Verbmobil
Verbundvorhaben

Prozedurale Anforderungen an die maschinelle Sprachverarbeitung

Workshop während der Jahrestagung KI-94, Saarbrücken

Günther Görz, Wolfgang Menzel, Peter Regel-Brietzmann (Hrsg.)

Vm
Report 60
Februar 1995

Februar 1995

Workshop während der Jahrestagung KI-94, Saarbrücken

IMMD VIII – Künstliche Intelligenz
Universität Erlangen-Nürnberg
Am Weichselgarten 9
D-91058 Erlangen

Tel.: (09131) 85-9909 - 9913
e-mail: goerz@immd8.informatik.uni-erlangen.de

Gehört zum Antragsabschnitt: —

Das diesem Bericht zugrundeliegende Forschungsvorhaben wurde mit Mitteln des Bundesministers für Forschung und Technologie unter dem Förderkennzeichen 01 IV 101 H9 gefördert. Die Verantwortung für den Inhalt dieser Arbeit liegt bei dem Autor.

Inhaltsverzeichnis

Vorwort

Während der KI-Jahrestagung 1994 in Saarbrücken fand am 20. und 21. September 1994 der Workshop "Prozedurale Anforderungen an die maschinelle Sprachverarbeitung" statt. Die Kurzfassungen aller Workshopbeiträge sind im Band "KI-94 Workshops — Extended Abstracts" (Hrsg. J. Kunze und H. Stoyan, GI: Bonn, 1994) abgedruckt (S. 166–195). Da ein allgemeines Interesse der Teilnehmer an einer Veröffentlichung der Langfassungen der Vorträge bestand, haben wir mit Freude den Vorschlag von Herrn Karger aufgegriffen, diese in der Reihe der Verbmobil-Berichte zu publizieren, obwohl nicht alle Beiträge im Rahmen des Verbmobil-Projekts entstanden waren. Dafür sei ihm herzlich gedankt.

Der vorliegende Bericht enthält, von einer Ausnahme abgesehen, Ausarbeitungen aller Beiträge des Workshops. Ihnen ist die von den Organisatoren verfaßte Einleitung aus dem Workshop-Band vorangestellt.

Wir danken allen Teilnehmern für ihre engagierte Mitwirkung und die angenehme Zusammenarbeit bei der Abfassung des vorliegenden Berichts.

Erlangen-Nürnberg, im Februar 1995

Im Namen der Organisatoren
Günther Görz

Prozedurale Anforderungen an die maschinelle Sprachverarbeitung

Günther Görz
Friedrich-Alexander-Universität Erlangen-Nürnberg
Wolfgang Menzel
Universität Hamburg
Peter Regel-Brietzmann
Daimler-Benz AG, Forschungszentrum Ulm

Zusammenfassung

Die Verarbeitung gesprochener Sprache stellt besondere Anforderungen an die Architektur und die Gestaltung einzelner Komponenten in Systemen zur maschinelle Sprachverarbeitung. Der Workshop thematisiert die Konsequenzen, die sich aus der Forderung nach inkrementeller und zeitsynchroner Verarbeitung, sowie aus der inhärenten Unsicherheit bei der Interpretation des Sprachsignals ergeben.

Sprachliche Kommunikation, insbesondere in natürlichen Dialogsituationen, unterliegt einer Reihe von einschneidenden prozeduralen Anforderungen. Solche Anforderungen ergeben sich zum einen aus der strikten Bindung von Sprachperzeption und -produktion an die *zeitliche Dimension des Sprachsignals*. Die prinzipielle Begrenztheit der menschlichen (und maschinellen) Verarbeitungskapazität setzt hierbei zwangsläufig eine hocheffiziente Organisation der Verarbeitungsprozesse voraus, um auch unter dem Einfluß von Streßfaktoren ein Schritthalten der Verarbeitung mit den Erfordernissen der jeweiligen Kommunikationssituation gewährleisten zu können.

Prozedurale Anforderungen resultieren zum anderen aus der erheblichen intra- und interindividuellen *Varianz des Sprachsignals*. Sie stellt eine wesentliche Quelle von Erkennungsunsicherheit dar und hat zur Folge, daß jedwede interpretierende Beschreibung des sprachlichen Inputs nur Hypothesencharakter tragen kann.

Im Bereich der Schriftsprache ist die Zeitdimension auf Anordnungsrelationen zwischen sprachlichen Zeichen reduziert. Erkennungsunsicherheit spielt – zumindest bei der tastaturgebundenen Eingabe – keine entscheidende Rolle. Nur aufgrund dieser Abstraktionen ist es letztendlich auch möglich, eine Verarbeitungsaufgabe vollständig auf der Basis rein deklarativer Spezifikationen zu beschreiben und von einem kombinatorischen und atemporalen Deduktionsmechanismus verarbeiten zu lassen. Wenn

nunmehr in diesem Workshop gerade die prozeduralen Aspekte der zeitlichen Strukturierung und der Verwaltung konkurrierender Hypothesen in den Mittelpunkt gestellt werden, so versteht sich das keineswegs als Absage an die Verwendung deklarativer Repräsentationsformalismen. Vielmehr wird hierbei gerade zu klären sein, auf welche Weise deklarative Spezifikationen in eine geeignete Verarbeitungsarchitektur eingebettet werden können, um auf dieser Grundlage die gewünschten Verarbeitungscharakteristika zu erzielen. Als Ausgangspunkt, Vergleichsmaßstab und Inspirationsquelle bietet sich naturgemäß das menschliche Vorbild an, ist es doch bisher einzig der Mensch der über Sprachverarbeitungskapazitäten der angestrebten Art verfügt.

Eine wesentliche Eigenschaft natürlicher Sprachverarbeitung ist ihre *Inkrementalität.* Teilabschnitte einer sprachlichen Äußerung werden dabei auf den verschiedenen Ebenen zeitlich parallel und praktisch verzögerungsfrei bearbeitet. Insbesondere schließt inkrementelle Verarbeitung ausdrücklich das traditionell dominierende Verarbeitungsmodell aus, das vom Vorliegen vollständiger Eingabedaten schon bei Verarbeitungsbeginn ausgeht. Zum einen ermöglicht erst eine inkrementelle Verarbeitung ein natürliches Dialogverhalten, das sich etwa durch unmittelbare Reaktionen auf die Beiträge der Dialogpartner, sowie die Fähigkeit zur Übernahme der Dialoginitiative auszeichnet. Zum anderen ist sie Voraussetzung für die dynamische Generierung von Diskurserwartungen, dem wohl wirksamsten Mittel zur Suchraumbeschränkung beim Sprachverstehen.

Die Forderung nach inkrementeller Verarbeitung besteht für ein Sprachverarbeitungssystem in seiner Gesamtheit. Inkrementalität kann in ihren Vorzügen nur dann voll zur Wirkung kommen, wenn sie auf allen Ebenen des Verarbeitungsprozesses durchgängig realisiert ist. Nicht zuletzt aus diesem Grunde sind auf dem Workshop Beiträge zu inkrementellen Verfahren in so unterschiedlichen Bereichen wie der Worterkennung, der syntaktischen Analyse und der Generierung vertreten.

Menschliche Sprachverarbeitung verläuft in ihren wesentliche Zügen *zeitsynchron.* Eine solche Forderung auch auf maschinelle Sprachverarbeitungssysteme zu übertragen, scheint auf den ersten Blick naheliegend, ist aber angesichts der kombinatorischen Eigenschaften der üblichen Verarbeitungsalgorithmen durchaus nicht selbstverständlich und schon gar nicht trivial. Die Anpassung der Verarbeitungsprozesse an den aktuell gegebenen zeitlichen Verarbeitungsdruck erfordert ein explizites und zeitsensitives Scheduling für die vorhandenen Verarbeitungskapazitäten. Fragen der Aufmerksamkeitsfokussierung und des systematischen Vergessens von Teilergebnissen werden zwangsläufig eine wichtige Rolle spielen. Grundlage dafür sind Relevanzabschätzungen, die mit der erforderlichen Zuverlässigkeit wiederum nur vor dem Hintergrund von aussagekräftigen Diskurserwartungen getroffen werden können. Zu all diesen Fragestellungen befindet sich die Forschung noch ganz am Anfang.

Aufgrund ihrer Wurzeln in der stochastischen Entscheidungstheorie zählte die Arbeit mit *Bewertungen und Konfidenzen* von Anfang an zu den elementaren Techniken im Bereich der Sprachsignalerkennung. Auch in mehrstufigen Erkennungssystemen hat sich die durchgängige Verwaltung von Bewertungen als effektive Möglichkeit zur Einschränkung des Suchraumes durch Selektion zwischen konkurrierenden Erkennungshypothesen bewährt. Ein wesentliches Ziel der Forschungen zu integrierten Systemen für die Verarbeitung gesprochener Sprache ist es daher, die Techniken der Arbeit mit Präferenzen und Bewertungen auch auf die nichtsequentiellen Strukturierungsmechanismen im Bereich von Syntax und Semantik zu übertragen. Es kann davon ausgegangen werden, das diese Frage gerade für die Behandlung spontan gesprochener Sprache mit ihren typischen Performanzphänomenen von entscheidender Bedeutung sein wird.

Kapitel 1

Integration von Erkennung und Interpretation gesprochener Sprache

Gernot A. Fink, Franz Kummert, Gerhard Sagerer
Universität Bielefeld
Technische Fakultät
Angewandte Informatik
Postfach 100 131
33501 Bielefeld

Zusammenfassung

Die Mehrzahl der heutigen sprachverstehenden Systeme sind aus zwei sehr unterschiedlichen Verarbeitungskomponenten aufgebaut, einer meist statistischen Spracherkennungskomponente und einer in der Regel wissensbasiert arbeitenden Komponente zur Interpretation. Die Kommunikation zwischen diesen Verarbeitungseinheiten ist stark eingeschränkt. In der Regel werden von der Spracherkennung nur Worthypothesenmengen berechnet und an die Interpretationskomponente weitergereicht *ohne weitere Interaktion*.

Wir stellen im folgenden ein Konzept zur Integration von Spracherkennung und Sprachverstehen vor, das die Interaktion dieser Komponenten entscheidend verbessert. Es stellt eine gemeinsame Wissensbasis und eine kompatible Wissensstruktur bereit und leistet eine integrierte Steuerung der Analyseaufgabe.

1.1 Einführung

Obwohl in den letzten Jahren wichtige Fortschritte in der Sprachtechnologie erzielt wurden, wird erst seit relativ kurzer Zeit versucht, integrierte sprachverstehende Systeme zu entwickeln. Die vielen hochentwickelten Spracherkennungssysteme stellen ihre

[1]Die vorliegende Arbeit wurde im Rahmen des Sonderforschungsbereichs 360 "Situierte künstliche Kommunikatoren" von der Deutschen Forschungsgemeinschaft gefördert sowie im Rahmen des Verbundvorhabens Verbmobil vom Bundesministerium für Forschung und Technologie (BMFT) unter dem Förderkennzeichen 01IV102G/7. Die Verantwortung für den Inhalt dieser Arbeit liegt bei den Autoren.

Leistungsfähigkeit in der Regel unabhängig von einer linguistische Weiterverarbeitung unter Beweis. Im Bereich des Sprachverstehens liegt der Schwerpunkt zum großen Teil immer noch auf der Weiterentwicklung natürlichsprachlicher Systeme, die bereits auf einer textuellen Repräsentation der Sprache aufsetzen.

Die Kombination von Spracherkennung und Sprachverstehen ist daher keine leichte Aufgabe, da die Algorithmen zur syntaktischen und semantischen Analyse stark von der Zuverlässigkeit der verarbeiteten Eingabedaten abhängig sind. Jedoch kann kein existierendes — und ebenso kein künftiges — Spracherkennungssystem absolut sichere Erkennungsergebnisse erzeugen.

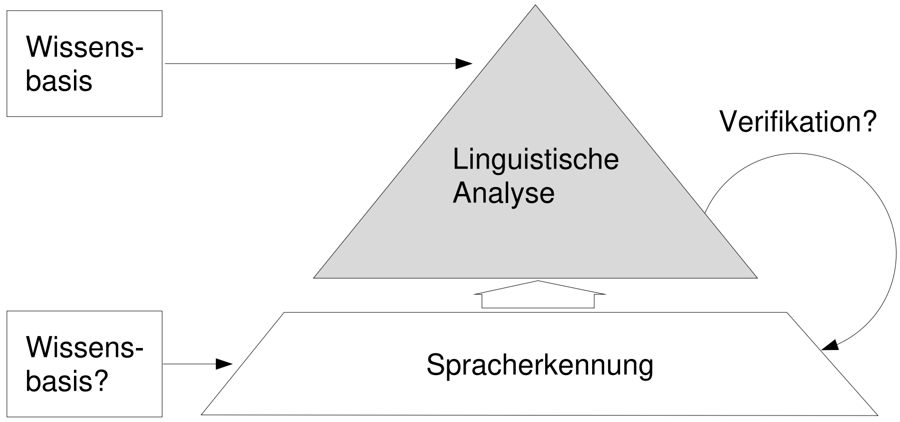


Abbildung 1.1: Architekturschema eines traditionellen sprachverstehenden Systems

Daher sind nur relativ wenige Systeme, die sich robuster Analysestrategien bedienen, in der Lage Erkennung *und* Verstehen gesprochener Sprache zu leisten. Diese Systeme sind in der Regel streng in zwei Verarbeitungseinheiten unterteilt — der statistischen Wortkerkennung und der wissensbasierten Verstehenskomponente. Als Folge dieses Aufbaus, der schematisch in Abbildung 1.1 dargestellt ist, und der unterschiedlichen zum Einsatz kommenden Analyseparadigmen entsteht ein Engpaß an der Schnittstelle der beiden Komponenten. Die am weitetsten verbreitete Kopplungsmethode besteht darin, die Worterkennungskomponente für komplett vorliegendes Sprachmaterial eine Menge von Worthypothesen in Form von Wortketten oder -graphen erzeugen zu lassen. Diese wird dann an die Interpretationskomponente weitergereicht. Eine echte Interaktion findet *nicht* statt, da es sich um eine unidirektionale Schnittstelle handelt.

Einige neuere Ansätze versuchen Erkennung und Verstehen gesprochener Sprache enger zu integrieren. Dabei werden teilweise die folgenden Fragestellungen behandelt, die aus unserer Sicht für den Entwurf eines integrierten sprachverstehenden Systems von Bedeutung sind.

- Sowohl bei der Erkennung als auch bei der Interpretation müssen konsistente Restriktionen für wohlgeformte Wortfolgen verwendet werden.

- Wenn im Erkennungsprozeß komplexe Restriktionen Anwendung finden, sollte die darin zusätzlich vorhandene Information nicht in den erzeugten Ergebnissen verloren gehen. Die Erkennungskomponente eines integrierten Systems sollte also in der Lage sein, Hypothesen zu generieren, die in der Komplexität über einfache Worthypothesen hinausgehen.

- Die Steuerung eines integrierten Systems sollte durch eine integrierte Kontrollstrategie erfolgen.

