und haben auf die möglichen Strukturbeschreibungen keinen Einfluß, da sie mit den Wert eins bestraft werden. Das Merkmal `reflexiv` wird von der CD-Grammatik nicht verwendet. Der Wert wird jedoch für die Abbildungsregeln im integrativen Ansatz benötigt.

```
{X:SYN} : CID_PREF_REFLEXIV : syn : 1.0 :
   exists(X@reflexiv)-> true;
```

In der verwendeten Softwareversion für das CDG-Parsing werden nur Merkmale als relevant eingestuft, welche in den verwendeten Regeln auf den unteren Knoten verweisen. Die Merkmale `arg1` und `arg2` werden aber nur im oberen Knoten referenziert.

```
{X:SYN} : CID_PREF_BUGGY : syn : 1.0 :
   exists(X@arg1) | exists(X@arg2) -> true;
```

A.4 Strukturabbildungen

Im folgenden werden die verwendeten Strukturabbildungen beschrieben, welche die Strukturbeschreibungen einer CD-Grammatik auf Fehlersimulationsmodelle abbilden. Außerdem wird der entwickelte Formalismus für die Notation der Abbildungsregeln erläutert. Der Formalismus verwendet einige Konstrukte aus [Stockfleth 1999].

A.4.1 Formalismus

Der Anweisungsblock mit den Abbildungsregeln wird durch das Schlüsselwort `CDG-MAPPING` eingeführt. Das Block wird mit der Anweisung `END` abgeschlossen. Die verwendete CD-Grammatik wird innerhalb des Abbildungsblocks mit der Anweisung `USE-GRAMMAR` angegeben. Alle Abbildungsregeln können sich auf eine voreingestellte Analyseebene beziehen, die mit der Anweisung `DEFAULT-LEVEL` definiert wird. Der folgende Anweisungsblock, eingeleitet mit dem Schlüsselwort `MAP-RULES`, enthält die verwendeten Abbildungsregeln.

```
INCLUDE "lexicon.dgl"
CDG-MAPPING
   USE-GRAMMAR "rules.cdg"
   DEFAULT-LEVEL "SYN"
   MAP-RULES
      // Hier stehen die Abbildungsregeln ...
   END
END
```

Die `INCLUDE` Anweisung liest dabei die Datei mit dem verwendeten Lexikon ein. Im folgenden werden die einzelnen Abbildungsregeln vorgestellt. Die Syntax für die Notation dieser Regeln wurde in Abschnitt 5.3 auf Seite 82 vorgestellt.

A.4.2 Abbildungsregeln

Die Entscheidung des CDG-Parsings für eine Wortart soll als nicht revidierbar angesehen werden.

```
value(WORD,cat,SET) => !restrict(WORD,cat,SET)
```

Ein Artikel bzw. ein Adjektiv und ein Substantiv müssen im Kasus, Numerus und Genus kongruieren. Das Genus des Substantivs darf in der Faktenfehlersimulation nicht angezweifelt werden.

```
modifies(DET,"DET",NOUN) => agree(DET,NOUN,case)
                            agree(DET,NOUN,num)
                            agree(DET,!NOUN,gender)
modifies(ADJ,"ADJ",NOUN) => agree(ADJ,NOUN,case)
                            agree(ADJ,NOUN,num)
                            agree(ADJ,!NOUN,gender)
```

Das Subjekt steht immer im Nominativ. Außerdem müssen Subjekt und finites Verb in Numerus und Person kongruieren. Die Person des Substantivs wird als gegeben angenommen.

```
modifies(NOUN,"SUBJ",VERB) => restrict(NOUN,case,{nom})
                              agree(VERB,NOUN,num)
                              agree(VERB,!NOUN,pers)
```

Das direkte Objekt muß mit dem Kasusrahmen des Verbs verträglich sein. Das Merkmal `dobj_case` des Verbs kann nicht verändert werden. Das indirekte Objekt steht immer im Dativ.

```
modifies(NOUN,"DOBJ",VERB) => agree(NOUN,!VERB,case,dobj_case)
modifies(NOUN,"IOBJ",VERB) => restrict(NOUN,case,{dat})
```

Eine Präposition kann entweder das finite Verb oder ein Substantiv modifizieren. Im ersten Fall muß die selektionale Forderung des Verbs, d.h. ob es eine lokale oder direktionale Ortsangabe fordert, mit dem Selektionstyp der Präposition übereinstimmen. Die Merkmalswerte können dabei nicht verändert werden. Modifiziert eine Präposition ein Substantiv, muß eine lokale Ortsangabe verwendet werden.

```
modifies(PREP,"PMOD",VERB)
value(VERB,cat,{v_fin,v_par,v_inf})
=> agree(!PREP,!VERB,sel)

modifies(PREP,"PMOD",NOUN)
value(NOUN,cat,{n,pers_pron,refl_pron,name})
=> restrict(!PREP,sel,{loc})
```

