Zusammenfassung der Magisterarbeit

"Auxiliarkonstruktionen und maschinelle Sprachverarbeitung"

1. Einordnung der Arbeit

Im weiten Feld der Computerlinguistik finden sich verschiedene Auffassungen hinsichtlich der Vorgehensweisen bei der Umsetzung maschineller Verarbeitung natürlicher Sprache. In der vorliegenden Arbeit spiegelt sich vor allem die Auffassung der maschinellen Sprachverarbeitung wider, in der natürliche Sprache als Teil eines kognitiven Systems aufgefaßt wird. Demnach wird hier versucht, für die Sprachverarbeitung nötige kognitive Prozesse zu beschreiben und auf den Computer zu übertragen.

Zum einen versteht sich die Arbeit daher in weiten Teilen als theoriegeleitet und verwurzelt in der kognitiven Linguistik. Andererseits werden die theoretisch gewonnenen Erkenntnisse auch mithilfe eines Expertensystems validiert und in ein System zur maschinellen Übersetzung eingebettet, was den direkten Bezug zur Computerlinguistik wiederherstellt.

2. Vorgehensweise

Untersuchungsgegenstand der Arbeit sind Auxiliarkonstruktionen. Die Klasse der Auxiliare umfaßt im allgemeinen Hilfsverben und Modalverben (auch modale Hilfsverben genannt), bisweilen werden ihr aber auch andere sprachliche Elemente, wie z.B. der Infinitmarker *zu* zugeordnet. Es bedarf also zunächst einer Definition von Auxiliarität, welche den grundlegenden Untersuchungsgegenstand für die weitere Arbeit liefert. Eine solche Definition kann nur aufgrund einer Beschreibung von Auxiliarphänomenen ausgearbeitet werden, was die Aufgabe des ersten Abschnitts des Haupteils ist.

Im zweiten Teil werden nun die gewonnenen Erkenntnisse in eine Theorie eingebettet, die den Anspruch erhebt, kognitiv plausibel zu sein. Die Kognitionswissenschaft und damit auch die Meinung darüber, was als kognitiv plausibel zu gelten hat, ist in zumindest zwei Lager gespalten, die sich zuweilen heftig bekämpfen: Auf der einen Seite finden sich die Anhänger des Symbolverarbeitungs-Paradigmas, auf der anderen Seite die Konnektionisten. Symbolverarbeitung fußt auf zwei Annahmen, die in der Mitte der siebziger Jahre publiziert wurden: der Hypothese über physikalische Symbolsysteme (Newell/Simon 1976) und der Computertheorie des Geistes (Fodor 1975). In diesem Paradigma wird Kognition als nachvollziehbarer, symbolmanipulierender Prozeß angenommen, m.a.W. wird eine Repräsentationsebene im menschlichen Hirn angenommen, die exakt, im Stile eines Computerprogramms, beschrieben und analysiert werden kann. Der Konnektionismus auf der anderen Seite leugnet die Existenz einer solchen konzeptionell-symbolischen Repräsentationsebene: Symbole sind hier keine Bestandteile der kognitiven Ebene, sondern Abstraktionen von dieser. Die eigentlich kognitive Arbeit vollzieht sich auf sub-symbolischer Ebene. Das Problem einer solchen Sichtweise liegt auf der Hand: durch das Fehlen einer konzeptionellen Ebene können kognitive Vorgänge nicht beschrieben, sondern allenfalls nachgeahmt werden, was bei sehr komplexen Gebilden, wie in diesem Fall dem menschlichen Gehirn oder zumindest einzelnen Teilen, schwierig bis unmöglich sein dürfte.