1.2 Linguistische Analyse

Die linguistische Analyse basiert auf einer Repräsentation linguistischen Wissens als semantisches Netzwerk mit Hilfe der Repräsentationssprache ERNEST [9]. Die Beschreibung linguistischen Wissens für die Domäne *Zugauskunft* ist in die folgenden vier Abstraktionsebenen unterteilt:

- Die *Hypothesenebene* stellt die Schnittstelle zur Worterkennung dar.
- Die *Syntaxebene* enthält Konzepte, die zum einen syntaktische Konstituenten wie Verbalgruppe oder Präpositionalgruppe und zum anderen spezielle Zeitangaben wie Datum und Uhrzeit modellieren. Auf die Erstellung einer kompletten Satzgrammatik wurde verzichtet, da Stellungsregularitäten in gesprochener Sprache praktisch nur *innerhalb* von Konstituenten auftreten.
- Die Modellierung von Bedeutungen auf der *Semantikebene* beruht auf der Tiefenkasustheorie. Sie geht davon aus, daß ein Verb für eine gewisse Bedeutung Leerstellen eröffnet, denen eine funktionale Rolle oder Tiefenkasus wie z.B. `Goal` zugeordnet wird. Dieses Vorgehen läßt sich auch auf Nomina übertragen.
- Die *Pragmatikebene* dient der Beschreibung anwendungsabhängiger Begriffe wie `Fahrplanauskunft`, `Ankunftsort` oder `''mit dem Zug fahren''`. Die Modellierung orientiert sich stark an der Semantikebene. Die dort vorhandenen Konzepte werden hier auf ihre spezifische Bedeutung im Anwendungsbereich eingeschränkt.
- Wissen über den erwarteten Dialogverlauf wird auf der *Dialogebene* repräsentiert. Diese enthält Modelle für Benutzerdialogschritte und Systemäußerungen.

Auf allen eben genannten Beschreibungsebenen können Abfolgerelationen zwischen Bestandteilen komplexerer Konstituenten spezifiziert werden. Syntaktische, semantische und anwendungsabhängige Merkmale werden durch Attribute der Netzwerkkonzepte beschrieben.

Neben der Fähigkeit, Wissen für Mustererkennungszwecke repräsentieren zu können, erlaubt ERNEST außerdem den effizienten Einsatz dieses Wissens in der Analyse. Eine problemunabhängige Kontrollstrategie basierend auf dem A*-Algorithmus ist Teil des Systems. Anwendungsabhängige Erweiterungen dieses Kontrollalgorithmus bestimmen die letztendliche Analysestrategie [10].

1.3 Akustische Analyse

Die akustische Verarbeitung im Rahmen der hier vorgestellten Untersuchungen erfolgt mit Hilfe des ISADORA-Systems [13], das hochflexible Spracherkennungsverfahren auf der Basis von Hidden-Markov-Modellen (HMM) bereitstellt. ISADORA besteht im wesentlichen aus Modulen zur Vorverarbeitung des Sprachsignals sowie zur Modellierung, dem Training und der Dekodierung von HMMs. Der Modellierung kommt dabei besondere Bedeutung zu, da insbesondere die Möglichkeit besteht, komplexe

Konstituentennetzwerke auf der Basis elementarer akustischer Modelle zu definieren. Mit Hilfe einer deklarativen Regelkomponente läßt sich jede reguläre Sprache über dem Alphabet der Elementarmodelle definieren. Jede Konstituente dieser Grammatik entspricht dabei einem komplexen HMM.

In der Erkennungsphase verwendet ISADORA einen durch Strahlsuche gesteuerten Viterbi-Algorithmus auf dem HMM des Analysekonzepts. Das Ergebnis der Dekodierung kann zum einen die bestpassende Konzeptfolge sein. Es lassen sich aber auch Ergebnisse in Form von Wortgraphen oder strukturierten Instanzen erzeugen. Jede elementare Instanz ist dabei durch den Namen des entsprechenden Netzwerkknotens, ihre akustische Bewertung sowie durch ihre zeitliche Zuordnung zum Sprachsignal gekennzeichnet, die sich aus der Viterbi-Suche ergibt.

1.4 Integration

Grundlage für die von uns angestrebte Integration von Spracherkennung und –interpretation ist die Möglichkeit, komplexe sprachliche Einheiten (z.B. linguistische Konstituenten) direkt als strukturiertes Modell der Erkennungskomponente zu repräsentieren und von dieser verarbeiten zu lassen. Da es sich bei diesem Verfahren um eine Kombination von Wissensrepräsentationsverfahren handelt, die auf semantischen Netzen bzw. Hidden-Markov-Modellen basieren, wurde für die formalismusübergreifenden Modelle die Bezeichnung *"Semantische Hidden-Markov Netzwerke"* gewählt [2]. Abbildung 1.2 veranschaulicht, wie mit diesem Verfahren Teile der Wissensbasis gemeinsam für die linguistische und die akustische Analyse genutzt werden können.

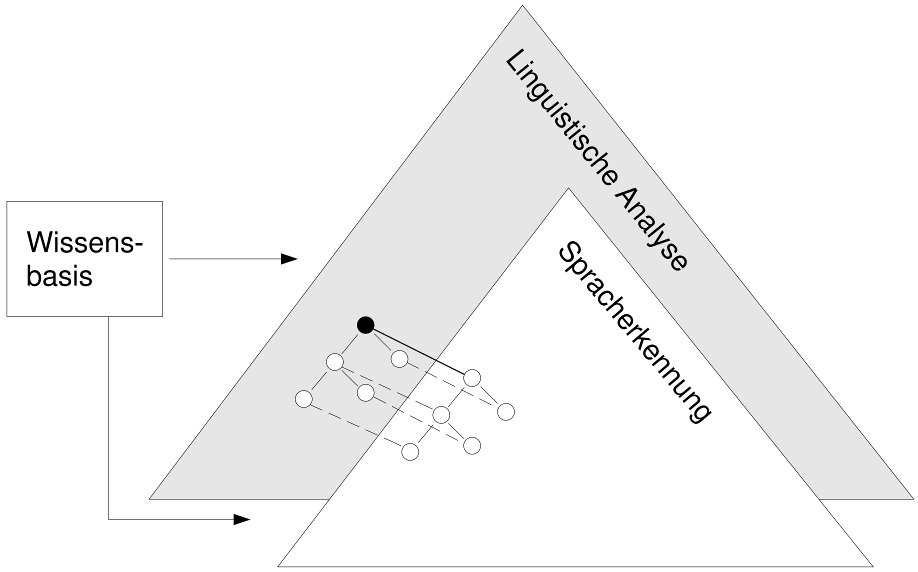


Abbildung 1.2: Schatische Darstellung der integrierten Wissensbaisis

Um die Kompatibilität der in der Erkennung und bei der Analyse eingesetzten linguistischen Restriktionen zu gewährleisten, werden die komplexen Erkennungsmodelle automatisch aus der linguistischen Wissensbasis extrahiert. Im Gegensatz zu ähnlichen Verfahren, bei denen versucht wird, automatisch Sprachmodelle aus deklarativem linguistischem Wissen abzuleiten [12, 14], bleibt bei dem von uns entwickelten Verfahren der strukturelle Aufbau der zu modellierenden Einheiten dabei erhalten [1].

Die so gewonnenen Modelle können wie ein Sprachmodell dazu benutzt werden, den Erkennungsprozeß zu steuern. Darüberhinaus stellen sie jedoch sicher, daß bereits die Erkennungsergebnisse der in der linguistischen Wissensbasis vorliegenden Struktur genügen [3]. Diese können dann *ohne weitere Suche* auf partielle Interpretationen abgebildet werden. Durch die strukturelle Äquivalenz ist sichergestellt, daß die Abbildung immer erfolgreich ist. Allerdings werden dabei eventuell zusätzliche in der linguistischen Wissensbasis enthaltene Restriktionen wirksam, die über Bewertungen die rücktransformierte Interpretation als im Sinne der Analyse unzulässig klassifizieren.

Die eben beschriebenen Verarbeitungsschritte, die dazu dienen, eine für akustisches und linguistisches Wissen gemeinsame Wissensbasis zu etablieren zeigt Abbildung 1.3 schematisch.

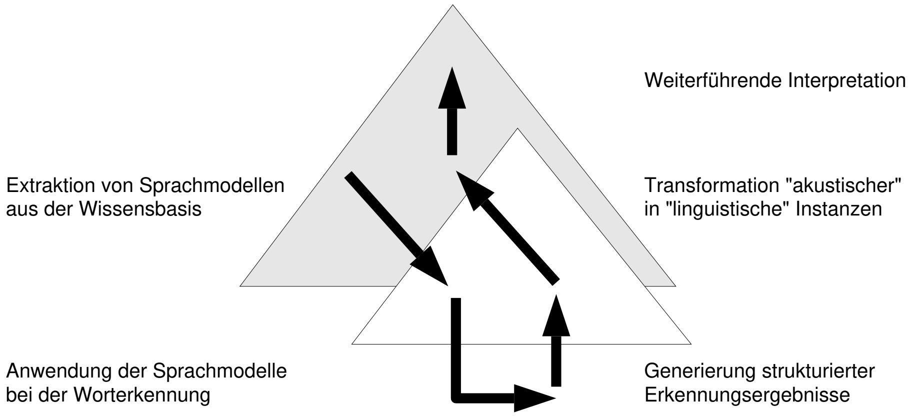


Abbildung 1.3: Verarbeitungszyklus innerhalb der virtuellen integrierten Wissensbasis

Neben der Kompatibilität der eingesetzten Restriktionen ist die Frage der Steuerung von akustischer und linguistischer Analyse von entscheidender Bedeutung für ein integriertes Verarbeitungssystem. Die derzeit verbreitetste Methode zum Entwurf sogenannter *"eng gekoppelter" Systeme* beruht auf einer Kombination eines LR-basierten Parsers für natürliche Sprache und eines Spracherkennungssystems [4, 8, 11]. Einen guten Vergleich verschiedener solcher Kopplungsmethoden findet man in [6]. Auch andere prädiktive Parsingtechniken können eine ähnliche Kopplung erlauben [5]. Allen Verfahren gemeinsam ist es, daß die komplexeste von der akustischen Verarbeitung generierte Hypothese aber immer noch nur *ein* Wort umfassen kann. Außerdem ist durch LR(0)-Grammatiken, wie sie meist zum Einsatz kommen, nur ein — häufig auf die Syntax beschränkter — Teil der linguistischen Verarbeitung abzudecken.

1.5 Robuste inkrementelle Verarbeitung

Für die Interpretation einer Äußerung schlagen wir ein Verfahren vor, das diese von links nach rechts abarbeitet und dabei zwischen Prädiktions- und integrierten Erkennungs- und Interpretationsphasen wechselt. Dadurch erfolgt sowohl die Spracherkennung als auch die linguistische Analyse inkrementell. Beide Phasen verarbeiten im Gegensatz zu den oben angesprochenen Verfahren komplexe Konstituenten.

In einer Prädiktionsphase wird aufgrund des aktuellen Analysezustandes eine Menge von linguistischen Konstituenten bestimmt, die als mögliche Erweiterungen der bislang berechneten Interpretation in Frage kommen. Diese werden an die Erkennungskomponente übergeben, die auf der Basis der korrespondierenden strukturierten Sprachmodelle einen weiteren Teil der Äußerung abarbeitet. Die strukturierten Erkennungsergebnisse werden auf partielle linguistische Analyseergebnisse abgebildet und in die bereits vorliegende Interpretation integriert.

Solange die Äußerung nicht vollständig abgearbeitet ist, wird ein weiterer Prädiktionsschritt angestoßen. Andernfalls werden die noch unverbundenen Teile der aktuellen Interpretation zu einer Gesamtinterpretation zusammengefaßt. Ist dieses Resultat zulässig, so ist das gewünschte Analyseergebnis gefunden. Ansonsten wird ein anderer Suchpfad mit einer alternativen Interpretation weiterverfolgt.

Beispielhaft soll der in Abbildung 1.4 gezeigte vereinfachte Ausschnitt einer integrierten Analyse betrachtet werden. Der Anfangsteil der Äußerung sei dabei bereits erfolgreich verarbeitet worden. Dabei entstand die Interpretation der Hypothesenfolge *"Ich möchte"*. Diese Teilinterpretation soll der Einfachheit halber nach rechts nur um die Konstituenten *Ankunftsort* (*P_Ankunftsort*), *Uhrzeit* (*Z_Uhr*) oder *Tagesangabe* (*Z_Tag*) erweitert werden können. Sie bilden zusammengefaßt die Prädiktionsmenge.

Nun wird ein kombiniertes Sprachmodell aus den zu diesen Konstituenten gehörigen automatisch erzeugten Sprachmodellen gebildet und für die Erkennung in dem Signalbereich eingesetzt, der an die Hypothese *"möchte"* anschließt. Die optimale Lösung für die Konstituente *Tagesangabe* sei die komplexe Hypothese *"am Montag"*. Für *Abfahrtsort* wird aufgrund des bereits auf akustischer Seite einsetzenden Prunings keine Lösung berechnet und daher der zu dieser Prädiktion gehörige Suchpfad von der linguistischen Analyse eliminiert.

Aus den komplexen Erkennungsergebnissen für die übrigen beiden Konstituenten werden linguistische Teilinterpretationen aufgebaut. Durch die Kombination mit der bereits vorhandenen Interpretation entstehen neue Analysezustände. Der bestbewertete sei derjenige, der durch Hinzunahme der *Tagesangabe* entstand. Von ihm aus wird eine neue Prädiktionsmenge berechnet, die jetzt nur noch aus den Konstituenten *Ankunftsort* und *Uhrzeit* besteht. Diese neuen Prädiktionen werden wieder an die Erkennungskomponente übergeben und das integrierte Analyseverfahren wird analog zum eben gezeigten weitergeführt.

Erst in letzter Zeit ist die Spracherkennungsforschung auf das Problem eingegangen, daß nicht notwendigerweise alle vom Benutzer geäußerten Wörter der Erkennungskomponente bekannt sein müssen. Es wurden erste Verfahren zur Detektion sogenannter "unbekannter Wörter" entwickelt.

In einem Erkennungssystem, das auf Konstituentenmodellen beruht, stellt sich dieses Problem in erweiterter Form. Zum einen können unbekannte Wörter verwendet werden, die gebildeten Konstituenten aber in ihrer Struktur bekannt sein. Zum anderen können aber auch Äußerungsstrukturen auftreten, die durch die linguistische Wissensbasis *nicht* abgedeckt werden.

Daher ist auch ein Modell für unbekannte Wörter bzw. Konstituenten Teil jeder Prädiktionsmenge [7]. Dieses wird in der Erkennungsphase wie auch bei der Interpretation genauso behandelt, wie "normale" Konstituentenmodelle. Dadurch ist das System in der Lage, Teile der Äußerung, die nicht durch seine sprachliche Kompetenz abgedeckt werden, als unbekannt zu klassifizieren und die Interpretation am Ende dieses unbekannten Bereichs wieder aufzunehmen.