Eine Präposition regiert den Kasus der folgenden Substantivgruppe. Die Kasusforderung kann nicht angezweifelt werden.

```
modifies(NOUN,"PN",PREP) => agree(!PREP,NOUN,pcase,case)
```

Modifizieren ein Artikel und ein Adjektiv ein gemeinsames Substantiv, muß die Klassifizierung des Adjektivs mit der Forderung des Artikels übereinstimmen.

```
modifies(DET,"DET",NOUN) modifies(ADJ,"ADJ",NOUN)
=> agree(!DET,ADJ,atype_sel,atype)
```

Der Kasus einer Genitivergänzung muß Genitiv sein.

```
modifies(GNOUN,"GMOD",NOUN) => restrict(GNOUN,case,{gen})
```

Die Konjugationsforderung des infiniten Verbs muß mit dem Hilfsverb verträglich sein.

```
modifies(VERB,"VPAR",AUX_VERB) => agree(!VERB,AUX_VERB,konj)
```

Wird ein Reflexivpronomen als indirektes oder direktes Objekt verwendet, muß es mit dem verwendeten Subjekt in Numerus und Person kongruieren. Wird ein Hilfsverb verwendet, müssen drei Kanten überprüft werden.

```
value(REFL,cat,{refl_pron}) modifies(REFL,L,VERB)
modifies(NOUN,"SUBJ",VERB)
=> agree(REFL,!NOUN,num)
   agree(REFL,!NOUN,pers)

value(REFL,cat,{refl_pron}) modifies(REFL,L1,X)
no-modifier("SUBJ",X)
modifies(X,L2,VERB) modifies(NOUN,"SUBJ",VERB)
=> agree(REFL,!NOUN,num)
   agree(REFL,!NOUN,pers)
```

Wird ein Personalpronomen als direktes und indirektes Objekt verwendet, so darf es *nicht* mit dem Subjekt in Numerus und Person kongruieren, falls das Subjekt ein Personalpronomen ist. Da Negationen in dem Formalismus nicht ausgedrückt werden können, werden zwei Merkmale verwendet, deren Werte genau dann verträglich sind, wenn die Kongruenzforderung nicht erfüllt ist.

```
value(PERS,cat,{pers_pron}) value(NOUN,cat,{pers_pron})
modifies(PERS,"DOBJ",VERB) modifies(NOUN,"SUBJ",VERB)
=> agree(NOUN,PERS,ref,ref_comp)

value(PERS,cat,{pers_pron}) value(NOUN,cat,{pers_pron})
modifies(PERS,"DOBJ",VERB) modifies(VERB,ANY,AUX_VERB)
modifies(NOUN,"SUBJ",AUX_VERB)
=> agree(NOUN,PERS,ref,ref_comp)

value(PERS,cat,{pers_pron}) value(NOUN,cat,{pers_pron})
modifies(PERS,"IOBJ",VERB) modifies(NOUN,"SUBJ",VERB)
=> agree(NOUN,PERS,ref,ref_comp)

value(PERS,cat,{pers_pron}) value(NOUN,cat,{pers_pron})
modifies(PERS,"IOBJ",VERB) modifies(VERB,ANY,AUX_VERB)
modifies(NOUN,"SUBJ",AUX_VERB)
=> agree(NOUN,PERS,ref,ref_comp)
```

In dieser Beispielgrammatik wird der Antezedent eines Possessivpronomens immer dann mit dem verwendeten Subjekt identifiziert, wenn das Possessivpronomen nicht das Subjekt modifiziert.

```
value(POSS,cat,{poss_pron})
modifies(POSS,"DET",ANY1)
modifies(ANY1,"DOBJ",ANY2)
modifies(NOUN,"SUBJ",ANY3)
=> agree(POSS,NOUN,spers,pers)
   agree(POSS,NOUN,snum,num)
   agree(POSS,NOUN,sgender,gender)

value(POSS,cat,{poss_pron})
```

```
modifies(POSS,"DET",ANY1)
modifies(ANY1,"IOBJ",ANY2)
modifies(NOUN,"SUBJ",ANY3)
=> agree(POSS,NOUN,spers,pers)
   agree(POSS,NOUN,snum,num)
   agree(POSS,NOUN,sgender,gender)

value(POSS,cat,{poss_pron})
modifies(POSS,"DET",ANY1)
modifies(ANY1,"PN",ANY2)
modifies(NOUN,"SUBJ",ANY3)
=> agree(POSS,NOUN,spers,pers)
   agree(POSS,NOUN,snum,num)
   agree(POSS,NOUN,sgender,gender)

value(POSS,cat,{poss_pron})
modifies(POSS,"DET",ANY1)
modifies(ANY1,"GMOD",ANY2)
modifies(NOUN,"SUBJ",ANY3)
=> agree(POSS,NOUN,spers,pers)
   agree(POSS,NOUN,snum,num)
   agree(POSS,NOUN,sgender,gender)
```