Ohne die Plausibilität des konnektionistischen Ansatzes für viele Teilgebiete der Sprachverarbeitung zu leugnen, werden in der vorliegenden Arbeit – jedenfalls für die betroffenen sprachlichen Ebenen der Syntax und der Morphologie – symbolische Repräsentationen angenommen, auf die sprachliche Ausdrücke abgebildet werden. Die Abbildung erfolgt in eine Struktur, in der morphosyntaktische und syntaktische Phänomene durch strukturelle Relationen ausgedrückt werden. Als einbettende Theorie wurde die Prinzipien und Parameter-Theorie (PPT, eine neuere Ausprägung der Generativen Grammatik

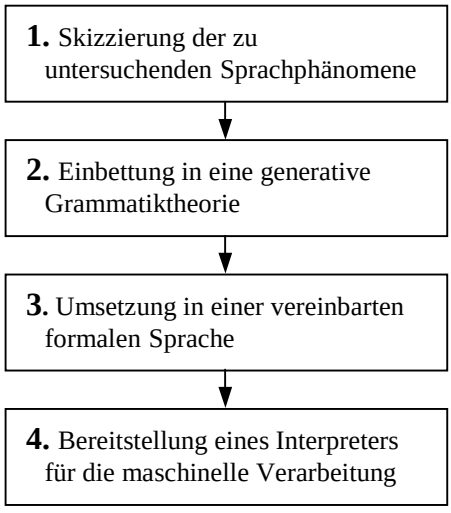
nach Chomsky 1981) ausgewählt, welche die möglichen Datenstrukturen und auf sie angewendbare Algorithmen restringiert.

Es bedarf nun noch einer Form, in der das im ersten Teil gewonnene und im zweiten Teil in die PPT implementierte linguistische Wissen für den Computer interpretierbar gemacht werden kann. Eine solche Form liefert u.a. die von Rolshoven (1987) entwickelte, modular konzipierte, deklarative, objektorientierte linguistische Programmiersprache LPS-OOP. Sie ist eigens für Linguisten konzipiert, nahe an der linguistischen Notation angelehnt und führt das Konzept der Objektorientierung in die linguistische Beschreibung ein. Die Aufgabe des dritten Hauptteilkapitels ist demnach die Umsetzung des linguistischen Wissens in die Programmiersprache LPS-OOP.

Im Schlußteil beschäftigt sich die Arbeit mit der Evaluation der entwickelten linguistischen Teiltheorien anhand des Expertensystems VisualGBX (Lalande 1997), sowie ihrer Anwendung im System LPS (einem System zur Maschinellen Übersetzung, vgl. Rolshoven 2002). Beide Systeme stellen einen Interpreter für LPS-OOP, der die Anwendung des in der Arbeit beschriebenen, eingebetteten und formalisierten linguistischen Wissens möglich macht.

Die Gliederung der Arbeit hält sich damit an die oben verdeutlichte Vorgehensweise, wie menschliche Sprachfähigkeit auf einem künstlichen Berechnungssystem modelliert werden kann, was *Abb. 1* nochmals graphisch darstellt:

(Abb. 1)



3. Auxiliare

3.1 Phänomenbeschreibung

Im ersten Drittel des Hauptteils setzt sich die Arbeit in erster Linie deskriptiv mit der Verwendung und dem Verhalten von Auxiliaren auseinander.

Zunächt bedarf es dabei einer Definition von Auxiliaren, um den Gegenstandsbereich einzugrenzen. Eine der auftretenden Schwierigkeiten bildet dabei die Tatsache, daß Auxiliare in vielen Sprachen auch als Vollverb gebraucht werden können. So stellt sich die Frage, ob Auxiliare als eine Untergruppe der Verben oder als eigenständige Kategorie betrachtet werden müssen. Es bedarf hier also einer Lösung, die sowohl den Verb- als auch den Eigenständigkeitscharakter von Auxiliaren in die Analyse mit einfließen läßt und die darüberhinaus in der Lage ist, beobachtbare Phänomene des Sprachwandels und des Sprachvergleichs zu erfassen. Der in der Arbeit verfolgte Ansatz für eine Definition basiert stark auf einer Analyse von Heine (1993): Auxiliare werden als Klasse betrachtet, die dem diachronen Prozeß der Grammatikalisierung (vgl. Meillet 1912) unterliegt. Im Verlauf dieses Prozesses erwirbt eine ehemals autonome lexikalische Einheit die Funktion einer abhängigen grammatischen Kategorie, was sich phonologisch, morphologisch, syntaktisch und semantisch ausdrückt.Der Prozeß hat jedoch auch synchrone Auswirkungen, da Grammatikalisierung sich stufenweise vollzieht und diese Stufen im Laufe des Prozesses unidirektional durchlässig sind. Es ist also möglich, daß zu einem Zeitpunkt des Sprachwandels sprachliche Einheiten sich sowohl als autonome lexikalische Einheit, als auch als Marker für grammatische Kategorien nutzen lassen. Für die Entwicklung von Verben zu TAM (Tense, Aspect, Modus) Markern werden insgesamt sieben Stufen angenommen, auf denen sich die voneinander unabhängigen Prozesse der Desemantisierung, Dekategorisierung, Klitisierung und Erosion vollziehen. Auxiliare nun sind diejenigen sprachlichen Einheiten, die sich auf einer der definierten Stufen, m.a.W. auf der *Verb-to-TAM-Chain*, befinden.