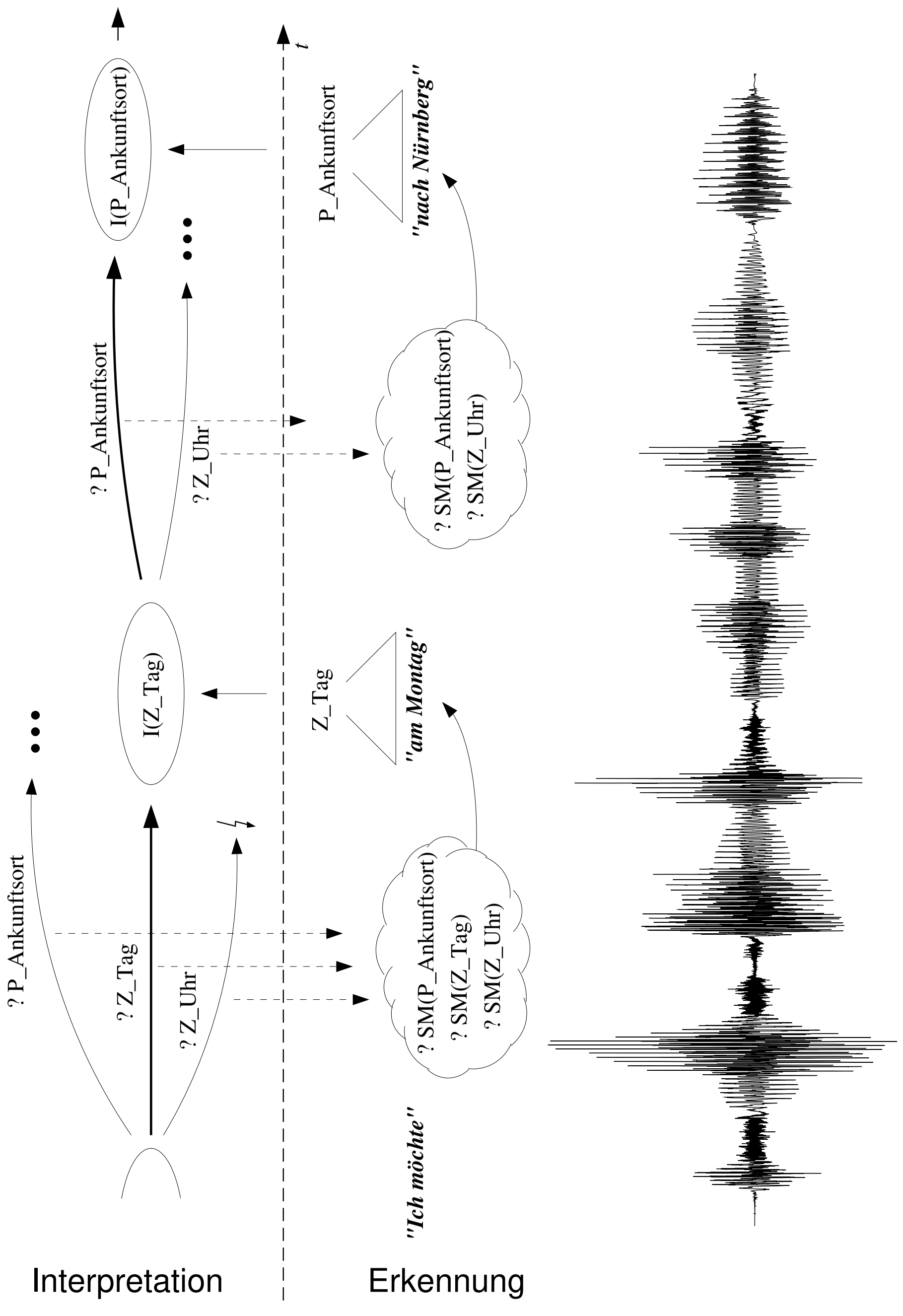


Abbildung 1.4: Interaktion der Verarbeitungskomponenten bei dem vorgeschlagenen inkrementellen Analyseverfahren für gesprochene Sprache

	Interpretationen
Korrekt	79 %
Partiell	1 %
Fehlerhaft	13 %
keine Lösung	7 %

Tabelle 1.1: Evaluierungsergebnisse: Prozentsatz korrekt, teilweise oder fehlerhaft interpretierter Äußerungen sowie Prozentsatz fehlgeschlagener Interpretationsversuche

1.6 Erste Ergebnisse

Zum Einsatz kam ein sprecherunabhängiges Spracherkennungssystem auf der Grundlage von ISADORA. Die Trainingsmenge bestand aus Daten von 74 Sprechern, die je 100 Äußerungen aus der Domäne Zugauskunft beigesteuert hatten. Zum Test wurden 21 Äußerungen von 4 Sprechern (3m, 1w) verwendet, die kein Trainingsmaterial geliefert hatten. Die Testäußerungen konnten im Prinzip durch die definierten Sprachmodelle vollständig analysiert werden. In dieser Konfiguration wurde das Modell für unbekannte Konstituenten nur als Referenz für die akustischen Bewertungen herangezogen. Obwohl das System in der Lage ist, gesprochene Dialoge zu analysieren, wurden zum Zwecke der Evaluierung nur Einzeläußerungen betrachtet.

Tabelle 1.1 zeigt, daß die Mehrzahl der Äußerungen korrekt und *vollständig* interpretiert werden konnten. Es konnte also alle für den Zweck der Zugauskunft relevante Information extrahiert werden. Partielle Interpretationen entstehen dann, wenn die Analyse aufgrund von Eigenschaften des Suchprozesses vorzeitig abgebrochen wird, die das Finden einer besseren Interpretation nicht wahrscheinlich erscheinen lassen. Bei fehlerhaften Ergebnissen wurde eine oder mehrere der linguistischen Konstituenten entweder nicht erkannt oder falsch interpretiert.

Die Hauptproblematik des Verfahrens ist seine große Sensitivität bezüglich der akustischen Bewertungen. Diese bieten nicht immer eine zuverlässige Grundlage für eine Entscheidung über das Vorhandensein einer linguistischen Konstituente im Sprachsignal. Noch schwieriger ist die Situation bei der Hinzunahme potentiell unbekannter Konstituenten. Dies erfordert jedoch noch genauere Untersuchungen.

1.7 Zusammenfassung

Es wurde ein integriertes inkrementelles Verfahren zur Spracherkennung und –interpretation vorgestellt. Eine integrierte akustische und linguistische Wissensbasis wird mit Hilfe von *"Semantischen Hidden-Markov Netzwerken"* etabliert. Dabei wird linguistisches Wissen in Form von automatisch erzeugten Modellen direkt im Erkennungsprozeß angewandt, um strukturierte Erkennungsergebnisse zu erzeugen. Diese können sofort die aktuelle linguistische Interpretation erweitern. Daher ist sichergestellt, daß die von der linguistischen und akustischen Analyse eingesetzten Restriktionen immer konsistent sind. Um den Irregularitäten gesprochener Sprache Rechnung zu tragen, ist ein Modell für unbekannte Konstituenten Teil jeder Erkennungsaufgabe.

Auf der Grundlage des anwendungsunabhängigen A*-basierten Kontrollalgorithmus, wie er von ERNEST bereitgestellt wird, wurde eine inkrementelle Analysestrategie vorgeschlagen, die zwischen Prädiktions- und strukturbildenden Erkennungsphasen alterniert. Die Äußerung wird dabei von links nach rechts abgearbeitet. Prädiktionen, die Mengen von linguistischen Konstituenten sind, können während der Interpretation dynamisch in Abhängigkeit vom Dialogkontext und dem aktuellen Stand der Analyse berechnet werden.

Unsere weiteren Arbeiten werden sich vor allem darauf konzentrieren, die Detektion unbekannter Wörter und Konstituenten zuverlässiger zu gestalten, um die Robustheit des integrierten Analysesystems zu erhöhen. Dafür werden sowohl andere Formen der akustischen Bewertung wie auch andere robuste Strategien der linguistischen Weiterverarbeitung untersucht. Außerdem wird die vorgestellte integrierte inkrementelle Analysestrategie auch auf die “Verbmobil”-Domäne übertragen.

Literaturverzeichnis

[1] G. A. Fink, F. Kummert, and G. Sagerer. Automatic Extraction of Language Models from a Linguistic Knowledge Base. In J. Vandewalle, R. Boite, M. Moonen, and A. Oosterlinck, editors, *Signal Processing VI: Theories and Applications*, volume 1, pages 547–550. Elsevier Science Publishers, Amsterdam, 1992.

[2] G. A. Fink, F. Kummert, G. Sagerer, E.G. Schukat-Talamazzini, and H. Niemann. Semantic Hidden Markov Networks. In *Proc. Int. Conf. on Spoken Language Processing*, volume 2, pages 919–922, Banff, Canada, 1992.

[3] Gernot A. Fink, Franz Kummert, and Gerhard Sagerer. Speech Recognition using Semantic Hidden Markov Models. In *Proc. European Conf. on Speech Technology*, pages 1571–1574, Berlin, 1993.

[4] David Goddeau. Using Probabilistic Shift-Reduce Parsing in Speech Recognition Systems. In *International Conference on Spoken Language Processing*, pages 321–324, Banff, Canada, 1992.

[5] David Goodine, Stephanie Seneff, Lynette Hirschmann, and Michael Philips. Full Integration of Speech and Language Understanding in the MIT Spoken Language System. In *Proc. European Conf. on Speech Technology*, pages 845–848, 1991.

[6] Andreas Hauenstein and Hans Weber. An Investigation of Tightly Coupled Time Synchronous Speech Language Interfaces Using a Unification Grammar. In Paul McKevitt, editor, *AAAI-94 Workshop Program: Integration of Natural Language and Speech Processing*, pages 42–49, Seattle, Washington, 1994.

[7] A. Jusek, H. Rautenstrauch, G. A. Fink, F. Kummert, G. Sagerer, J. Carson-Berndsen, and D. Gibbon. Detektion unbekannter Wörter mit Hilfe phonotaktischer Modelle. In W.G. Kropatsch and H. Bischof, editors, *Mustererkennung 94, 16. DAGM-Symposium und 18. Workshop der ÖAGM Wien*, pages 238–245. 1994.

[8] Kenji Kita and Wayne H. Ward. Incorporating LR Parsing into SPHINX. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing*, pages 269–272, 1991.

[9] F. Kummert, H. Niemann, R. Prechtel, and G. Sagerer. Control and Explanation in a Signal Understanding Environment. *Signal Processing, special issue on 'Intelligent Systems for Signal and Image Understanding'*, 32:111–145, 1993.

[10] Franz Kummert. *Flexible Steuerung eines sprachverstehenden Systems mit homogener Wissensbasis*, volume 12 of *Dissertationen zur Künstlichen Intelligenz*. Infix, Sankt Augustin, 1992.

[11] Hy Murveit and Robert Moore. Integrating Natural Language Constraints into HMM-based Speech Recognition. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing*, pages 573–576, 1990.

[12] Fernando Peireira. Finite-State Approximations of Grammars. In *Speech and Natural Language Workshop*, pages 20–25, Hidden Valley, Pennsylvania, 1990. Morgan Kaufmann.

[13] E.G. Schukat-Talamazzini and H. Niemann. Das ISADORA-System — ein akustisch-phonetisches Netzwerk zur automatischen Spracherkennung. In *Proc. 13. DAGM-Symposium*, pages 251–258. Springer, Berlin, 1991.

[14] Victor Zue, James Glass, David Goodine, Hong Leung, Michael Phillips, Joseph Polifroni, and Stephanie Seneff. Integration of Speech Recognition and Natural Language Processing in the MIT VOYAGER System. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing*, pages 713–716, 1991.

Kapitel 2

Stochastisches Parsing mit Kopfgrammatiken

Hans Ulrich Block
Siemens AG
ZFE T SN 5
Otto Hahn- Ring 6
D-87130 München 83
Tel.: 089 / 636-44537, FAX: 089 / 636-49802
`block@ztivax.zfe.siemens.de`

Zusammenfassung

In dem Beitrag werden "Kopfgrammatiken" als eine sehr restringierte Form von Unifikationsgrammatiken vorgestellt, und es wird ein effizientes Parsing-Verfahren zur Berechnung der n wahrscheinlichsten Strukturen beschrieben. Ein besonderes Problem stellt hierbei die Ermittlung der Köpfe von leeren Produktionen dar.

2.1 Beschreibung des Verfahrens

Stochastische Grammatiken stellen eine interessante Alternative zu herkömmlichen unifikationsbasierten Ansätzen dar, da sie 1. einfacher zu entwickeln sind, 2. einfacher in robuste Analysemethoden integriert werden können, 3. für die Auflösung von Ambiguitäten besser geeignet scheinen und 4. durch Training leichter auf eine spezifische Anwendung zugeschnitten werden können. Allerdings hat sich gezeigt, daß mit einfachen stochastischen kontextfreien Grammatiken nur unzureichende Ergebnisse zu erzielen sind, da viele die Struktur eines Satzes bestimmende Faktoren lexikalischer Natur sind. Es ist daher wünschenswert, lexikalische Information in der stochastischen Analyse zu berücksichtigen.

Das hier vorgestellte Verarbeitungsmodell basiert auf folgenden Annahmen: Im Lexikon sind den (flektierten) Wörtern syntaktische Kategorien zugeordnet. Die Kategorien enthalten die Grundform oder die syntaktisch/semantische Sorte des Wortes

[1] Die vorliegende Arbeit wurde im Rahmen des Verbundvorhabens Verbmobil vom Bundesministerium für Forschung und Technologie (BMFT) unter dem Förderkennzeichen 01IV102AO unterstützt. Die Verantwortung für den Inhalt dieser Arbeit liegt bei dem Autor.

als Argument,
z.B. `lexikon(mannes,n(mann))`, bzw. `lexikon(mannes, n(human))`.