Wenn ein Substantiv ohne Artikel verwendet werden kann und ein Adjektiv das Substantiv modifiziert, muß die starke Flexionsklasse für das Adjektiv verwendet werden.

```
modifies(ADJ,"ADJ",NOUN)
no-modifier("DET",NOUN)
value(NOUN,nodet,{t})
=> restrict(ADJ,atype,{strong})
```

Ein reflexives Verb benötigt als Komplement ein Reflexivpronomen.

```
value(VERB, reflexiv, {DOBJ}) modifies(COMP, "DOBJ", VERB)
=> restrict(COMP,cat, {refl_pron})
```

Wenn das Merkmal `nodet` eines Substantivs den Wert `t` hat, kann der Numerus nur Plural sein. Diese Forderung wird verwendet, um lexikalische Mehrdeutigkeiten aufzulösen. Zum Beispiel kann das Substantiv „Opas“ sowohl Genitiv Singular als auch Nominativ Plural sein.

```
value(X,nodet,{t}) => !restrict(X,num,{plur})
```

A.5 Testsätze

A.5.1 Korrekte Sätze

Valenzrahmen der Verben

Verb „lachen“ und „laufen“
(K1) das kind lacht
(K2) die menschen lachen
(K3) du lachst
(K4) Niels läuft
(K5) ich habe gelacht
(K6) wir sind gelaufen

Verb „suchen“ und „sehen“
(K7) der mann sucht ein bild
(K8) den ball sucht das kind
(K9) sie suchen ihn
(K10) er sieht sich
(K11) wir sehen uns
(K12) die tür sucht die frau
(K13) der opa hat ein buch gesehen
(K14) ein segel habt ihr gesucht

Verb „geben“
(K15) die oma gibt dem kind ein buch
(K16) ihr gebt die ferngläser den männern
(K17) ihm geben wir den apfel
(K18) das bild gibst du der frau
(K19) ich habe der frau das bild gegeben
(K20) sie hat die bücher den kindern gegeben
(K21) ihm haben wir ein fernglas gegeben
(K22) ein fernglas habt ihr ihm gegeben
(K23) ich habe mir es gegeben

Verb „liegen“
(K24) das fernglas liegt auf dem tisch
(K25) der wald liegt jenseits des berges
(K26) neben dem stuhl liegt der ball
(K27) der apfel hat in dem schrank gelegen
(K28) über den bergen haben die wälder gelegen

Verb „legen“
(K29) ich lege die bücher auf den tisch
(K30) den ball legst du neben den stuhl
(K31) auf die bücher legen die kinder die äpfel
(K32) wir haben das fernglas neben das fenster gelegt
(K33) sie haben wir auf den tisch gelegt

(K34) in die wälder habe ich das fernglas gelegt
(K35) du hast dich auf das deck gelegt
(K36) du hast ihn auf das deck gelegt

Verb „haben“
(K37) ich habe ein haus
(K38) den ball habt ihr
(K39) die kinder haben die äpfel gehabt
(K40) die segel hast du gehabt

Verb „sein“
(K41) ich bin eine frau
(K42) ein buch ist es
(K43) du bist ein mann gewesen
(K44) das segel ist es gewesen
(K45) ihr seid es gewesen

Verb „freuen“
(K46) ich freue mich
(K47) ihr habt euch gefreut
(K48) du hast dich über die vase gefreut
(K49) über die vase haben wir uns gefreut

Substantivgruppen

Adjektiv (bestimmter und unbestimmter Artikel, Nullartikel)
(K50) das kleine kind sucht den ball
(K51) der kleine mann lacht
(K52) die gelbe vase liegt auf dem tisch
(K53) er sieht ein gelbes fernglas
(K54) ein kleines kind sucht den ball
(K55) ein kleiner mann lacht
(K56) eine gelbe vase liegt auf dem tisch
(K57) er sieht das gelbe fernglas
(K58) kleine kinder suchen den ball
(K59) kleine männer lachen
(K60) gelbe vasen liegen auf dem tisch
(K61) er sieht kleine ferngläser
(K62) der schnelle mann sucht das fernglas
(K63) der schnelle mann sucht die vase
(K64) der schnelle mann sucht die gelbe vase