Nach dieser notwendigen Definition von Auxiliaren werden drei ihrer wichtigsten Eigenschaften näher untersucht: Dabei wurde festgestellt, daß Besonderheiten auf der Ausdrucksseite der Auxiliare mit Phänomenen auf der Inhaltsseite des gesamten sprachlichen Ausdrucks, in den sie eingebettet sind, korrespondieren. Die Arbeit geht hier vor allem auf Phänomene des Deutschen ein, setzt sie aber in einen indoeuropäischen Kontext (vgl. auch 3.3):

1. An Auxiliaren manifestieren sich die Finitheitsmerkmale eines Satzes. Die Finitheit steht im engen Zusammenhang mit der Prädikation: Finite Sätze weisen ein Subjekt

auf, das in Kongruenz mit dem finiten verbalen Bestandteil steht und den höchsten strukturellen Kasus (in Nominativsprachen wie den Indoeuropäischen ist dies der Nominativ) zugewiesen bekommt. Ist der Satz infinit, so fehlen sowohl dieser Kasus wie auch die Kongruenz.

2. Treten Auxiliare nicht als Vollverben auf, so korrespondieren sie mindestens mit einem infiniten Vollverb; Konstruktionen solcher Art nennt man analytische Bildungen. Aus dem Zusammenspiel der verbalen Bestandteile lassen sich die temporalen, modalen und aspektuellen Eigenschaften eines Satzes ableiten.
3. Die Auxiliarselektion (im Deutschen *sein*/*werden* vs. *haben*) schließlich bestimmt die Anzahl der im Satz auftretenden Argumente: Externe Argumente werden von der infiniten Verbform PartizipII blockiert. Damit diese Argumente dennoch im Satz auftreten können, müssen sie durch das Auxiliar *haben* deblockiert werden. Eine solche deblockierende Wirkung weist das Auxiliar *sein* nicht auf, weshalb Sätze im Passiv ihrer externen Argumente verlustig gehen.

Diese deskriptiv beschriebenen Phänomene müssen nun durch struktuelle Relationen innerhalb einer einheitlichen Repräsentation dargestellt werden können. Dabei sind folgende Fragen zu beantworten:

1. Wie läßt sich der Unterschied zwischen Vollverben und Auxiliaren (definiert durch ihre Position auf dem Verb-to-TAM-Chain) erfassen?
2. In welchem Verhältnis steht das Auxiliar zum Subjekt? Wie erlangt es seine Finitheitsmerkmale?
3. In welchem Verhältnis steht das Auxiliar zum infiniten Vollverb? Wie werden die morphosyntaktischen Kategorien abgeglichen?
4. Wie kann der Blockier-/Deblockiermechanismus beschrieben werden?

3.2 Interpretation in der generativen Theorie

Grundlage der traditionellen generativen Sprachtheorie bildet die Annahme, alle gebildeten Sätze einer Sprache ließen sich zurückführen auf zugrundeliegende, aus Phrasensrukturregeln abgeleitete Datenstrukturen, auf die Algorithmen in Form von Transformationsregeln angewendet werden, die diese zugrundeliegenden Tiefenstrukturen in abgeleitete Oberflächenstrukturen überführen. Innerhalb der seit Ende der siebziger Jahre entwickelten Rektions- und Bindungstheorie (Government & Binding-Theory – GB, vgl. Chomsky 1981) wurden den Algorithmen Beschränkungen auferlegt, die gewährleisten sollen, daß aus den Transformationen keine ungrammatischen Sätze abgeleitet werden können. Die neueste