In der Grammatik erbt die linke Seite einer Grammatikregel die Grundform/Sorte ihres Kopfes, z.B. `np(Kopf) ---> det(_) n(Kopf)` oder eine Menge von Merkmalen der rechten Seite, z.B. `pp(P,N) ---> p(P) np(N)`. Der Informationsfluß ist stets bottom up. Zur Behandlung diskontinuierlicher Abhängigkeiten dienen Spuren, d.h. leere Produktionen, die ihre Merkmale von ihren Antezedenten erben. Der Satz *Karl sieht Peter* hat z.B. die Struktur

$$[_{s2(sehen)}[_{np(karl)}Karl]_i[_{s1(sehen)}[_{v(sehen)}sieht]_j \\ [_{s(sehen)}np(karl)_i[_{vp(sehen)}[_{np(peter)}Peter]v(sehen)_j]]]].$$

Die möglichen Antezedenten einer Spur werden durch Antezedenzregeln in Form von endlichen Automaten über die Dominanzrelation der Spur und des Antezedens definiert, für o.g. Beispiel z.B. `np->s->s1->s2 und np->s2`. Beim Training werden zwei Wahrscheinlichkeiten geschätzt, zum einen die Wahrscheinlichkeiten des kontextfreien Regelteils, z.B. `np ---> det n`, zum anderen die Wahrscheinlichkeiten der Kopfregeln mit jeweils gebundenen Variablen,
z.B. `vp(lesen) ---> np(buch) v(lesen)`.
Die Analyse mit Kopfgrammatiken erfolgt in den Schritten

1. kontextfreie Analyse,
2. approximative Instantiierung der Kopfmerkmale,
3. Restkostenschätzung und
4. A*-Suche mit definitiver Instantiierung

Zunächst wird basierend auf dem kontextfreien Teil der Grammatik mit einem bottom up parser ein gepackter Analysewald erzeugt. Wir verwenden derzeit einen erweitertern CYK-Parser, prinzipiell ist jedoch jeder chart-basierte Parser einsetzbar. Daraufhin werden die Kopfmerkmale für alle Knoten des Waldes außer den Spuren ermittelt. Im nächsten Schritt werden die Antezendenzregeln auf den Analysewald angewandt und die Kopfmerkmale der Spuren ermittelt. Durch diese beiden Schritte entsteht ein Analysewald mit instantiierten Köpfen. Hierbei wird in Kauf genommen, daß insbesondere bei der Bindung von Spuren die Instantiierung übergeneriert, d.h. es ist möglich, daß eine Kopfvariable mit einer Menge von möglichen Werten belegt ist, die je nach Ausfaltung des gepackten Waldes in der folgenden A*-Suche vereindeutigt wird. Im folgenden Schritt werden mit einem Viterbi-artigen Algorithmus die jeweils größten Wahrscheinlichkeiten für jeden Teilbaum des Analysewalds ermittelt. In die Berechnung der Wahrscheinlichkeiten gehen die Wahrscheinlichkeiten der kontextfreien Regeln und der instantiierten Kopfregeln gewichtet ein. Im Falle einer nicht eindeutigen Variablenbelegung wird die Wahrscheinlichkeit nach dem jeweils wahrscheinlichsten Merkmal berechnet. Das Ergebnis der Berechnung ist ein gewichteter gepackter Analysewald. Die ermittelten Teilbaumwahrscheinlichkeiten gehen als Restkosten in die folgende A*-Suche ein. Bei der Suche werden mit einem herkömmlichen A*-Suchverfahren die n wahrscheinlichsten Linksableitungen des Waldes ermittelt. Bei der Expansion eines Knotens kommen die Kopfregeln ein zweites Mal zur Anwendung, um die mehrfachen Variablenbelegungen abhängig von der aktuellen Ableitung zu vereindeutigen.

2.2 Ergebnisse

Zum Training der Grammatik und zum Testen des Ansatzes wurde eine treebank für 2400 Verbmobil-Äußerungen erstellt. Die Äußerungen sind mit vollständigen Strukturen einer "naiven" Grammatik versehen.

Die restringierte Unifikationsgrammatik wurde an einem Trainingskorpus von 1871 Äußerungen trainiert und an 400 Äußerungen getestet. Trainings- und Testkorpus sind disjunkt. Im folgenden sind die wichtigsten Kennzahlen der Korpora zusammengefaßt.

Trainingskorpus	1871
Testkorpus	400
durchschnittliche Anzahl von Wörtern/Satz	7.1
durchschnittliche Anzahl von Lesarten	1.6299e+20
maximale Anzahl von Lesarten	6.4874e+22
minimale Anzahl von Lesarten	1.0

Hierbei fällt die ungewöhnlich hohe durchschnittliche und maximale Anzahl der Lesarten pro Äußerung auf. Sie resultiert aus einer relativ hohen lexikalischen Ambiguität der Wörter, aus der Tatsache, daß die Grammatik keine harten Restriktionen über morphologische Kongruenz und Subkategorisierung enthält, und letztlich aus der für die Verbmobil-Domäne typischen spontansprachlichen Äußerungen, deren Beschreibung eine sehr permissive Grammatik verlangt.

Zur Evaluierung des Verfahrens wurde die Strukturkonsistenz und die Strukturgleichheit bei den eins bis fünf wahrscheinlichsten Analysen ermittelt.

Die folgende Tabelle gibt an, wie oft die in der Test-treebank zugewiesene Struktur mit einer der N wahrscheinlichsten Analysen des stochastischen Parsers konsistent ist.

N	konsistent (absolut)	konsistent (%)	durchschnittliche Position
1	294	73.5	1.00
2	327	81.8	1.10
3	343	85.8	1.18
4	347	86.8	1.22
5	361	90.3	1.37

Die folgende Tabelle gibt an, wie oft die in der Test-treebank zugewiesene Struktur mit einer der N wahrscheinlichsten Analysen des stochastischen Parsers identisch ist.

N	gleich (absolut)	gleich (%)	durchschnittliche Position
1	234	58.5	1.00
2	279	69.8	1.16
3	297	74.3	1.27
4	303	75.8	1.32
5	316	79.0	1.47

In einem nächsten Schritt soll die Grammatik in einem *bootstrapping*-Verfahren an einem größeren, nicht mit Strukturen annotierten Korpus trainiert werden. Hierbei sollen, ausgehend von der an der Baumbank trainierten Grammatik, für jeden Satz die jeweils n wahrscheinlichtsten Analysen ermittelt und für das Training verwendet werden. Da die Strukturerkennungsrate bei $N = 5$ bereits bei ca. 80 % liegt, besteht die Hoffnung, daß relativ viele korrekt strukturierte Phrasen in den ersten n Lesarten konstant bleiben und somit beim Training forciert werden.

Kapitel 3

Time Synchronous Chart Parsing of Speech Integrating Unification Grammars with Statistics

Hans Weber
IMMD 8
University Erlangen-Nurnberg
Germany
email: weber@faui80.informatik.uni-erlangen.de

Abstract

We present an active chart parser which parses left connected wordgraphs in a strictly time synchronous way. The parser performs a beam search on the possible paths through the word graph and on the possible derivations of the unification grammar simultaneously. A metric is given to assign scores to edges, taking into account the whole left context thereby combining acoustic probabilities, n-gram probabilities and unification grammar probabilities. A specialized model for the derivation of typed unification grammars is introduced. Different ways of coupling the parser with an LR beam decoder in an online time synchronous fashion are defined and several experimental results are presented. Two top down and one bottom up method are investigated. In bottom up mode, the decoder sends word hypotheses as they are found from left to right, while the parser keeps step. In verify mode, the decoder is always a frame ahead, while the parser verifies received hypotheses, providing language information to the decoder. In predict mode, the parser is a frame ahead, sending possible successor information to the decoder. The latter uses this information to restrict its search space. Finally a method to maximally reduce multiple paths in a left connected word graph produced by a beam decoder ist presented, which can be used in all of the three strategies.

[1] This work was funded by the German Federal Ministry for Research and Technology (BMFT) in the framework of the Verbmobil Project under Grant BMFT 01 IV 101 H / 9. The responsibility for the contents of this study lies with the author.

3.1 Introduction

This article gives an overview of ongoing research in the area of time synchronous processing in speech understanding. Here we are mainly concerned with the design of a chart parser for that issue. The parser which we call a *Left-Right Incremental Active Chart Parser* (LR-ACP) has the following features specific to the task.

- All word hypotheses are processed simultaneously in one chart from left to right.
- The parser uses a typed unification grammar the derivations of which are subject to a probabilistic model.
- Agenda items and edges are supplied with a combined normalized score of acoustic, bi-gram and grammar model probabilities.
- The parser performs a beam search implemented on the agenda, thereby constraining forward and backward search.
- Extensions of the algorithm for time synchronous interaction with a one pass beam decoder are implemented.
- Receiving the decoder's word ending hypotheses frame by frame, a mapping of hypotheses belonging to the same word copy is performed during parsing.

The ideas which led to this work are based, besides others, on works like Shieber 85 [19], Görz 88 [9], Ney 93 [16], Päseler 88 [17], Fujisaki et al. 91 [8]. For a more detailled description see Weber 94 [20]. Some of the work concerning the coupling with the beam decoder is a joint work with Andreas Hauenstein[1] and has already been published in Hauenstein & Weber 94 [10, 11].

3.2 LR-ACP with Beamsearch on the Agenda

The use of active chart parsers and extensions of these in parsing speech has tradition. Our LR-ACP can be seen as a consequent next step to the work of Päseler 88 [17].

3.2.1 Basic Operations

The basic data structures of an active chart parser can be given as follows:

Definition 3.2.1 *Basic data structures of an ACP*

Vertices: *A chart is a directed acyclic graph, whose vertices are totally ordered. Vertices correspond to word boundary hypotheses.*

Edges: *An edge consists of a pair of vertices, a dotted rule, a (record of) score(s) and a couple of book-keeping information as there a are the covered string of word hypotheses, pointers to daughter edges and also pointers to the left and right hand sides of rule instances, aso.*

Agendas: *For every vertex i there exists an* $agenda_i$, *keeping triples of actives, inactives and scores, such that the end vertex of the inactive of a triple is vertex i. Agendas are accessed best first.*

[1] University Hamburg, Natural Language Dept.

We use the standard active chart parser operations, adapted to unification grammars to drive the LR-ACP. For an edge A, we write selectors for the components of A, using A.from and A.to to refer to the begin and end vertices of A, A.mother to the left hand side of A's rule, A.next to the category to the immediate right of the dot, aso.[2] We write $\tilde{\cup}$ for the relation of being unifiable, and $\cup$ for the unification operation.

Definition 3.2.2 *Basic operations of an ACP*

SEEK-UP: *If an inactive A is inserted, for every rule $\alpha \longrightarrow \beta$, such that A.mother $\tilde{\cup}$ first(β), insert B, with B.rule := $\alpha \longrightarrow .\beta$, B.from := A.from, B.to := A.from, if B does not already exist.*

SEEK-DOWN: *If an active A is inserted, for every rule $\alpha \longrightarrow \beta$, such that A.next $\tilde{\cup}$ α, insert B, with B.rule := $\alpha \longrightarrow .\beta$, B.from := A.from, B.to := A.from, if B does not already exist.*

FUNDAMENTAL RULE: *For a pair of active A and inactive B, if A.next $\tilde{\cup}$ B.mother, insert C with C.from := A.from, C.to := B.to, C.rule := copy(A.rule), C.next := A.next. Perform the following actions: $\cup$ copy(B.mother),C.next. Shift C.next one position.*

PUSH-EDGES: *When edge A is inserted into the chart, then if A is active, push all pairs (A,B) to agenda, such that B is inactive and A.to = B.from. Else, push all pairs (B,A) to the agenda, such that B is active and B.to = A.from. Insertion is binary according to the score of (A,B) and (B,A).*[3]

The LR-ACP can be driven top down using the SEEK DOWN operation as well as bottom up with the SEEK UP operation, as usual active chart parsers can.

3.2.2 Efficiency Matters

Efficiency of the basic operations is really a challenge, since the search space given by a couple of thousand word hypotheses is huge and in no way comparable to the easy task of parsing deterministic input.

In the specification above, the SEEK DOWN (and SEEK UP) operations cost a lot of unifications, although no top down restriction is propagated during the rule insertion in order to keep the rule instances finite. In order to achieve efficiency with the unification grammar, we preprocessed and hashed all SEEK-DOWN operations besides the redundancy check.

By testing for unifiability only, but inserting the original rule objects, we can keep the instances of empty active edges finite.

In the preprocessing step, we take every daughter of every rule and compute the series of successor rules possibly being inserted by a SEEK-DOWN operation. In this preprocessing, we use only the type skeleton of the grammar and (nondestructive) type unification in a first pass. A local reduncancy check ends up recursion.

The series produced is filtered afterwards, using all of the features and propagating them through the series.

All rules and daughters have integer codes assigned, so the table lookup as well as the redundancy check in the chart become cheap.

[2] We hope to meet the intuitions of the reader, since we do not explain all notational details.

[3] So the agendas are always kept sorted.

Our treatment of rules can be seen as a variant of Shieber 85's *Restriction* mechanism [19], based on types.

As a result of this technique, there are no more unifications involved in predictor or scanner operations in Earley's terms. Only during the completer step unifications are actually carried out.

The same technique cannot be used when parsing bottom up. The trigger for a SEEK UP operation are inactive edges, which may have infinitely many extensions of feature structures. Nevertheless, the number of types the feature structure (FS) can have is finite. So we again use the type skeleton of the grammar to do a preprocessing. Anyway, we leave out the filtering step in the preprocessing of the SEEK UP operation, accepting a slight overgeneration of rules to be inserted. These additional edges will fail in the completer operation, consuming a part of those full unifications which would have to be done without a preprocessing anyway.[4]

3.2.3 Initial Local Agendas and Beams

To achieve an incremental time synchronous behaviour we propose the following control loop, where BEAMWIDTH must be given in advance.

1. Create Vertex V_0. Insert active start edge. Set T := 1.
2. Create V_T with agenda$_T$. Read all word hypotheses W which end at T, and insert edges for W into the chart.
3. Set BEAMVALUE := score(top(agenda$_T$)) - BEAMWIDTH
4. Apply fundamental rule to pop(agenda$_T$), until score(top(agenda$_T$)) $\leq$ BEAMVALUE.
5. Save agenda$_T$.
6. Increment T. Goto 2.

This pseudo code implements a beam search directly on the agenda. There are a couple of reasons for this strategy.

First of all, an agenda$_T$ in a cycle T, right after the insertion of edges due to the reading of new hypotheses consists only of entries involving pairs of actives and word hypotheses. We call the agendas in this state of processing *Initial Agendas*. An *Initial Agenda* encodes the possible future completer operations in form of their initial pair of active edge and word hypothesis.

Since the completer step is the expensive operation in Earley's Algorithm anyway and especially in our handling of unification operations, we prune on agenda items.

Further on, since we use acoustic, n-gram and grammar scores in combination, we need a representation where global scores can be compared or at least, where good estimations of global scores of paths are admissible to apply pruning.

By virtue of the INSIDE and OUTSIDE scoring of edges described in section 3.3 below, we can assign an optimistic estimate of a path's global score to an active edge not yet covering that path. As a consequence, the maximum score of the initial agenda is the maximum of all possible scores arising from successive completer actions.

In step 5 of the algorithm above, we save those entries of a local agenda which fell below the beam. Since we use a combination of symbolic and probabilistic restriction,

[4]The preprocessing seems to favorize a top down parsing schema.

we can never guarantee that one of the paths inside the beam will result in a valid analysis of some spanning sequence of word hypotheses. In order to make the system robust, a second search phase can be initiated, parsing the global agenda of pruned entries when the time synchronous search failed. When there is a spanning analysis, it will be discovered then.

3.3 Metrics

This section is concerned with the combination of probabilities coming from different models and the partition of combined scores into an INSIDE score and an OUTSIDE score.[5]

3.3.1 Combining Scores

Different models used in one system tend to have different numbers of parameters and to be trained on different data. So, taken seriously, they cannot be combined.

In practice we overcome this problem by adjusting the magnitude of the scores involved. This is done by introducing weights relating of the scores coming from different models. We use log probabilities, since the overall search is Viterbi like, maximizing rather than summing up.

$$\text{CS(W,D)} = AS(W) + \gamma NGS(W) + \delta GS(D) \tag{3.1}$$

Acoustic score (AS), bi-gram score (NGS) and grammar score (GS) are just added, γ and δ regulating the relative weights. W stands for an utterance and D for a possibly partial grammatical derivation of it.