Possessivpronomen
(K65) der mann sucht seine kinder
(K66) die frau sieht ihr haus
(K67) ihrem mann hat sie ein buch gegeben
(K68) ich suche meinen ball

(K69) du legst dein buch auf den stuhl
(K70) sie sieht ihre oma
(K71) wir geben dem kind unseren gelben ball
(K72) ihr gebt eurem kind ein buch
(K73) sie suchen ihre ferngläser

Genitivergänzung
(K74) das buch der frau liegt neben dem stuhl
(K75) er gibt dem kind des mannes ein fernglas
(K76) sie sucht die frau des kindes des opas
(K77) die oma des kindes sieht die frau des mannes

Adverbiale Ergänzungen
(K78) das haus in dem wald liegt auf dem berg
(K79) er sucht die vase auf dem tisch
(K80) der mann mit dem kleinen kind lacht
(K81) der mann mit gelben augen läuft
(K82) der mann sieht das kind auf dem berg mit dem fernglas
(K83) er sucht mit dem fernglas das kind
(K84) den ball in dem wald hat der mann neben dem haus gesucht

A.5.2 Fehlerhafte Sätze

Fehlende Konstituenten

Fehlende Komplemente des Verbs
(F1) lacht
(F2) sucht den ball
(F3) freue mich
(F4) er sucht
(F5) der mann gibt dem kind
(F6) wir legen auf den tisch
(F7) die frau gibt das buch
(F8) opa gibt einen ball
(F9) das buch liegt
(F10) er legt das fernglas
(F11) ich freue

Fehlende Bestandteile in einer Substantiv- und Präpositionalgruppe
(F12) er sucht ball
(F13) kind sieht den mann
(F14) wir geben frau ein buch
(F15) mann gibt der frau buch
(F16) mann gibt frau buch
(F17) er legt buch auf tisch
(F18) kleine mann sucht das fernglas
(F19) er legt kleines buch auf den tisch

(F20) er legt den ball den tisch
(F21) das buch liegt dem stuhl

Sonstige
(F22) das kind
(F23) er den ball
(F24) wir haben das buch suchen
(F25) er ist laufen

Überflüssige Konstituenten

Artikel
(F26) der mann sucht den ball das

Präposition
(F27) er sieht es in

Substantivgruppe
(F28) er läuft der ball

Pronomen
(F29) er ich suche das buch
(F30) wir sehen ihm das fernglas

Anordnung

Artikel
(F31) mann der legt das buch in den schrank
(F32) sie gibt ihm buch das
(F33) die frau mannes des lacht

Adjektive
(F34) das buch gelbe liegt auf dem tisch
(F35) gelbe die vase liegt auf dem tisch

Präpositionen
(F36) es liegt dem tisch auf
(F37) es liegt dem auf tisch

Verben
(F38) er den ball sucht
(F39) sucht er den ball
(F40) er gesucht hat den ball
(F41) hat er gesucht den ball
(F42) wir haben gesehen den berg

Morpho-syntaktische Fehler

Kongruenz in einer Substantivgruppe
(F43) die ball sucht das kind
(F44) wir sehen dem berg
(F45) das bücher gebt ihr dem mann

Flexionsparadigma und Kongruenz
(F46) ein kleine mann lacht
(F47) der kleiner opa legt es neben das fenster
(F48) die kleine fernglas liegt auf dem tisch
(F49) ich gebe dem kleine mann ein buch
(F50) er gibt ein kleine kind das fernglas

Ortsangaben
(F51) ihr legt die vase auf dem tisch
(F52) die vase liegt auf den tisch
(F53) er sieht den mann mit den fernglas

Genitivergänzung
(F54) das buch der mann liegt auf dem tisch
(F55) das buch des mann liegt auf dem tisch

Possessivpronomen
(F56) er hat ihre kinder gesucht
(F57) sie legt den ball auf meinen stuhl
(F58) wir legen deine vase auf den tisch
(F59) er sucht seine ball
(F60) euren kinder geben uns den ball

Konjugation
(F61) die kinder haben gelaufen
(F62) ihr seid den kindern ein buch gegeben
(F63) wir lege die vase neben den schrank
(F64) ihm gibt ich ein buch

Reflexive Verben und Reflexivpronomen
(F65) ich freue den mann
(F66) sie freuen ihn
(F67) du freust mich
(F68) ich habe dich gefreut
(F69) wir haben euch gefreut
(F70) ihr hat euch über die vase gefreut
(F71) mich habe ich gefreut