Ausprägung der generativen Sprachtheorie, das Minimalistische Programm (*Minimalist Program* – MP, vgl. Chomsky 1995) verzichtet auf die Unterscheidung zwischen Oberflächen- und Tiefenstruktur. Stattdessen ist die Sichtweise des MP ist eine dynamische, da nicht von einer vollständig aufgebauten Struktur ausgegangen wird, innerhalb der sich durch verschiedene Subtheorien gestützte Bewegungen vollziehen können. Der Strukturaufbau wird stattdessen in einem bottom-up-Verfahren durch die Interaktion zweier Prozesse gewährleistet, einerseits dem Prozeß der *lexical insertion*, die über einen sogenannten *Merger* läuft und für den Aufbau von lexikalischen Phrasen verantwortlich ist, andererseits dem Prozeß, der Bewegungen steuert, durch die *Checking-Theorie* motiviert ist und Strukturen funktionaler Phrasen erzeugt.

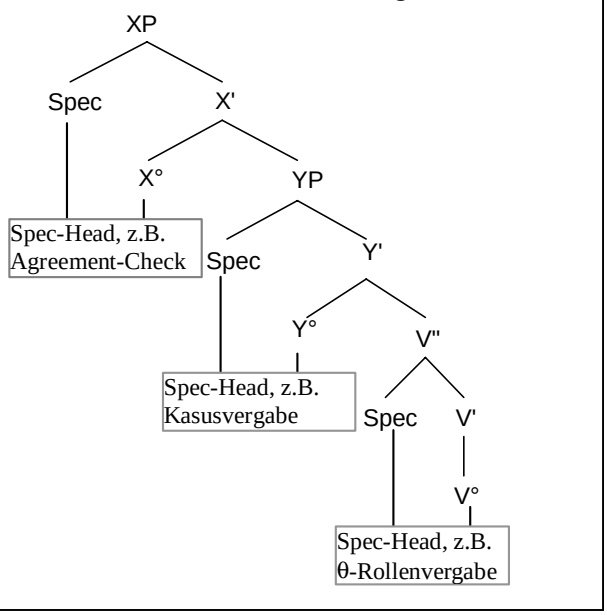
Letztlich gemeinsam ist den verschiedenen Ausprägungen der Theorie, daß Sätze einer Sprache auf eine grammatische Struktur abgebildet werden können, die Positionen bereitstellt, in denen durch Bewegung (z.B. morphosyntaktische) Features abgeglichen werden können.

Um die im letzten Abschnitt aufgeworfenen Fragen zu beantworten, muß demnach zuerst geklärt werden, welche Position in der Struktur die Basisposition für Auxiliare stellt, welche Bewegungen das Auxiliar innerhalb der Struktur vollführt und welche Merkmale wo abgeglichen oder gesetzt werden.

Schon die Valenzgrammatik (vgl. Tesniere 1959) sah das Verb als den zentralen Knoten (*Zentralnexus*) des gesamten Satzes an, da der Satz zumindest aus dem Verb und den von ihm geforderten (und damit von ihm abhängigen) Aktanten bestehen muß. Diese Vorstellung läßt sich auch auf die generative Theorie übertragen, da hier in den Kopfpositionen des Satzes immer ein Verb oder zumindest ein mit ihm korrespondierendes Element - Auxiliar oder Komplementierer - zu finden ist. Von seiner Basis-Kopfposition kann sich das Verb nur in andere (und zwar funktionale) Kopf- und nicht etwa auf Spezifiziererpositionen bewegen (vgl. 3.2). Diese funktionalen Köpfe expandieren gemäß dem X-bar-Schema zu Phrasen, die neben der Kopfposition noch einen Spezifizierer und ein Komplement dominieren. Der Spezifizierer kann durch bestimmte lexikalische XPs belegt werden, das Komplement ist wiederum eine Phrase einer funktionale Kategorie, oder - auf der untersten Ebene - die Verbalphrase. Zwischen dem Kopf und dem Spezifizierer einer Phrase werden die Features der Phrasen-Kategorie mittels Spec-Head-Beziehung gecheckt bzw. zugewiesen. Als Veranschaulichung dieser abstrakten Vorüberlegung soll *Abb. 2* dienen:

(Abb. 2) Funktionale Kategorien und VP

X und Y: funktionale Kategorien



Im Laufe der Entwicklung der generativen Theorie wurde eine Fülle von funktionalen Kategorien angenommen, die in der neuesten Ausprägung, dem v.a. auf dem Ökonomieprinzip beruhenden MP wieder rigoros zusammengestrichen wurde. Übrig blieben drei Kategorien:

- Die Prädikationsphrase (PrP) als Domäne für Prädikation selegiert eine Verbalphrase zum Komplement, ist dieser also direkt übergeordnet.
- Die Tempusphrase (TP) als Domäne für Zeitrelationen, Finitheit und Kongruenz selegiert die PrP als Komplement.
- Die Komplementiererphrase (CP) als Domäne für den Satzmodus selegiert schließlich die TP als Komplement.

Auf Basis dieser Überlegungen beantwortet die Arbeit die im letzten Abschnitt aufgeworfenen Fragen folgendermaßen:

Der Unterschied zwischen Vollverben und Auxiliaren drückt sich durch unterschiedliche Basispositionen aus. Als Basisposition von Vollverben wird die Kopfposition der Verbalphrase, als Basisposition von Auxiliaren die Kopfposition der Prädikationsphrase angenommen. In dieser Basisposition wird der Status des Vollverbs regiert und ein eventuell vorhandenes (im Falle des Auxiliars *haben*) externes Argument deblockiert. Von der Basisposition bewegt sich das Auxilar in die Kopfposition der Tempusphrase und gleicht dort die Tempusmerkmale des Satzes, sowie die Kongruenzmerkmale des Subjekts ab und weist diesem (im finiten Kontext) schließlich den Kasus Nominativ zu. Ist ein overter Abgleich von

Modusmerkmalen nötig, so bewegt sich das Auxiliar in die Kopfposition der maximalen Projektion des Satzes, der Komplementiererphrase.

3.3 Formalisierung in einer definierten Programmiersprache

Die gewonnenen Erkenntnisse werden in der Arbeit in der Programmiersprache LPS-OOP formalisiert. In LPS-OOP werden Knoten im Strukturbaum durch Objekte von Knotenklassen repräsentiert. Diese Objekte erben aus allgemeineren Klassen, welche Methoden für den Merkmalabgleich oder die Merkmalszuweisung bereitstellen. Die Klassen sind in Modulen organisiert; diese Module können auf einzelsprachlicher und/oder sprachübergreifender Ebene angesiedelt sein.

Kongruenzphänomene und die Kasuszuweisung lassen sich für alle indoeuropäischen Sprachen gemeinsam formulieren. Demnach werden die dafür ausgearbeiteten Module (*EurAgreement* und *EurCase*) direkt auf der sprachübergreifenden indoeuropäischen Ebene angesiedelt. Die Ermittlung der Zeitrelationen und die Vergabe der Thetarollen (hier vor allem der Deblockierungsmechanismus) verläuft im Deutschen stark idiosynkratisch. Die beteiligten Module (*DtsTempus* und *DtsTheta*) werden demnach zunächst einzelsprachlich formuliert. Sprachübergreifende Phänomene, wie das für ein Argument notwendige Erlangen einer Theta-Rolle, wird wiederum in einem indoeuropäischen Modul (*EurTheta*) geprüft.

Veranschaulicht wird die Formalisierung hier am Beispiel des für die Kongruenz zuständigen Moduls *EurAgreement*. Kongruenz wird über eine Methode gecheckt, die versucht, Subjekt und Finitum hinsichtlich der Attribute Numerus und Person zu unifizieren. Methoden haben in LPS-OOP die Form von Prolog-Klauseln. Unifizierung erfolgt über das BuiltIn-Prädikat `UnifyValues(ErstesArgument, ZweitesArgument, zuUnifizierendesAttribut)`, so daß die Kongruenzmethode folgende Form hat:

```
agreement(Subjekt, Finitum):-
      UnifyValues(Subjekt, Finitum, 'Person'),
      UnifyValues(Subjekt, Finitum, 'Numerus'),
```

Gemäß dem objektorientierten Paradigma sind Methoden an Objekte gebunden. `agreement` muß an das Subjekt und das Finitum des Satzes gebunden werden, die beide diese Methode – allerdings mit komplementär instanziierten Argumenten – aufrufen:

```
CLASS SubjectAgr;
      agreement(Self, Other);
END;
CLASS FinitumAgr;
      agreement(Other, Self);
END;
```