The single scores themselves are normalized by the number of operations in order to be able to compare word hypotheses of different length, utterances made up of a different number of words and trees made up of different amounts of grammatical operations.[6]

$$AS(W) = \frac{logP(W|s_{i,j}, \text{hmm})}{j-i} \tag{3.2}$$

$$NGS(W) = \frac{logP(W|\text{bi-gram})}{\text{words}(W)-1} \tag{3.3}$$

$$GS(D) = \frac{logP(D|\text{gm})}{\text{rules}(D)+\text{typeshifts}(D)} \tag{3.4}$$

The acoustic score for an utterance W is given by the probability assigned by the acoustic model to the signal from frame i to frame j, the portion spanned by W, divided by the number of frames spanned. Analogously we define the bi-gram score and the grammar score, the latter being normalized by the number of rule applications and type shifts, the items of observation of the grammar model *gm* (which will become more transparent in section 3.4).

This is not the only possible way to combine scores. Another method, first summing weighted probabilities, and normalizing them afterwards by the sum of all operations

[5] Although these two scores correspond more or less to Päseler's IS and AS ([17]), we take the names from Baker's [2] Inside-Outside-Algorithm, which we think was the first idea in that direction.

[6] We write X(Y) for the number of operations of type X used to produce Y.

did not show improvements. The behavior of the system is more sensitive to the choice of values for the weights. It turned out that it was really difficult to find good settings by hand. While it was comparably easy to tune only one weight[7], using only acoustic and n-gram scores in the system, things get more complex when additional models are involved. Although we did not yet test it, we feel that the weights should be subject to an optimizing hill climbing procedure.

For the counters for operations and the absolute log probabilities we write AO, NGO, GO, AP, NGP and GP respectively. Edges in fact carry a record with fields for these single values, since the log probabilities and operations are kept seperate for edges. This makes it easier to compute new values for new edges. Combined scores are only computed for comparisons which happen to be done only on agenda items.

3.3.2 Inside and Outside Scores

Similar to Päseler 88 [17], we give two different scores to edges. The first one is the INSIDE score of an edge — the score coming from the portion spanned by that edge. Roughly speaking the OUTSIDE score is the cumulated score of those edges in a left context which were leading to the introduction of a certain edge.

In top down mode the OUTSIDE score of an edge i is the best score of some edges j to m, where j to m lead to the introduction of edge i plus the INSIDE score of i.[8] Generally the edges j to m leading to an edge i are spanning the portion from frame 0 to the beginning of i.

In bottom up mode, where empty actives are introduced on the basis of inactive edges with the SEEK-UP operation, INSIDE and OUTSIDE score of an edge fall together. There is no left context for an edge to be determined from which we could say it led to the introduction of that edge.[9] So the following is on the building of the OUTSIDE score in the case of top down parsing.

We define the INSIDE score as well as the OUTSIDE score for the start edge directly below, without referring to the single components due to space limitations.

Definition 3.3.1 *INSIDE score for an edge E, from i to j, spanning string W with analysis D_W*

$$\mathrm{IS}(E_{i,j,W,D_W}) = \begin{cases} CS(W, D_W) & i < j \\ \delta GS(E.rule) & i = j \end{cases} \tag{3.5}$$

Describing the OUTSIDE scores for the initial start edge is easy. The start edge is initialized with a zero value for all parts of the OUTSIDE score. New empty actives inherit the acoustic and n-gram scores from the introducing active. Since we introduce a new grammar rule in an empty active edge, the grammar score has to be updated accordingly. This mechanism ensures that an outside score is always maximal: Since we process the agenda above the beam in a best first fashion, it is guaranteed that the first SEEK DOWN operation leading to a certain empty active is the one with the best score. OS refers to the combined score, OS.X to the single component X of that score accordingly.[10]

[7] The experiments in Hauenstein & Weber 1994 were done only with acoustic and n-gram scores.
[8] ..supplied with the transition penalties between m and j.
[9] This is not really true, since we could use top down filtering (as described by Wirén [21]) during bottom up parsing.
[10] Again in record notation.

Definition 3.3.2 *OUTSIDE score for the start edge*

$$\text{OS}(E_{0,0,\emptyset,\text{Startgraph}}) = 0 \tag{3.6}$$

Definition 3.3.3 *OUTSIDE score for empty actives resulting from a SEEK DOWN operation.*

$$\begin{aligned}
&\text{OS.X}(E_{j,j}) = \text{OS.X}(E_{i,j})\\
&\text{OS.GO}(E_{j,j}) = \text{OS.GO}(E_{i,j}) + 2\\
&\text{OS.GP}(E_{j,j}) = \text{OS.GP}(E_{i,j}) +\\
&logP(type(E_{j,j}.\text{mother})|type(E_{i,j}.\text{next}), \text{gm}) +\\
&logP(E_{j,j}.\text{rule}|type(E_{j,j}.\text{next}), \text{gm})
\end{aligned} \tag{3.7}$$

$E_{j,j}$ is introduced by $E_{i,j}$,
and X being AO, AP, NGO, NGP

Do define the OUTSIDE score which is given to a resulting edge by an application of the fundamental rule to its daughter active and inactive, we must refer to AO, NGO, GO, AP, NGP GP, and the strings covered by edges. In record access[11] notation: A.intro ist the string of word hypotheses from vertex 0 through the lattice, on which an OUTSIDE score of A ist based on. A.string ist the string of word hypotheses actually covered by A.

Definition 3.3.4 *OUTSIDE score of an edge C, resulting from an active A and an inactive B by an application of the fundamental rule.*

$$\begin{aligned}
&\text{OS.AO}(C) = \text{OS.AO}(A) + \text{IS.AO}(B)\\
&\text{OS.AP}(C) = \text{OS.AP}(A) + \text{IS.AP}(B)\\
\\
&\text{OS.NGO}(C) = \text{OS.NGO}(A) + \text{IS.NGO}(B) + 1\\
&\text{OS.NGP}(C) = \text{OS.NGP}(A) + \text{IS.NGP}(B) +\\
&\text{logP}(\text{last}(\text{A.intro})|\text{first}(\text{B.string}), \text{n-gram})\\
\\
&\text{OS.GO}(C) = \text{OS.GO}(A) + \text{IS.GO}(B) + 1\\
&\text{OS.GP}(C) = \text{OS.GP}(A) + \text{IS.GP}(B) +\\
&\text{logP}(\text{type}(\text{B.mother})|\text{type}(\text{A.next}), \text{gm})
\end{aligned} \tag{3.8}$$

$$\text{OS(C)} = \frac{\text{OS.AP(C)}}{\text{OS.AO(C)}} + \gamma \frac{\text{OS.NGP(C)}}{\text{OS.NGO(C)}} + \delta \frac{\text{OS.GP(C)}}{\text{OS.GO(C)}} \tag{3.9}$$

When we combine two edges we compute new values for all the single components of the INSIDE and OUTSIDE score.

3.4 Probabilistic Typed Unification Grammars

We wanted our parser not only to help the decoder to find the best word sequence. What was inteded originally was a mapping from a signal to a formal representation of a meaning. So, since the typical grammar produces a lot of derivations for a given string, we had to add some disambiguation device in order to choose one of those multiple analyses. Secondly, we did not intend to compute all of the derivations but rather prune those which were not intended early.

[11] which is really the way we implemented it.

The straight way was to have a probabilistic model of the derivations of our unification grammar. The scores can be combined with the other scores and the general beam search will have effect on the completer operations coming from different analyses of the same word string.

Besides the work on PCFGs by Baker [2], Jelinek [13], Fujisaki [8], and others there is a handfull of publications on probabilistic versions of unification grammars, like Hemphill & Picone 89[12] [12], Briscoe & Carroll 93[13] [3] or Magerman & Marcus 91 [14][14].

All this work is on unification grammars that do not use typed feature structures. So the main thing is to identify finitely many classes of feature structures in order to apply PCFG methods to the observation of derivations of the unification grammar. Mapping of infinitely many FS to a set of classes is usually done by creating a set of *restrictions* in the sense of Shieber 85 [19], which do not unify which each other. Observing the classes in a derivation instead of the original FS leads to the possibility to decide to which class a left hand side of a rule belongs. So we can train some n-gram models on right and left hand sides of rules.

In a typed unification grammar like ours, we use a type system with *appropriatness* as defined in Carpenter 91 [4]. In such a system, we have a finite set of type names associated with well formedness conditions on FS. Using the types of FS as class names, we do not face the same problem as we do using unification grammars without types.

On the other hand, a type is not a fixed label as eg in CFG. Simple types are usually defined in an IS-A hierarchy which determines the unifiability and subsumption relations of types. Complex types arise from unification of types where two types unified have several common subtypes.

Assume the unification of two typed FS as in (3.10):

$$A[f1 : v1] \cup B[f2 : v2] = C[f1 : v1, f2 : v2] \tag{3.10}$$

The type C can be equal to A or B or be a common subtype of both.

In our grammar rules global types of FS encode much linguistic information and a lot of linguistic relations are encoded in the type hierarchy instead of grammar rules. An example is the instantiation of the so called *Vorfeld* in german sentences. The following simplified example shows the method.

$$\text{S2\_intrans[..]} \rightarrow \text{Vorfeld[..] V\_intrans[..]} \tag{3.11}$$

$$\text{Vorfeld} > \text{NP PN Perspron S\_adv...} \tag{3.12}$$

Having a number of rules like the one in (3.11) in our typed unification grammar, the type declaration in (3.12) specifies, FS of which type can instanciate with FS of type *Vorfeld*. The former are just stated to be subtypes of the latter.

Classifying FS by type names and using them directly for a PCFG style probabilistic model leads to trouble. We cannot guarantee to always have $\sum_\beta P(\beta|\alpha) = 1$ for rule schemata $\alpha \rightarrow \beta$. since we do not Know what α's actual instances will be. Compiling out the types and multiplying the rules accordingly would be possible, since in our system we only have a finite number of types – the power set of all simple types. One the one hand this leads to a huge set of rule instances, on the other hand there

[12] Don't read it.

[13] Read it.

[14] The work on Pearl is rather on replacing a unification grammar, but nevertheless on this topic.

are systems like the one of Emele & Zajac 90 [7], where infinitely many types may occur. So the method is not general.[15]

So what we do in order to get a proper statistics on derivations on the typed FS rules is to decompose a rule application and a unification of two typed FS into two parts: The shift of the types beeing unified and the application of the rule itself. When for instance, in a sentential form of a left derivation a FS of type *Vorfeld* is unified with a FS of type *NP*, we have two observations, the shift of types *(NP, Vorfeld)* and the application of a rule with original lhs type *NP*. In fact we do not care about the result type of the type unification which might be a complex type.[16]

Definition 3.4.1 *Given a left derivation in a typed unification grammar of the form:*

$$S[..] \stackrel{*}{\Rightarrow} x\alpha' y \Rightarrow x\beta' y \stackrel{*}{\Rightarrow} w \tag{3.13}$$

where α' has been generated as an instance of α'' a member of the right hand side of some rule, and β' has been generated by application of a rule $\alpha \rightarrow \beta$ by unification of α and α', we call the pair type(α''),type(α) an **observed type shift***.*

In our model *gm* for the derivation of the typed unification grammar we describe each rule application – in other words: step in a left derivation – as a pair of a type shift and original rule of the grammar.

Implicitly this is adding some unary rules for all type shifts and guaranteeing that in every derivation type shifts and rules are applied alternatingly.

Definition 3.4.2 *A probabilistic typed unification grammar is:*

Type hierarchy: *A lattice of types with a top and bottom element which defines the subsumption relations of types.*

Lexicon: *We assume the lexicon to be a set of unary productions, where the right hand side of the production is a word string.*

N-ary grammar rules: *The grammar rules consisting of one typed feature structure as a left hand side and a sequence of typed feature structures as a right hand side.*

Model gm: *Assigning a probability to all pairs of two types and to all pairs of type and rule.*

The model *gm* is organized as two bi-grams, one for the type shifts and one for the rules. The type shifts and rules are thought to be independent. So the following relation holds:

$$\sum_{B} \sum_{\beta} (A \stackrel{\text{tsh}}{\rightarrow} B, B[..] \stackrel{\text{rule}}{\rightarrow} \beta | gm) = 1 \tag{3.14}$$

During training, we treated the type shifts as if they were unary productions. We parsed a corpus with the original typed unification grammar. Parses were represented as lists of the type shifts and numbers of rules that were involved. We used a variant of the unsupervised training method of Fujisaki et al. 91 [8] to estimate the models.[17]

During parsing we can distinguish three types of type shifts.

[15] It is ugly, which is the main argument.

[16] We could have done this, but the amount of paramters would have increased by $O(2^n)$, where n is the number of simple types.

[17] We do not object to supervised learning. We just did not have a tree bank.

- Those pairs of types which are not unifiable are type shifts of probability 0. Thus, we can prune an impossible analysis on grounds of *gm* instead of performing a unification that will fail anyway.
- Possible but not observed type shifts should receive a smoothed value as usual in standard bi-gram techniques.
- Observed type shifts receive their trained probabilities.

By our method of having an additional bi-gram of type shifts, we achieve a couple of pleasing properties. The first one is, that we can have a correct probabilistic model of a unification grammar which relates typed feature structures with each other. Furthermore, by using the types as model classes, we can capture a lot of information of the unification grammar in our model, since types are tied to the feature structures by *appropriatness* checking as described in Carpenter 91 [4] and the subsumption relations of types in the hierarchy are captured as well. The method generalizes to the powerful type systems in the style of Emele & Zajac 90 [7] since only pairs of simple types[18] are used in the model. Since in type unification the resulting type is determined by these two types we do not lose any information.

Finally, if for a given grammar the hierarchy of types is flat, the model works like a real PCFG backbone for a unification grammar, as it is used for example by the TINA parser [18]. In that case, all types are unifiable only with themselves and with the top element of the type lattice. When no rules of the grammar use top as a global type of a FS, we will arrive at a collection of observed type shifts all of which have the form (X,X) with probability 1.

3.5 Tightly Coupling the Parser with a Beam Decoder

Some extensions of the LR-ACP allow for different time synchronous couplings with a beam decoder for word hypotheses. So far we investigated three modes, namely time synchronous parallel bottom up (*BUI*), time synchronous bottom up with top down verification (*BUITDV*) and time synchronous top down predicting mode (*TDPI*). Some results of these couplings driving the parser with an n-gram only have already been presented in Hauenstein & Weber 94 [10, 11]. The decoder is a viterbi one pass beam decoder[19] extended for the different protocols as explained below.

3.5.1 BUI

In BUI we run the decoder and the parser concurrently. In every frame processed by the decoder, word ending hypotheses above the decoders beam are immediately sent to the parser. The decoder does not use language model[20]. The hypotheses are quadruples *(from,to,key,score)*, where *key* is the name of the wordform and *score* is the acoustic score assigned to the frames *from* to *to* by a word copy of the model for *key*.

The LR-ACP works exactly as described in section 3.2.3.

[18] In fact complex types may also be used. The bound comes from the type names occuring in the original grammar rules before any grammatical operation took place.

[19] .. developed in the VERBMOBIL TP 15 project, by Andreas Hauenstein, University of Hamburg.