Semantische Fehler
(F72) der ball sucht die tür

(F73) er gibt dem ball das buch
(F74) das buch legt die vase auf den tisch
(F75) der berg freut sich

Kombinatorische Fehler
(F76) ich gibt die buch dem kind
(F77) er ist die mann gesucht
(F78) die kind hat die frau die buch gegeben
(F79) ein kleine mann sieht der berg mit den fernglas
(F80) die ball sieht ein kleine mann mit fernglas
(F81) ich hat suchen die ball auf tisch

Anhang B

Java-Schnittstelle des integrativen Verfahrens

Im folgenden werden ausgewählte Methoden der drei wichtigsten Java-Klassen für das integrative Verfahren vorgestellt. Eine Verwendung dieser Klassen kann im Quelltext des Entwicklungswerkzeugs *Grammar Diagnosis* genauer studiert werden. Einige nicht näher aufgeführten Klassen werden für das Verständnis der Schnittstelle kurz erläutert:

- Eine Merkmalsstruktur nach Definition 3.2.1 wird durch ein Objekt der Klasse `SlotValuePairList` repräsentiert.

- Ein Prädikat eines Fehlersimulationsmodells nach Definition 3.2.7 wird als Komponente durch ein Objekt der Klasse `NetComponent` realisiert. Es werden restrict-, agree- und value-Komponenten unterstützt.

- Eine Fehlerbeschreibung nach Abschnitt 5.2 enthält eine optimale Strukturbeschreibung und eine Menge von minimalen Diagnosen der Fehlersimulation. Sie wird durch ein Objekt der Klasse `CDGParseResult` dargestellt. Eine Diagnose der Fehlersimulation wird durch ein Objekt der Klasse `NetSimulationResult` repräsentiert. Eine Strukturbeschreibung nach Definition 4.1.4 besteht aus jeweils einem Baum pro Analyseebene. Dependenzbäume werden durch Objekte der Klasse `CDGParseTree` beschrieben. Die Regelverletzungen und die beteiligten Dependenzkanten nach Definition 4.3.1 werden durch Objekte der Klasse `CDGParseViolation` dargestellt.

Die eigentliche Funktionalität des integrativen Verfahrens wird durch eine C++-Bibliothek zur Verfügung gestellt. Es werden insgesamt 11 Java-Klassen für die Java-Schnittstelle zum integrativen Verfahren benötigt. Die Implementation der Java-Klasse `GrammarChecker` wird in der C++-Bibliothek vorgenommen. Die weiteren Java-Klassen sind meist Anpassungen den zugehörigen C++-Klassen aus der Bibliothek.

Klasse GrammarChecker

Die Klasse `GrammarChecker` stellt die Schnittstelle zum integrativen Parsingverfahren zur Verfügung. Die Klasse ist statisch und muß vor der ersten Benutzung initialisiert werden. Nach der Benutzung sollte sie wieder freigegeben werden.

```
class GrammarChecker {
    void initialize();
    void release();
    void loadGrammar(String filename);
    CDGParseResult [] parseSentence(String sentence);
}
```

Klasse CDGParseResult

Ein Objekt der Klasse `CDGParseResult` repräsentiert eine Fehlerbeschreibung, die im wesentlichen aus einer Menge von Bäume, den verletzten Regeln und den minimalen Diagnosen als Ergebnis der Fehlersimulation besteht. Die Wertebindung, d.h. die den Wortformen zugeordneten Merkmalsstrukturen, können ausgelesen werden. Es wird die Wertebindung des CDG-Parsingverfahrens verwendet. Die einzelnen minimalen Diagnosen können unterschiedliche Wertebindungen haben.

```
class CDGParseResult {
    CDGParseViolation [] getViolations();
    String [] getLevels();
    CDGParseTree findTree(String level);
    int getNoOfNodes();
    String getNodeName(int i);
    SlotValuePairList getNodeBinding(int i);
    NetSimulationResult [] getNetSimulationResults();
}
```

Klasse NetSimulationResult

Ein Objekt der Klasse `NetSimulationResult` repräsentiert eine minimale Diagnose als Ergebnis der Fehlersimulation. Für eine Faktenfehlerdiagnose kann ein Korrekturvorschlag angefordert werden. Die Wertebindung kann vor und nach der Simulation ausgelesen werden.

```
class NetSimulationResult {
    int getNoOfFailedComponents();
    NetComponent getFailedComponent(int i);
    SlotValuePairList getInitialNodeBinding(int i);
    SlotValuePairList getNodeBinding(int i);
    boolean isFactDiagnosis();
    String getNodeCorrection(int i);
}
```

Literaturverzeichnis

[Aho 1988] Aho, A.V., Sethi, R. und Ullman, J.D. *Compilerbau.* Bonn, etc. : Addison-Wesley, 1988.