Beide Kongruenzklassen sind Erblasser für Knotenklassen im Strukturbaum (genauer: für SpecTP und T°). Die Methode wird ausgeführt, wenn ihre beiden Argumente in einem Strukturbaumknoten (hier: T'') instanziiert sind. Sie validiert, wenn Subjekt und Finitum hinsichtlich der Attribute Numerus und Peron unifiziert werden können. Die anderen erwähnten Module werden entsprechend formuliert.

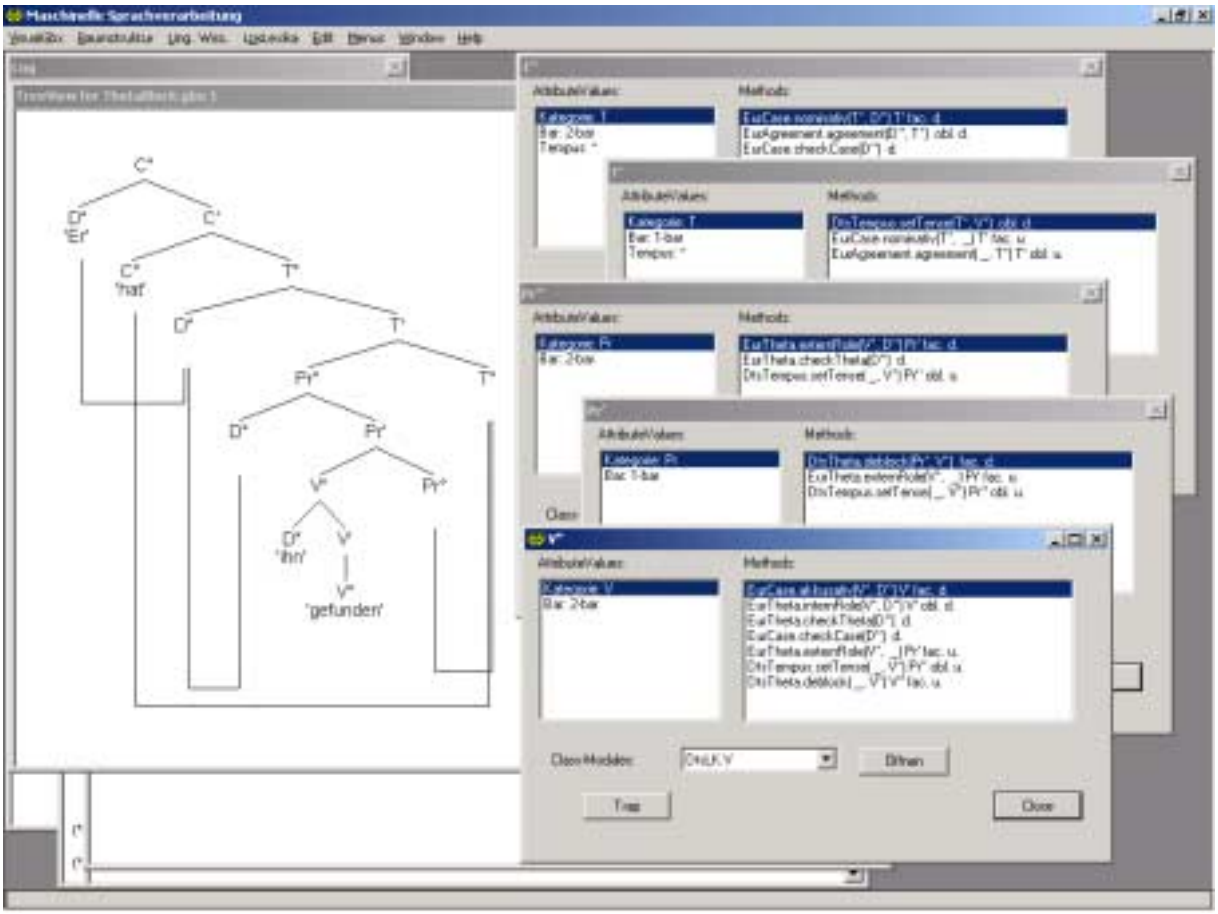
4. Computerlinguistische Umsetzung