[20] But a fixed transition penalty in order to prevent an inflation of short words.

The BUI coupling uses no feedback messages from the parser to the decoder, so real parallelism is possible.

3.5.2 BUITDV

In BUITDV mode the decoder uses top down verifications of the parser as a language model. Since for one word copy a language penalty has to be added only once, a fifth field is added to the bottom up word hypotheses as a flag signalling to the parser whether a hypothesis is new[21] or has already been verified. The whole procedure works as follows.

Definition 3.5.1 *An example cycle of BUITDV for frame I:*

1. *The decoder finds m+n new word ending hypotheses at frame I. n had already been found at I-1, m are new. All of them are sent to the parser in cycle I with the flags set accordingly.*
2. *The parser takes all m+n word hypotheses and performs PUSH-EDGES resulting in an initial* $agenda_I$.
3. *The parser processes* $agenda_I$. *The first successful application of the fundamental rule to a pair of edges involving a new word hypothesis leads to a verification of that hypothesis.*
4. *When the beam of* $agenda_I$ *is reached, the rest of* $agenda_I$ *is searched for word hypotheses not yet verified. The first one found leads to a verification of that hypothesis.*
5. *The decoder receives verification messages. All those new hypotheses in I, which were not verified are set to a zero probability. All other new hypotheses have the verified language penalty added.*

Verification messages are built by the parser using agenda items. They are quintuples *(from, to, key, score, flag)* as the associated bottom up hypotheses are. *from*, *to* and *key* identify the decoder's active word copy which was to be verified. The *flag* field is supplied with the predecessor word hypothesis' string, which led to the best score inside the parser's search. For *score* a couple of options are given.

Definition 3.5.2 *Transition score with bi-gram and gm supplied by a verification message for a pair (A, W). of edges on the agenda.*

$$\begin{aligned}&score = \\ &\gamma * logP(first(W.string)|last(A.intro), n\text{-}gram) + \\ &\delta * logP(type(W.mother)|type(A.next), gm) + \\ &\delta * logP(W.rule|type(W.mother), gm)\end{aligned} \tag{3.15}$$

Definition 3.5.3 *Transition score with bi-gram supplied by a verification message for a pair (A, W). of edges on the agenda.*

$$\begin{aligned}&score = \\ &\gamma * logP(first(W.string)|last(A.intro), n\text{-}gram)\end{aligned} \tag{3.16}$$

[21] Then this word copy's final state has been above the decoder's beam for the first time

The score which is used as a transition penalty between words by the decoder can consist of a bi-gram score and a score given by the grammar model for those grammar operations, which are used to incorporate the word hypothesis into an existing partial analysis. The latter consist always of the score for the lexical access[22] and the type shift which takes place when the lexical entry is combined with some rule actually processed by some active edge. This is the part of the grammar operations which can directly be related to a local transition between words. Again both scores are weighted, hence not normalized. Alternatively, only the bi-gram score can be used as in common decoding.

Maximization is not local here. Since predecessor word and analysis are selected on the base of agenda items, the criterion for the maximization on predecessors is global.

One difficulty we have to face is that the decoder will have access to transition penalties when a word model runs into a final state and not at the beginning of a word. We can overcome this problem with a method inspired by Aubert's et al. 1994 [1] handling of tree lexica in one pass decoding. For both the n-gram transition and the part of the grammar score used above there exists a stable maximum for a word with respect to all possible immediate predecessor words.

So starting a word copy in the decoder we can use this maximum as a transition penalty adding the difference with respect to the maximum when the copy has been verified.

3.5.3 TDPI

TDPI is the opposite method of coupling to BUITDV in terms of contol, since the parser predicts possible successor word hypotheses starting in a given frame while the decoder selects among those.

The basic procedure in TDPI goes as follows:

Definition 3.5.4 *An example cycle of TDPI for frame I:*

1. *The decoder has sent all word ending hypotheses at frame I.*
2. *Having parsed all word hypotheses ending in vertes I, the parser takes all the active edges in vertex I and calculates a set of predicted predecessor words. These are sent to the decoder supplied with transition scores as a* **prediction** *for I.*
3. *The decoder in frame I starts only those new word copies which occur inside the prediction. Newly started word copies are initialized with the supplied transition penalty.*

A grammar based calculation of predictions is too expensive since a full backward search (completer step) would have to be executed for each candidate. So we propose a generate and test algorithm for the computation of a set of predictions, based on the chart up to vertex I:[23]

Definition 3.5.5 *Computing predictions*

1. *Let ACTIVE be the nonempty active edges ending in* $vertex_I$, $P_I = \emptyset$

22 treated as an unary rule, namely W.rule in def. 3.5.2.

23 A broad discussion of efficiency in chart based computation of predictions can be found in Weber 94 [20].

2. *Take the n best successors s of any w = last(A.intro), $A \in ACTIVE$, maximizing* $logP(s|w, n\text{-}gram)$.

3. *Create* $vertex_{-I}$ *and insert inactive edges for all s into* $vertex_I$, $vertex_{-I}$, *thus filling* $agenda_{-I}$.

4. *Until below_beam*($agenda_{-I}$), *let (A,S) be pop*($agenda_{-I}$):

   When combine(A,S) $\neq$ *fail, remove all pairs (X,S) from* $agenda_{-I}$ *and set* $P_I = P_I \cup$ *(I,0,S,last(A.intro),score).*

The score of a quintuple (from, to, predicted word, predecessor, score) in P_I is the same as for the verification messages given in 3.5.2. Again, the actual score returned is globally selected. The reason for the n best technique in step 2 is, that we can precompute this step and hash the results. So the complexity of the whole calculation is determined by n times the number of active edges ending in vertex I in the worst case. We do not use a completer step to test the predictions, since one success with some active edge is sufficient to show that there is at least one spanning analysis according to the type skeleton of the grammar. Adding all the features to the test would be prohibitive. If we use a class based n-gram model where the classes correspond with types of our grammar we could reduce n considerably.[24]

3.6 Incremental Time Mapping using Edge Inheritance

The set of word hypotheses transferred bottom up from the decoder to the parser in a time synchronous coupling constitutes a *left connected word graph* seen a posteriori. In a non incremental architecture where word recognition ends before parsing starts, we can delete all dead ends in a word graph using an offline backward search phase through the set of word hypotheses. An abstracting step from frames to vertices can omit superfluos paths in the graph consisting of word ending hypotheses differing only in one frame. Chien et al. 90, 93 [6, 5] present such a mapping from frames to vertices.[25]

In our incremental architecture this does not work. When the decoder finds a word ending hypothesis we never know whether the word copy in question will still be above the beam in the next frame and if yes, whether its normalized acoustic score will increase or decrease. This leads to the effect, that sometimes the parser receives exactly the same set of word hypotheses in two successor frames, differing only with respect to their end vertex. The same completer actions are carried out two times in such a case. In order to implement a normalization without any lookahead, we developed a method we call incremental time mapping, which consists of a slight modification of the LR-ACP where some book-keeping is used to do all completer actions only once which stem from the same decoder's word copy. The basic idea is to inherit all those edges from vertex I to vertex I+1 which resulted from a word hypothesis in vertex I for which an identical one has later been found with ending time I+1. All word hypotheses found for the first time are parsed as usual.

For that issue, we add a little non monotonism to the chart parser:

[24] We did not try that yet.

[25] Their method is a bit cryptical, but the only reference we found in the literature.

- One edge can belong to a number of vertices. Seen from the vertices the edge is shared. Seen from the edges, an edge now possesses a list of end vertices the head of which is the actual one.
- We allow for the replacement of edges by others. This mechanism is only used on empty active edges. It is guaranteed that no inconsistencies arise, since we replace only edges ending in vertex I in cycle I.

The basic modified control loop for the LR-ACP can be paraphrased as follows.[26]

Definition 3.6.1 *Descripition of a $cycle_I$ in an LR-ACP with edge inheritance.*

READ HYPOS *:*

KNOWN, NEW, INH_EMPTIES be $\emptyset$.

Read in all hypotheses ending at I. Add those which have a predecessor at I-1 differing only in ending time to KNOWN.

Let pred(KNOWN) be the corresponding hypotheses in $vertex_{I-1}$. Add all others to NEW.

INHERITANCE *:*

For all edges A ending in $vertex_{I-1}$, A coming from some $w \in$ pred(KNOWN):

When score(w) < score(w)' in KNOWN, update the acoustic score and operations of A.

If empty_active(A), add A to INH_EMPTIES

Else, add A to $vertex_I$.edge-in, add $vertex_I$ to A.to. When A is active, perform a SEEK-DOWN on A in $vertex_I$.

For all the pairs (A,B) in $save\_agenda_{I-1}$, B coming from some $w \in$ pred(KNOWN), push (A,B) to $agenda_I$.

PARSE *:*

Insert all word hypotheses in NEW into the chart. MAXVALUE := max(max(OS(A) | A $\in$ $vertex_I$.edge-in), score(top($agenda_I$)))

BEAMVALUE := MAXVALUE - BEAMWIDTH

Apply fundamental rule to pop($agenda_I$) until score(top($agenda_I$) $\leq$ BEAMVALUE.

Save $agenda_I$.

UPDATE EMPTIES *:*

For all A in INH_EMPTIES: If there exists no empty edge B in $vertex_I$, with A.rule = B.rule, add copy(A) to $vertex_I$. Else, if B.OS < A.OS, replace B by copy(A).

In order to keep the beam search mechanism working as in the original LR-ACP we have to take care of the inherited edges, when the BEAM is calculated. Furthermore, since the scores of edges inherited may be subject to an updating and since the new maximum in cycle I may differ from the maximum in cycle I-1 those pairs with inherited edges, which had been pruned in cycle I-1 must be pushed on the new agenda.

[26] A more precise description can be found in Weber 94 [20].

Nevertheless, scoring is different from the original version, since we only update an edges acoustic score until a maximum is reached.

In the original version, a sequence of say 5 word hypotheses coming from the same decoder,s word copy was represented by 5 edges[27]. One of those had the maximal acoustic score. Using incremental time mapping, we represent the 5 word hypotheses by only one edge, having 5 possible updates on the acoustic scores. In order to keep the global Viterbi search correct, we stop updating when the maximum is reached, so the whole sequence of original hypotheses will be represented by its maximum. The mechanism is similar to the pruning used in Ney's 91 [15] dynamic programming parsing based on a CYK parser, but constitutes an incremental time synchonous version of the latter.

While the algorithm above works well for BUI and TDPI, in BUITDV some slight modification have to be made. In BUITDV the decoder sends all those hypotheses first which are to be verified, waiting for response, and sends all known hypotheses later. So the decoder can prune based on verified hypotheses only but the parser cannot use the maximum of all scores in the first phase, parsing the new ones. A version of the parser with time mapping using two distinct steps for the new and known hypotheses worked well in experiments although the first step did not have certain knowledge about the maximum in every cycle.

The calculation of predictions in TDPI can be subject to the inheritance technique too. All predictions in a cycle I will be pointed to by the edges they are based on. When the relevant edges are inherited, the predictions coming from them will be inherited with them then.[28]

3.7 Selected Experiments

In this section we present a couple of selected experiments with the LR-ACP. The n-gram model used in all of the experiments is word based and has a test set perplexity of 54.28. We used two unification grammars for our experiments. A small and rather restrictive one (GRS) was used in the experiments without the probabilistic grammar model. A larger, less restrictive grammar (GRB) with more structure building features was used with the model *gm* in order to have a more realistic test bed. A corpus of 200 sentences of the train information domain was used for training. Tests were carried out on 10 sentences with the small grammar and 5 sentences with the big one. The acoustic models used where well adapted and led to an acoustic word accuracy of 88.6 on the test sentences.

The experiments should be regarded with care. They are of a small scope they and preliminary in nature. Further testing of the architecture and its variants on a new domain with more realistic grammar and bi-gram are currently at work.

The two tables below present the results of Hauenstein & Weber 94 [10, 11] for decoding alone (AC), BUI (BU), BUITDV (BV) and TDPI (TP), using GRS with the bigram only. We measured cpu time (T)[29], bottom up hypotheses (BUH), top down hypotheses (TDH), sentence recognition (SR), edges used (E) and maximal active gridpoints (GP).

The first table shows results for narrow beams, the second for wide beams.

[27] Or $5n$ edges, when n lexical entries where found.
[28] Again, details can be found in Weber 94 [20].
[29] Allegro Commom Lisp on a SUN Sparc10, parsing time.

	T	BUH	TDH	SR	E	GP
AC	47.4	–	–	0.5	–	790
BU	96.9	385	–	0.7	6195	790
BV	58.2	289	41	0.5	5827	1045
TP	220.5	223	7621	0.7	4954	78

	T	BUH	TDH	SR	E	GP
BU	2911.7	1465	–	0.7	19769	1712
BV	115.8	776	135	0.7	9765	2047
TP	282.7	415	8304	1.0	7519	184

The effect of the top down strategies cannot be overseen here. The amount of bottom up hypotheses is heavily reduced by the restriction supplied to the decoder.[30] For the relatively well adapted hmms, the TDPI strategy leads to a sentence recognition of 100 percent using wide beams. BUITDV is superior to BUI only in terms of efficiency.

The effect of the time mapping can be seen in the following table, which changes the overall impression given by the first results. All parameters besides the time mapping are as those used in the wide beam case. The table shows BUI, BUITDV and TDPI with incremental time mapping. TDPI has been tested with standard time mapping only (TM) and with time mapping for the computation of predictions (TMP). The comparison with time mapping has been done on five of the test sentences, which were well recognized by all the couplings.

	Time in Sek.	Edges
BUI, TM	68.8	6613
BUITDV, TM	51.4	4662
TDPI, TM	172.6	5004
TDPI, TM & TMP	58.2	4226

We can see, that in terms of efficiency the strategies do not lead to similar results, when redundant completer operations are omitted. The TDPI case with standard time mapping gives a good impression of the enormous overhead caused by the calculation of predictions.

The first results with a larger grammar (GRB) and the model gm are presented below. The first column shows the weights for *n-gram* and *gm*. The single weights were optimized on the test sentences by hand, maximizing sentence recognition rate. The rows show *gm* only vs *n-gram* only vs a combination of both models. PR and TR stand for pruned and tried agenda items. T, SR and E are defined as above. The table shows a version of GRB without rules for performance phenomena and a BUI coupling.

ng/gm	PR	TR	T	SR	E
0.0 0.4	19920	2075	103	4/5	7774
0.2 0.0	7031	25796	137	3/5	8874
0.1 0.2	26381	2486	108	4/5	8192

The full GRB grammar and TDPI coupling with n-gram value supplied to the decoder with the prediction, led to the following results:

[30]BUI without time mapping often exhausted the machines 48mb RAM causing swapping which explains the huge increase in cpu time.

ng/gm	PR	TR	T	SR	E
0.0 0.4	69052	530	145	5/5	6627
0.2 0.0	22588	5343	171	5/5	6648
0.1 0.2	69858	599	157	5/5	6646

The most prominent observation is, that when the influence of *gm* is set to zero, much more agenda items are tried. The reason for this is, that on the basis of *gm* a lot of unifications can be pruned, that would fail with a type mismatch if tried. So these agenda items do not lead to additional edges. They are responsible for a growth of processing time, since the processing of the scores is cheaper than unification. Astonishing there is a slight increase in processing time and edges from *gm* alone to the combination of the models. We had expected the opposite effect. The weights, which were optimized by hand, might be the reason for that. A better tuning of the weights by an optimizing procedure, taking smaller steps than our hand tuning, should correct this.