[Alpert 1998] Alpert, S.R., Singley, M.K. und Fairweather, P.G. *AlgeBrain : Intelligent Tutoring on the Web.* IBM Research Report RC21358, Computer science/mathematics, Yorktown Heights, NY : IBM Watson Research Center, 1998.

[Allen 1995] Allen, J.F. *Natural Language Understanding.* Redwood City, Calif., etc. : Benjamin/Cummings, 1995.

[Carbonell 1983] Carbonell, J.G. und Hayes, P.J. „Recovery Strategies for Parsing Extragrammatical Language." In: *American Journal of Computational Linguistics.* 9:123-146, 1983.

[Chen 1989] Chen, L. und Kurtz, B.K. „XTRA-TE: Using Natural Language Processing Software to Develop an ITS for Language Learning." In: *Proc. of the 4th International Conference on AI and Education.* Amsterdam, Netherlands, 54-63, 1989.

[Clocksin 1994] Clocksin, W.F. und Mellish, C.S. *Programming in Prolog.* Berlin, etc. : Springer, 1994.

[Davis 1984] Davis, R. „Diagnostic Reasoning Based on Structure and Behavior." *Artificial Intelligence.* 24:347-410, 1984.

[Eckstein 1998] Eckstein, R., Loy, M. und Wood, D. *Java Swing.* Beijing, etc. : O'Reilly, 1998.

[Eisenberg 1998] Eisenberg, P., Gelhaus, H., Wellmann, H., Henke, H. und Sitta, H. *Duden, Grammatik der deutschen Gegenwartssprache.* DUDEN Verlag, Bd.4, 1998.

[Flanagan 1998] Flanagan, D. *Java in a Nutshell.* Cambridge, etc. : O'Reilly, 1998.

[Foth 1999a] Foth, K. *Disjunktive Lexikoninformation im eliminativen Parsing.* Universität Hamburg, Fachbereich Informatik, Studienarbeit, 1999.

[Foth 1999b] Foth, K. *Transformationsbasiertes Constraint-Parsing.* Universität Hamburg, Fachbereich Informatik, Diplomarbeit, 1999.

[Foth 2000] Foth, K., Menzel, W. und Schröder, I. „A Transformation-Based Parsing Technique with Anytime Properties.“ In : *Proceedings of the 4th International Workshop on Parsing Technologies.* IWPT 2000, 89-100, Trento, Italien, 2000.

[Freuder 1992] Freuder, E.C. und Wallace, R.J. „Partial Constraint Satisfaction.“ *Artificial Intelligence.* 58:21-70, 1992.

[Gazdar 1985] Gazdar, G., Klein, E., Pullum, G.K. und Sag, I.A. *Generalized Phrase Structure Grammar.* Cambridge, Mass.: Havard University Press, 1985.

[Genesereth 1987] Genesereth, M.R. und Nilson, N.J. *Logical Foundations of Artificial Intelligence.* Los Altos, California : Morgan Kaufmann Publishers Inc, 1987.

[Gordon 1998] Gordon, R. *Essential JNI : Java Native Interface* . Upper Saddle River, NJ : Prentice Hall PTR, 1998.

[Grewendorf 1996] Grewendorf, G., Hamm, F. und Sternefeld, W. *Sprachliches Wissen: eine Einführung in moderne Theorien der grammatikalischen Beschreibung.* Frankfurt am Main : Suhrkamp, 1996.

[Hagen 1994] Hagen, L.K. „Unification-Based Parsing Applications for Intelligent Foreign Language Tutoring Systems.“ *CALICO Journal.* 12(2-3):5-31, 1994.

[Harper 1993] Harper, M.P. und Helzermann, R.A. *Managing Multiple Knowledge Sources in Constraint-Based Parsing of Spoken Language.* Technical Report EE 94-16, School of Electrical Engineering, Purdue University, West Lafayette, IN 47907-1285, 1993.

[Harper 1995] Harper, M.P et al. „Implementation Issues in the Development of the PARSEC Parser.“ In: *Software - Practice and Experience.* 25(8):831-862, 1995.

[Hellwig 1989] Hellwig, P. „Parsing - Grundlagen“. In: Batori, I.S, Lenders, W. und Putschke, W. (Hrsg.): *Computational Linguistics. An International Handbook on Computer Oriented Language Research and Applications.* Berlin: Walter de Gruyter, 1989.

[Hopcroft 1994] Hopcroft, J. und Ullman, J. *Einführung in die Automatentheorie, formale Sprachen und Komplexitätstheorie.* Bonn : Addison-Wesley, 1994.