4.1 Evaluation der Ergebnisse anhand eines Expertensystems

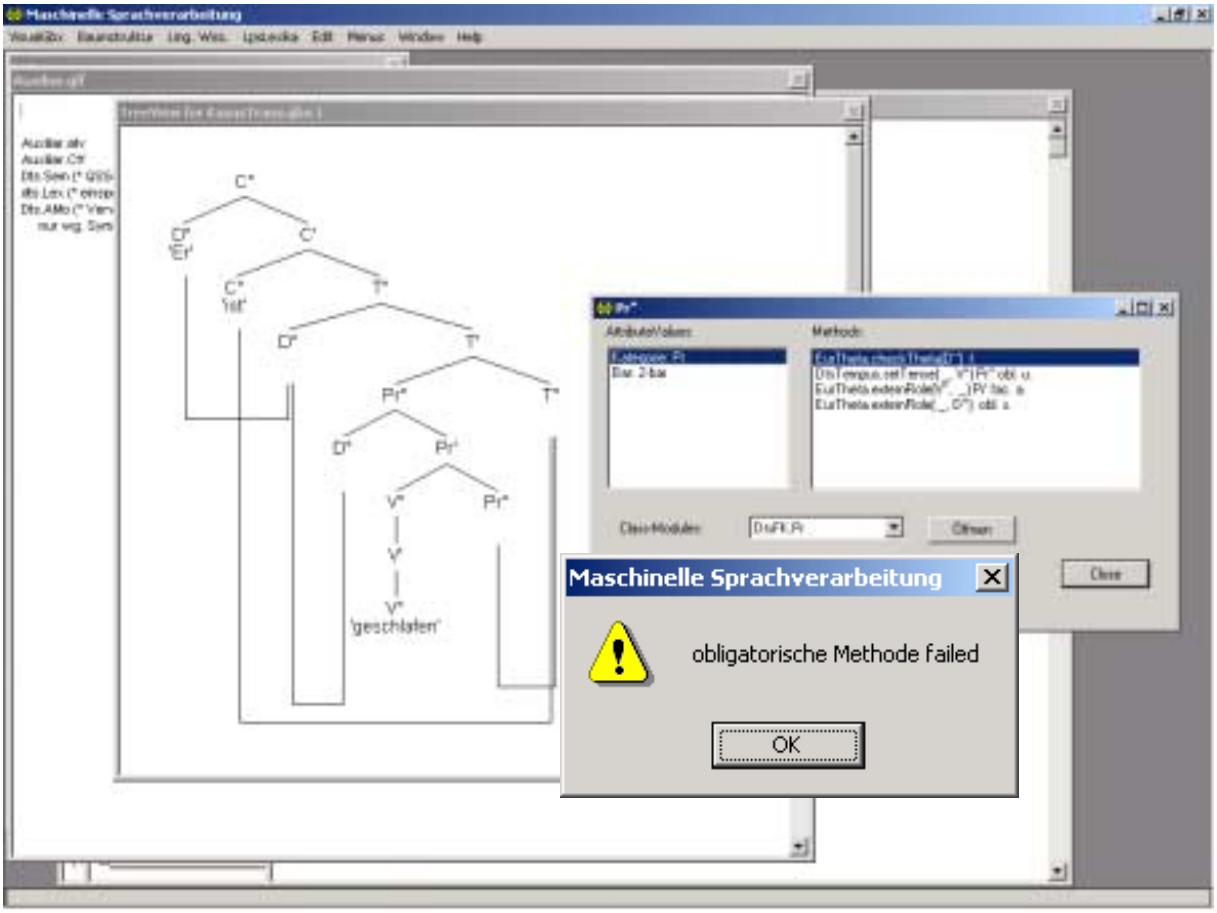
Das ausgewählte Expertensystem VisualGBX (vgl. Lalande 1997) besitzt einen Interpreter für in LPS-OOP formalisiertes Wissen. Die ausgearbeiteten Module werden mit den Modulen der Knotenklassen (aufgeteilt in lexikalische und funktionale Klassen LK und FK) und einem Lexikon als linguistisches Wissen in das System integriert und anhand von Sprachdaten evaluiert.