3.8 Conclusions

We presented an active chart parser which combines different statistical knowledge sources in parsing speech in a time synchronous fashion. The beam search implemented on the agenda takes global scores into account.

On the basis of this parser, different couplings with a decoder in the style of Hauenstein & Weber 94 were explicated. The time mapping technique presented here leads to the effect, that differences in efficiency between the couplings observed by Hauenstein & Weber 94 become small.

TDPI is superior to the other couplings with respect to recognition rate in the experiments performed. This does still hold, when a statistical model of the unification grammar's derivations is included.

We introduced a simple method to supply a typed unification grammar with a statistical model of its derivations by observing type shifts on the one hand and type-rule pairs on the other hand. The method generalizes to most known type systems used for such applications.

Further experiments, larger in scale have to be performed in the future.

Bibliography

[1] X. Aubert, C. Dugast, H. Ney, and V. Steinbiss. Large vocabulary continuous speech recognition of wall street journal data. In *ICASSP*, 1994.

[2] J.K. Baker. Trainable grammars for speech recognition. In *Speech Communication Paper, 97th Meetingh of the Acoustical Society of America*, Cambridge, Mass., 1979. MIT Press.

[3] Ted Briscoe and John Carroll. Generalized probabilistic lr parsing of natural language (corpora) with unification-based grammars. *Computational Linguistics*, 19(1):25–59, 1993.

[4] Bob Carpenter. *Typed Feature Structures: Inheritance (In)equality and Extensionality.* CMU, 1991.

[5] Lee-Feng Chien, Keh-Jiann Chen, and Lin-Shan Lee. A best-first language processing model integrating the unification grammar and markov language model for speech recognition applications. In *IEEE Transactions on Speech and Audio Processing*, volume 1,2, pages 221–240, 1993.

[6] Lee-Feng Chien, K.J. Chen, and Lin-Shan Lee. An augmented chart data structure with efficient word lattice parsing scheme in speech recognition applications. In *Proceedings of COLING*, pages 60–65, 1990.

[7] Martin Emele and Rémi Zajac. A fixed-point semantics for feature type systems. In *Proc. of the 2nd International Workshop on Coditional and Typed Rewritung Systems (CTRS)*, Montreal, June 1990.

[8] T. Fujisaki, F. Jelinek, J. Cocke, E. Black, and T. Nishino. A probabilistic parsing method for sentence disambiguation. In Masura Tomita, editor, *Current Issues in Parsing Technology*, pages 139–148, Norvell, Mass., 1991. Kluver Akademic Publishers.

[9] Günther Görz. *Strukturanalyse natürlicher Sprache.* Addison Wesley, Bonn, 1988.

[10] A. Hauenstein and H. Weber. An investigation of tightly coupled time synchronous speech language interfaces using a unification grammar. In Paul McKevitt, editor, *Proceedings of the Workshop on Integration of Natural Language and Speech Processing at AAAI 94*, pages 42–49, Seattle, August 1994.

[11] A. Hauenstein and H. Weber. An investigation of tightly coupled time synchronous speech language interfaces. In *Proceedings of the CONVENTS 94*, Wien, September 1994.

[12] Charles Hemphill and Joseph Picone. Chart parsing of stochastic spoken language models. In *DARPA Workshop on Speech and Natural Language*, Philadelphia, Pensylvania, February 1989. DARPA.

[13] F. Jelinek. Basic methods of probabilistic context free grammars. In P. Laface and R. De Mori, editors, *Speech Recognition and Understanding: Recent Advances*, volume NATO ASI Series, F 75, pages 345–360, Berlin Heidelberg, 1992. Springer Verlag.

[14] D.M. Magermann and M.P. Marcus. Pearl: A probabilistic chart parser. In *Proc. of the European ACL*, March 1991.

[15] Hermann Ney. Dynamic programming parsing for context-free grammars in continuous speech recognition. *IEEE Transactions on Signal Processing*, 39(2):336–340, February 1991.

[16] Hermann Ney. *Architecture and Search Strategies for Large-Vocabulary Continuous-Speech Recognition.*, pages 59–84. NAT)-ASI Bubión, 1993.

[17] Annedore Paeseler. Modification of earleys algorithm for speech recognition. *NATO ASI Series: Recent Advances in Speech Understanding*, F46:465–472, 1988.

[18] S. Seneff. Tina: A probabilistic syntactic parser for speech understanding systems. In *DARPA Speech and Natural Language Workshop*, Philadephia, February 1989.

[19] Stuart M. Shieber. Using restriction to extend parsing algorithms for complex-feature-based formalisms. In *Proc. of the ACL 1985*, volume 23, pages 145–152, 1985.

[20] Hans H. Weber. *LR-inkrementelles probabilistisches Chartparsing von Worthypothesenmengen mit Unifikationsgrammatiken: Eine enge Kopplung von Suche und Analyse.* PhD thesis, Submitted to Universität Hamburg, FB Informatik, Dezember 1994.

[21] M. Wirén. *Studies in Incremental Natural-Language Analysis.* Number Dissertation No. 292 in Linköping Studies in Science and Technology. Linköping University, 1992.

Kapitel 4

Parsing unter Zeitbeschränkungen

Wolfgang Menzel
Fachbereich Informatik
Universität Hamburg
Vogt-Kölln-Straße 30
D-22527 Hamburg

Zusammenfassung

Systeme zur Interpretation gesprochener Sprache sind in anspruchsvollen Anwendungssituationen einem ständigen zeitlichen Verarbeitungsdruck ausgesetzt. Um zu sichern, daß die Verarbeitung mit dem kontinuierlich eingehenden Sprachsignal schritthält, ist eine flexible Anpassung der vorhandenen Ressourcen an die wechselnden Lastbedingungen des Analyseprozesses erforderlich. Insbesondere muß sichergestellt werden, daß temporäre zeitliche Engpässe nicht zu einem abrupten Zusammenbruch der Erkennungsleistung führen. Unter den jeweils verfügbaren Constraints sind in Abhängigkeit vom Grad der lokalen Mehrdeutigkeit diejenigen auszuwählen, die eine Reduktion der Hypothesenvielfalt mit möglichst geringem Aufwand ermöglichen.

Interpretiert man Parsing als eine Prozedur zur Disambiguierung in Constraint-Netzen, so erlaubt dies eine Bewertung des Analysefortschritts zu jedem beliebigen Zeitpunkt. Auf dieser Grundlage lassen sich dann Verfahren angeben, die über wesentliche Merkmale des angestrebten (qualitativen) Echtzeitverhaltens verfügen und einige Aspekte der menschlichen Fähigkeit zur Aufmerksamkeitsfokussierung beim Sprachverstehen simulieren.

[1]Dieser Aufsatz ist die deutsche Fassung eines Vortrags, der ursprünglich auf der 11. Europäischen Konferenz für Künstliche Intelligenz, Amsterdam 1994, gehalten wurde [6]. Das Original ist auch als Verbmobil-Report 26 erschienen.

Das dieser Arbeit zugrundeliegende Forschungsvorhaben wurde mit Mitteln des Bundesministers für Forschung und Technologie unter dem Förderkennzeichen 01 IV 101 A/O gefördert. Die Verantwortung für den Inhalt der Arbeit liegt beim Autor.

4.1 Einleitung

Nicht nur im Hinblick auf das menschliche Vorbild, auch aus ganz praktischen Gründen ist die Inkrementalität der Verarbeitungsprozesse eine wesentliche Eigenschaft für Systeme zur Verarbeitung gesprochener Sprache:

- Realitätsnahe Dialogszenarien erfordern die Fähigkeit zur unmittelbaren Reaktion auf Nutzereingaben, die mit der traditionellen Orientierung an "vollständigen", d.h. explizit abgeschlossenen Eingabesequenzen kaum erreicht werden kann. Die Analyse muß mit den eintreffenden Sprachdaten Schritt halten und auch für Folgeaktivitäten, wie die Planung und Generierung einer geeigneten Reaktion ist eine hochgradig parallele Verarbeitung erforderlich, um eine unterbrechungsfreie Kommunikation und damit auch eine reibungslose Mensch-Maschine-Interaktion zu ermöglichen.

- Inkrementelle Verarbeitung ist eine grundsätzliche Voraussetzung für eine Mitwirkung in Dialogen mit verteilter Initiative: Jede Entscheidung zur Übernahme der Initiative setzt eine unmittelbare Analyse und Interpretation partieller Äußerungen voraus, damit die Zeitverzögerung so kurz wie möglich gehalten werden kann.

- Verarbeitungsprozeduren mit minimaler Zeitverzögerung sind besonders vorteilhaft, da sie die Notwendigkeit zur Beschränkung des Speicheraufwandes für Zwischenergebnisse auf natürliche Weise unterstützen.

- Die Verarbeitung gesprochener Sprache ist ein stark erwartungsgesteuerter Prozeß, wobei Erwartungen, die sich aus einem Diskurs- oder Domänenmodell ableiten lassen, eine besonders wichtige Rolle spielen. Daraus ergibt sich, daß Inkrementalität bei der Sprachanalyse eine wesentliche Voraussetzung für die wirksame Einbeziehung von Vorhersagen auf allen Ebenen der Sprachverarbeitung darstellt.

- Inkrementalität ist schließlich unverzichtbar für sehr anspruchsvolle Anwendungen, wie sie etwa die Simultanübersetzung für gesprochenen Sprache darstellt.

Unter all diesen Bedingungen kann eine mit steigender Analysezeit monoton fallende Nützlichkeitsfunktion angenommen werden: Hypothesen über weit zurückliegende Äußerungen können kaum als Grundlage für sinnvolle Reaktionen bzw. wirkungsvolle Vorhersagen über künftige Beiträge des Dialogpartners dienen. In fast allen Fällen dürften approximierte oder unvollständige Analysen wesentlich nützlicher sein als vollkommene aber verspätete Beiträge.

Grundsätzlich erfordert eine zeitsynchrone und inkrementelle Analyse gesprochener Sprache Systemarchitekturen, die zumindest die erforderliche Synchronisation zwischen dem Sprachsignal und den Verarbeitungsaktivitäten unterstützen. Das entspricht dem Minimum an externer Steuerung, welches für hochgradig dezentralisierte, verteilte Architekturen mit Nachrichtenaustausch angenommen werden muß [7]. Die Synchronisation mit dem Sprachsignal soll hierbei dadurch erreicht werden, daß der individuelle Zeithorizont jeder einzelnen Komponente überwacht und die Übermittlung von Erkennungshypothesen mit zu großer Nachlaufzeit eingestellt wird.

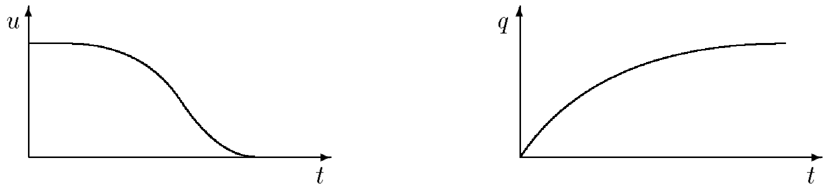


Abbildung 4.1: Nützlichkeitsfunktion und Performanzprofil für eine hypothetische Anytime-Prozedur

Die Steuerung der zeitlichen Verarbeitungscharakteristika eines Systems durch einfache Unterbrechung des internen Kommunikationsflusses setzt jedoch Systemkomponenten voraus, die in der Lage sind, ihren Verarbeitungsaufwand in Abhängigkeit von der noch verfügbaren Zeit zu planen. Dies ist nicht weiter problematisch, so lange die Verarbeitungszeit keine kritische Ressource darstellt. In den meisten praktischen Anwendungsszenarien ist das jedoch nicht der Fall. Sogar für die einzigartigen Sprachverstehensfähigkeiten des Menschen kann die Verarbeitungszeit ein entscheidender Faktor werden, der die "Verstehenstiefe" beträchtlich beeinflußt. Dies ist vor allem dann der Fall, wenn ein oder auch mehrere Stressfaktoren wie Schnellsprache, Fremdsprachenbenutzung, schlechte Artikulation und Nebengeräusche die Sprachwahrnehmung beeinträchtigen.

Charakteristisch für derartige Situationen ist eine ungleichmäßige Verteilung des Analyseaufwands über das Sprachsignal hinweg: In einer Art von "kursorischem Verstehen" versucht der Hörer bestimmte Teile des Sprachsignals herauszugreifen, wobei Relevanzabschätzungen eine zentrale Rolle spielen. Jeder Versuch ungerechtfertigt viel Aufmerksamkeit auf unwesentliche Abschnitte des Sprachsignals zu konzentrieren, kann den Fortgang der Sprachverarbeitung ernsthaft gefährden.

Werden Sprachverarbeitungssysteme angestrebt, die sich ebenfalls flexibel an wechselnde zeitliche Beschränkungen anpassen können, müssen ihre Komponenten in der Lage sein, auf einen ständig enger werdenden Zeithorizont angemessen zu reagieren. Um unter derartigen Bedingungen noch sinnvolle Resultate zu erzeugen, muß ein reproduzierbarer Zusammenhang zwischen dem Zeitbedarf für die Lösung einer speziellen Aufgabe und der zu erwartenden Qualität der erzeugten Ergebnisse gegeben sein, auf dessen Grundlage letztendlich ein Abwägen zwischen Aufwand und Nutzen möglich wird.

Prozeduren, die dies gewünschte monotone Wachstum der Ausgabequalität aufweisen, werden auch Anytime-Module genannt (Boddy and Dean [1], [2], Russel and Zilberstein [8]). Auf ihre Bedeutung für die Entwicklung von Systemen zur Verarbeitung gesprochener Sprache hat erstmalig Wahlster [9] hingewiesen. Die wesentliche Eigenschaft eines Anytime-Moduls ist die Existenz eines *Qualitätsmaßes* (z.B. Sicherheit, Genauigkeit oder Spezifität). Sein (probabilistischer) Verlauf in Abhängigkeit von der Zeit wird durch ein Performanzprofil erfaßt. Russell und Zilberstein unterscheiden zwischen zwei Fällen von Anytime-Verhalten:

- *Unterbrechbare Algorithmen*, bei denen Unterbrechungen ohne vorherige Ankündigung auftreten können. Die betreffende Komponente ist jederzeit in der Lage ein Resultat mit dem durch das Performanzprofil vorgegebenen Qualitätsniveau zur Verfügung zu stellen.

- *Vertragsalgorithmen*, die nur dann ein sinnvolles Resultat mit der angegebenen Qualität liefern, wenn ihnen ein Zeitintervall mit zuvor genau spezifizierter Dauer zur Verfügung steht. Vor Ablauf dieses Intervalls kann überhaupt kein brauchbares Ergebnis erwartet werden.