[Kernighan 1990] Kernighan, B.W. und Ritchie, D.M. *Programmieren in C : mit dem C-Reference Manual in deutscher Sprache.* München, etc. : Hanser, 1990.

[de Kleer 1986] de Kleer, J. „An Assumption-Based TMS.“ *Artificial Intelligence.* 32(1):97-130, 1986.

[de Kleer 1992] de Kleer, J., Mackworth, A.K. und Reiter, R. „Characterizing Diagnoses and Systems.“ *Artificial Intelligence*. 56(2-3):197-222, 1992.

[Kumar 1992] Kumar, V. „Algorithms for Constraint Satisfaction Problems: A Survey.“ *AI Magazine*. 13(1):32-44, 1992.

[Kunze 1975] Kunze, J. *Abhängigkeitsgrammatik*. studia grammatica XII, Berlin: Akademie-Verlag, 1975.

[Lelouche 1999] Lelouche, R. „Intelligent Tutoring Systems from Birth to Now.“ *Künstliche Intelligenz*. 13(4):5-11, 1999.

[Lusti 1992] Lusti, M. *Intelligente Tutorielle Systeme : Einführung in wissensbasierte Lernsysteme*. Handbuch der Informatik, Bd. 15.4, München : Oldenburg Verlag, 1992.

[Maruyama 1990] Maruyama, H. „Structural Disambiguation with Constraint Propagation.“ In: *Proc. of the 28th Annual Meeting of the ACL*. 31-38, 1990.

[Melcuk 1988] Melcuk, I. A. *Dependency Syntax: Theory and Practice*. Albany, N.Y. : State Univ. of New York Press, 1988.

[Mellish 1989] Mellish, C. „Some Chart-Based Techniques for Parsing Ill-Formed Input.“ In: *Proc. 27th Annual Meeting of the ACL*. 102-109, Vancouver, 1989.

[Menzel 1988] Menzel, W. „Error Diagnosing and Selection in a Training System for Second Language Learning.“ In: *Proc. 12th International Conference on Computational Linguistics*. Coling'88, 414-419, Budapest, 1988.

[Menzel 1990] Menzel, W. „Anticipation-Free Diagnosis of Structural Faults.“ In: *Proc. 13th International Conference on Computational Linguistics*. Coling'90, 422-424, Helsinki, 1990.

[Menzel 1992a] Menzel, W. *Modellbasierte Diagnose in Sprachlehrsystemen*. Sprache und Information, Tübingen : Max Niemeyer Verlag, 1992.

[Menzel 1994] Menzel, W. „Parsing of Spoken Language under Time Constraints.“ In: *Proc. 11th Europ. Conf. on Artificial Intelligence*. ECAI-94, Amsterdam, 560-564, 1994.

[Menzel 1995] Menzel, W. „Robust Processing of Natural Language“. In: *Proc. 19th Annual German Conference on Artificial Intelligence*. Bielefeld, Lecture Notes in Computer Science 981, Berlin, etc. : Springer, 19-34, 1995.

[Menzel 1998a] Menzel, W. „Constraint Satisfaction for Robust Parsing of Spoken Language.“ *Journal of Experimental and Theoretical Artificial Intelligence*. 10(1):77-89, 1998.

[Menzel 1998b] Menzel, W. und Schröder, I. „Constraint-Based Diagnosis for Intelligent Language Tutoring Systems.“ In: *Proc. IT&KNOWS, XV. IFIP World Computer Congress*. Vienna/Budapest, 484-497, 1998.

[Menzel 1998c] Menzel, W. und Schröder, I. „Model-Based Diagnosis under Structural Uncertainty.“ In: *Proc. of the 13th European Conference on Artificial Intelligence.* ECAI'98, Brighton UK, 284-288, 1998.

[Menzel 1998d] Menzel, W. und Schröder, I. „Decision Procedures for Dependency Parsing using Graded Constraints.“ In: *Proc. Coling-ACL Workshop on Processing of Dependency Grammars.* Montreal, Canada, 78-87, 1998.

[Menzel 1998e] Menzel, W. und Schröder, I. „Error Diagnosis for Language Learning Systems“. In: *Proc. Int. Conf. Natural Language Processing and Industrial Applications.* NLP&IA'98, Moncton, Canada, 45-51, 1998.

[Menzel 2000] Menzel, W. „Dependenzgrammatik und Semantische Netze.“ Erscheint in: Agel, V., Eichinger, L.M., Eroms, H.-W., Hellwig, P., Heringer, H.J. und Lobin, H. (Hrsg.): *Dependenz und Valenz - Ein internationales Handbuch der zeitgenössischen Forschung.* Berlin, New York : Walter de Gruyter, 2000.