Die Interaktion der Module wird im folgenden anhand zweier Beispielsätze demonstriert: Zunächt wird der transitive Satz *"Er hat ihn gefunden"* analysiert und für grammatikalisch korrekt befunden. Anhand ausgewählter Knoten wird die Ausführung der für die Knotenkommunikation notwendigen Methoden gezeigt. Danach wird anhand des Satzes *"Er ist geschlafen"* gezeigt, daß das System (hier wird durch das Partizip die externe Theta-Rolle blockiert, die durch sein nicht deblockiert werden kann, er erhält also keine Theta-Rolle) mittels des linguistischen Wissens ungrammatische Konstruktionen ausschließen kann: Die Inferenzmaschine meldet das Scheitern einer obligatorischen Methode. Die Abbildungen zeigen jeweils die abgeschlossene Analyse.

(Abb. 3): Analyse des Satzes "Er hat ihn gefunden"



(Abb. 4): Analyse des Satzes "*Er ist geschlafen"



4.2 Einbettung in ein System zur maschinellen Übersetzung

Maschinelle Übersetzung besteht in der Regel aus mindestens zwei Teilprozessen: einem Parsingprozeß über die Eingabekette der Ausgangssprache und einem Generierungsprozeß der zielsprachlichen Ausgabekette. Strukturell kann die Generierung als eine Umkehrung des Parsingprozesses angesehen werden, weshalb beide Aufgaben von einem einzigen Modul übernommen werden können. Dies ist im maschinellen Übersetzungssystem LPS, in das die Ergebnisse der vorliegenden Arbeit implementiert wurden, der Fall. Der Parser/Generator des Systems wird gesteuert durch das objektorientiert in Klassen in LPS-OOP codierte Wissen, sowie durch lexikalische Information aus dem LPS-Lexikon.

In dieser Arbeit konnten wegen ihres begrenzten Umfangs natürlich nur Ausschnitte des linguistischen Wissens, das für die Analyse und Generierung von natürlicher Sprache notwendig ist, formuliert werden. Es wurde bereits darauf eingegangen, daß das Hauptaugenmerk auf der Erzeugung von Modulen beruhte, die das linguistische Wissen hinsichtlich deutscher Auxiliare enthalten.

Dabei wurde aber auch die Art der Vorgehensweise bei der Erstellung von LPS-OOP-Modulen spezifiziert, so daß ein Muster für ihre künftige Erstellung generiert wurde. Das betrifft v.a. die Schnittstellendefinition zwischen einzelnen Modulen und die Setzung von Domänen für den Abgleich morphosyntaktischer, syntaktischer und semantischer Merkmale.

Literatur

Chomsky, N. (1981): *Lectures on Government and Bindin*g. Dordrecht: Foris.

Chomsky, N. (1995): *The Minimalist Progra*m. Cambridge, Mass.: MIT Press.

Fodor, J. (1975): *The Language of Thought.* Harvard: University Press.

Heine, B. (1993): *Auxiliaries: cognitive forces and grammaticalization*. New York: Oxford University Press.

Lalande, J.-Y. (1997): *Verbstellung im Deutschen und Französischen. Unter Anwendung eines CAD-basierten Expertensystems.* Tübingen: Niemeyer (= Linguistische Arbeiten 365).

Meillet, A. (1912): "L'evolution des formes grammaticales." In: A. Meillet: *Linguistique historique et linguistique générale*. (^{2}1921) Paris: Champion.

Newell, A. & H.A. Simon (1976): "Computer Science as Empirical Inquiry: Symbols and Search." In: *Communications of the Association for Computing Machinery* 19, 113-126.

Rolshoven, J. (1987): "LPS. Eine linguistische Programmiersprache." In: U. Klenk, P. Scherber & M. Thaller (eds.): *Computerlinguistik und philologische Datenverarbeitung*. Hildesheim: Olms, 115-129.

Rolshoven, J. (2002): "Die Organisation linguistischen Wissens." <http://www.spinfo.uni-koeln.de/lehre/HS_Rolshoven/papers/LPS_Kap_2.pdf>

Tesnière, L. (1959): *Eléments de syntaxe structurale*. Paris: Klincksiek.