[Mitrovic 1999] Mitrovic, A. und Ohlsson, S. „Evaluation of a Constraint-Based Tutor for a Database Language“. *Int. Journal of Artificial Intelligence in Education.* 10(3-4):238-256, 1999.

[Naumann 1994] Naumann, S. und Langer, H. *Parsing : eine Einführung in die maschinelle Analyse natürlicher Sprache.* Stuttgart : Teubner Verlag, 1994.

[Puppe 1996] Puppe, F., Gappa, U., Poeck, K. und Bamberger, S. *Wissensbasierte Diagnose- und Informationssysteme.* Berlin, etc.: Springer Verlag, 1996.

[Reiter 1987] Reiter, R. „A Theory of Diagnosis from First Principles.“ *Artificial Intelligence.* 32:57-59, 1987.

[Sanders 1991] Sanders, R. „Error Analysis in Purely Syntactic Parsing of Free Input.“ In: *CALICO Journal.* 9:72-89, 1991.

[Schneider 1998] Schneider, D. und McCoy, K. „Recognizing Syntactic Errors in the Writing of Second Language Learners.“ In: *Proc. of Coling-ACL-98.* University of Montreal, Montreal, Quebec, Canada, August 10-14, 1998.

[Schwind 1986] Schwind, C.B. „Overview of an Intelligent Language Tutoring System“. In: *Proc. 2nd Int. Conf. on Artificial Intelligence.* CIIAM'86. Marseile, Editions Hermes, Paris: 389-409, 1986.

[Schwind 1988] Schwind, C.B. „Sensitive Parsing: Error Analysis and Explanations in an Intelligent Language Tutoring System.“ In: *Proc. 12th International Conference on Computational Linguistics.* Budapest: 608-613, 1988.

[Schwind 1995] Schwind, C. „Error Analysis and Explanation in Knowledge Based Language Tutoring.“ *Computer Assisted Language Learning.* 8(4):295-324, 1995.

[Schröder 1995] Schröder, I. *Analyse natürlicher Sprache durch Beschränkungserfüllung.* Universität Hamburg, Fachbereich Informatik, Studienarbeit, 1995.

[Schröder 1998] Schröder, I. et al. *CDG-Benutzerhandbuch.* Memo HH-13, Projekt DAWAI, Universität Hamburg, Fachbereich Informatik, 1998.

[Schröder 1999] Schröder, I., Menzel, W., Foth, K. und Schulz, M. „Dependency Modeling with Restricted Constraints." In: *Traitement Automatique de Langage.* 38(1) (eingereicht), 1999.

[Shieber 1987] Shieber, S.M. *An Introduction to Unification-Based Approaches to Grammar.* CSLI Lecture Notes No.4, CSLI, Stanford,Calif.:University of Chicago Press, 1987.

[Stede 1992] Stede, M. „The Search for Robustness in Natural Language Understanding." *Artificial Intelligence Review.* 6(4):383-414, 1992.

[Stockfleth 1999] Stockfleth, N. *Modellbasierte Diagnose in der Sprachverarbeitung.* Universität Hamburg, Fachbereich Informatik, Studienarbeit, 1999.

[Stroustrup 1992] Stroustrup, B. *Die C++ Programmiersprache.* Bonn : Addison-Wesley, 1992.

[Tesnière 1959] Tesnière, L. *Éléments de syntaxe structurale.* Paris: Klincksieck, 1959.

[Tsang 1993] Tsang, E. *Foundations of Constraint Satisfaction.* London: Academic Press, Harcort Brace and Company, 1993.

[Uszkoreit 1991] Uszkoreit, H. „Strategies for Adding Control Information to Declarative Grammars." In: *Proc. 29th Annual Meeting of the ACL.* 237-245, 1991.

[Weischedel 1978] Weischedel, R.M., Voge, M.V. und James, M. „An Artificial Intelligence Approach to Language Instructions." *Artificial Intelligence.* 10:225-240, 1978.

[Weischedel 1983] Weischedel, R.M. und Sondheimer, N.K. „Meta-rules as a Basis for Processing Ill-Formed Input." In: *American Journal of Computational Linguistics.* 9(3-4):161-177, 1983.

[Witschital 1990] Witschital, P. *Intelligente tutorielle Systeme in der Programmierausbildung.* Technische Universität Braunschweig, Dissertation, 1990.

Erklärung gemäß §23.9 DPO

Hiermit erkläre ich, daß ich die vorliegende Arbeit selbständig durchgeführt und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

....

Ort, Datum (Niels Stockfleth)