In beiden Fällen wird davon ausgegangen, daß das Performanzprofil für die zu bearbeitende Aufgabe im voraus bekannt ist. Dies ist allerdings eine verhältnismäßig strenge Bedingung, die für bestimmte Arten von kombinatorischen Algorithmen einfach nicht vorausgesetzt werden kann.

4.2 Anytime-Parsing

Traditionelle Parsingalgorithmen erfüllen die Anytime-Bedingung in keiner Weise. So ist etwa eine Tiefe-zuerst-Analyse mit Ausnahme des allerletzten Zeitabschnitts vor dem Auffinden einer Interpretationsvariante fast ausschließlich mit der Untersuchung nutzloser Sackgassen beschäftigt. Auf den ersten Blick scheint eine Breite-zuerst-Strategie bessere Voraussetzungen für ein Anytime-Verhalten mitzubringen, allerdings betrifft das beobachtbare montone Wachstum eher die *Vollständigkeit* konkurrierender Strukturbeschreibungen und nicht die eigentlich gewünschte Verbesserung bezüglich eines ausgewählten Qualitätsparameters für *eine* globale Beschreibung der zu analysierenden Daten. Unterbricht man einen Chart-Parser noch ehe er eine einzige komplette Analyse ermittelt hat, verfügt er entweder über eine Menge unvollständiger und noch nicht verifizierter Strukturbäume (Top-down-Modus) oder eine Menge alternativer und möglicherweise widersprüchlicher Baumfragmente. Im allgemeinen Fall muß man davon ausgehen, daß kein Wissen darüber verfügbar ist, wie sich diese partiellen Beschreibungen zu einem brauchbaren Analyseresultat kombinieren lassen. Darüberhinaus existiert auch kein geeignetes Qualitätsmaß, mit dem man den Analysefortschritt beschreiben kann [4], ganz zu schweigen von einem vorhersagbaren Performanzprofil.

Tatsächlich gibt es aber einige alternative Parsing-Ansätze, die erheblich besser mit den Erfordernissen einer Anytime-Analyse verträglich sind als der übliche Ansatz des Chart-Parsing. Ein Besispiel dafür ist die Satzanalyse durch Baumtransformation, wie sie schon vor mehr als zehn Jahren im ARIANE-System zur Maschinellen Übersetzung [3] verwendet worden ist. Die Eingaberepräsentation besteht hierbei aus einem flachen Baum für den zu übersetzenden Satz, in dem alle terminalen Knoten direkt vom Spitzenknoten dominiert werden. Das Parsing besteht aus einem schrittweisen Ersetzen von Teilbäumen durch jeweils stärker strukturierte Teilbäume, um dadurch schließlich eine Beschreibung der Konstituentenstruktur im üblichen Sinne zu erhalten. Einerseits gestattet es dieser Ansatz, die Module für Analyse, strukturellen Transfer und Generierung auf der Grundlage eines einheitlichen Formalismus (ROBRA) zu entwickeln. Andererseits bietet er einen ganz natürlichen Auffangmechanismus für den Fehlerfall: Scheitert der Parser bei der Suche nach einschlägigen Baumtransformationsregeln, übergibt er den (teilweise) unveränderten Baum an die nachfolgende Transferstufe, die dann eine Wort-für-Wort-Übersetzung mit deutlich schlechterer Qualität produzieren wird.

Geht man von einer im wesentlichen monotonen Verbesserung der Übersetzungsergebnisse durch die schrittweise Anwendung zusätzlicher Transformationsregeln aus, dann kann der Grad der strukturellen Komplexität einer Strukturbeschreibung durchaus als grobes Qualitätsmaß im Sinne der Anytime-Bedingung interpretiert werden. Unterbricht man die Baumtransformation zu einem beliebigen Zeitpunkt, ist sie stets in

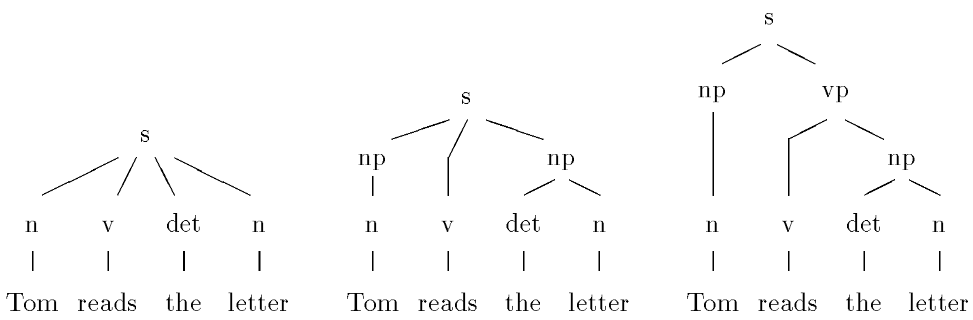


Abbildung 4.2: Ablaufschema für eine (deterministische) Baumtransformation

der Lage eine oder auch mehrere Beschreibungen des Eingabesatzes zur Verfügung zu stellen. Diese Beschreibungen erfassen in jedem Fall die vollständige Eingabe, werden aber in Abhängigkeit von der aufgewendeten Analysezeit mehr oder weniger strukturiert sein. Dies korrespondiert tatsächlich mit einer Art von Anytime-Verhalten in genau dem gewünschten Sinne: Je mehr Zeit zur Verfügung steht, desto höher wird die strukturelle Komplexität der Analysebäume sein und günstigenfalls wird auch die Qualität der Übesetzung entsprechend anwachsen.

Leider ist jedoch die strukturelle Komplexität als geeignetes Strukturmaß auf die spezielle Domäne der Maschinellen Übersetzung beschränkt. Ist man dagegen an einer semantischen Beschreibung, z.B. einer logischen Form, interessiert, so sind partiell strukturierte Bäume nur von geringem Nutzen. Darüberhinaus leidet die Baumtransformation an genau denselben Nachteilen, wie auch das klassische Chart-Parsing: Es existiert keine (zumindest probabilistisch) vorhersagbare Abhängigkeit zwischen der zu erwartenden Ausgabequalität und der dafür aufzuwendenden Analysezeit.

Diese Schwierigkeit scheint prinzipieller Natur zu sein und resultiert aus einer ganz spezifischen Eigenschaft des Parsing-Problems. Demnach ist das Parsing für natürliche Sprachen durch einen relativ unstrukturierten Suchraum charakterisiert, der zudem auch erst während des Parsing-Prozesses erst schrittweise aufgebaut werden muß. Im Gegensatz zu anderen bekannten Suchproblemen (z.B. VITERBI-Suche im Bereich der Spracherkennung) kann weder die Breite noch die Tiefe des Suchraums noch vor dem Start des Parsings abgeschätzt werden. So könnte etwa eine Technik wie das *iterative deepening* eingesetzt werden, um die Ausgabequalität eines Baumtransformationsparsers zu beeinflussen, allerdings wird dadurch das Performanzprofil keinesfalls besser vorhersagbar. Obwohl offensichtlich ein individuelles (d.h. instanzenspezifisches) Performanzprofil angenommen werden kann, läßt es sich schlecht auf ganze Klassen von Eingabesätzen generalisieren.

Diese Beobachtung legt dann aber einen Begriff von Anytime-Verhalten nahe, der von der Vorhersagbarkeit des Performanzprofils eines Moduls unabhängig ist. Daher soll hier im folgenden eine Unterscheidung zwischen Algorithmen mit strengem und solchen mit schwachem Anytime-Verhalten eingeführt werden. Eine Systemkomponente erfüllt die *strenge Anytime-Eigenschaft* wenn für einen bestimmten Qualitätsparameter ein allgemeines Performanzprofil existiert und bereits vor der Berechnung bekannt ist. *Schwache Anytime-Algorithmen* besitzen ebenfall ein Performanzprofil, da dieses aber instanzenspezifisch ist, läßt es sich nicht im voraus abschätzen und erlaubt daher keine Vorhersage des zu erwartenden Qualitätsniveaus.

Aufgrund dieser Unterscheidung läßt sich die Baumtransformationsanalyse als unterbrechbarer schwacher Anytime-Algorithmus einordnen. Die Verarbeitung kann zu jedem beliebigen Zeitpunkt abgebrochen werden, wobei ein späterer Abbruch ein besseres Übersetzungsresultat erwarten läßt.

Auch für den Begriff des Vertragsalgorithmus kann man im Bereich der schwachen Anytime-Algorithmen eine sinnvolle Interpretation angeben, wobei wiederum eine minimale Verarbeitungszeit garantiert werden muß, um brauchbare Resultate zu erzeugen. Schwaches Anytime-Verhalten liegt dabei immer dann vor, wenn ein Modul in der Lage ist, seine internen Verarbeitungsprozesse so zu optimieren, daß er in dem zur Verfügung stehenden Zeitintervall das bestmögliche Qualitätsniveau erreicht. Dies erfordert eine Art von Scheduling-Mechanismus, der eine dynamische Kosten-Nutzen-Analyse bezüglich der bisher ermittelten Teilergebnisse und der noch zur Verfügung stehenden Zeit realisiert.

Weder die traditionelle Chart-Analyse, noch der Baumtransformationsansatz scheinen sich auf einfache Weise mit den Bedingungen für einen schwachen Anytime-Algorithmus in Übereinstimmung bringen zu lassen. Im allgemeinen Fall treten ja bereits erhebliche Schwierigkeiten auf, wenn über eine optimale Fortsetzung der Verarbeitung entschieden werden soll, um eine bestimmte Analyse überhaupt rechtzeitig abschließen zu können. Vom Scheduling-Mechanismus werden dann Lösungen für ein zweidimensionales Entscheidungsproblem erwartet:

- Welche Folge von inaktiven Kanten in der Chart (bzw. welches Baumfragment im Transformationsansatz) erscheint am erfolgversprechendsten im Hinblick auf die Anwendung der nächsten Regel?

- Welche Regel in der Grammatik sollte gewählt werden, um eine Teilanalyse fortzusetzen?

Offensichtlich sind die dafür notwendigen Heuristiken nicht ohne weiteres verfügbar. Sogar nach einer erfolgreichen Regelanwendung besteht kaum eine Möglichkeit einigermaßen verläßlich einzuschätzen, ob dies ein Beitrag in Richtung auf das angestrebte Gesamtresultat war. Es ist durchaus nicht überraschend, daß derzeitige Parsing-Systeme überwiegend auf rein kombinatorischen Algorithmen basieren, deren Zeitverhalten schwer vorhersagbar ist. Angesichts dieser Schwierigkeiten erscheint es durchaus naheliegend, zur Realisierung schwacher Anytime-Vertragsalgorithmen radikal andersartige Berechnungsverfahren in Betracht zu ziehen.

4.3 Parsing als Bedingungsüberprüfung

Lösungsansätze zur Bedingungsüberprüfung in Constraint-Netzen stellen innerhalb der Verfahren zum kombinatorischen Problemlösen einen gewissen Sonderfall dar, da sie bereits aufgrund ihrer elementaren Funktionsprinzipien die Eigenschaft eines allmählichen Leistungsverfalls unter Zeitbeschränkungen mitbringen. In einem Suchraum, der durch die Zuweisung von (endlichen) Domänen zu einer endlichen Anzahl von Variablen $V_i = \{x | x \in Dom(V_i)\}$ gegeben ist, wird eine Lösung gewünscht, die gleichzeitig alle Bedingungen aus einer Constraintmenge C erfüllt. Constraints können damit als Bedingungsnetz interpretiert werden, in dem sich Werterestriktionen ausbreiten. Die

Berechnungsprozedur verfügt zu jedem beliebigen Zeitpunkt über all diejenigen Lösungen die hinsichtlich der bereits überprüften Constraints noch konsistent sind. Im besonderen enthält das Bedingungsnetz - unter anderem - auch all die global konsistenten Lösungen für das gestellte Problem.

Verfahren zur Bedingungsüberprüfung wurden erstmalig von MARUYAMA [5] auf das Parsingproblem angewandt. Grundlage dafür sind lokale Bedingungen für die zulässigen Dependenzstrukturen einer Äußerung, wobei Wortformen $w_i \in W$ durch andere Wortformen durch Dominanzrelationen $l_i \in L$ modifiziert werden.

Eine potentielle Modifikation eines Knotens im Abhängigkeitsbaum ist dann ein geordnetes Paar aus einem dominierenden Knoten und einer Dominanzrelation. Diese Paare bilden die möglichen Werte für die Bedingungsüberprüfung $V_i = \{p|p \in W \times L\}$. Der aktuelle Stand der Analyse ist dann stets durch die Menge aller verbleibenden Relationen gegeben, durch die eine Wortform eine andere modifizieren kann. Constraints $c \in C$ sind Relationen, die über Wertezuweisungen für eine beliebige Teilmenge von Variablen definiert sind: $c \subset V_m \times \ldots \times V_n$. Im Interesse einer handhabbaren Implementation sollten allerdings nur lokale (d.h. unäre oder binäre) Constraints Verwendung finden. Bezeichnet $pos(x)$ den Positionsindex eines Knotens, $mod(x)$ seinen Modifizierer, $lab(x)$ die Dominanzrelation und $cat(x)$ die Kategorie der einem Knoten zugeordneten Eingabewortform[1], so beschreibt das einstellige Constraint

$$cat(x) = \mathrm{D} \rightarrow (lab(x) = \mathrm{DET} \wedge cat(mod(x)) = \mathrm{N} \wedge pos(x) < pos(mod(x)))$$

die Tatsache daß ein Artikel (D) ein rechts von ihm stehendes Nomen (N) mit der Dominanzrelation DET modifizieren kann [5]. Die meisten einstelligen Constraints beschränken die Menge möglicher Variablenbindungen nicht in dem üblichen Sinn, sondern *lizensieren* statt dessen einen bestimmten Anfangszustand des Bedingungsnetzes. Die gegenseitige Verträglichkeit von Wertezuweisungen kann dann mit Hilfe zweistelliger Constraints formuliert werden, wie sie etwa mit der Bedingung für die Verbzweitstellung im deutschen Hauptsatz gegeben sind:

$$(mod(x) = mod(y) \wedge lab(mod(x)) = \mathrm{V} \wedge pos(x) < pos(mod(x))$$

> *"zwei Wortformen die das Hauptverb modifizieren, dürfen nicht beide links vom Verb angeordnet sein"*

Ein anderes binäres Constraint wäre etwa die Projektivitätsbeschränkung, die üblicherweise für Dependenzstrukturen angenommen wird [5]

$$pos(mod(x)) < pos(y) < pos(x) \rightarrow pos(mod(x)) \leq pos(mod(y)) \leq pos(x).$$

Durch die Anwendung derartiger Constraints auf die Modifikationsmengen an den Netzknoten, lassen sich bestimmte Wertekombinationen ausschließen, wodurch sich der Suchraum sukzessive verkleinert. Sind genügend Constraints verfügbar, so wird schließlich ein Zustand erreicht, in dem jeder Knoten (mit Ausnahme des Spitzenknotens) genau einen anderen modifiziert, so daß eine eindeutige Strukturbeschreibung für den Eingabesatz gegeben ist.

[1] Es wird nur eine Rollenmarkierung pro Wortform berücksichtigt. Der Ansatz kann jedoch sehr leicht auf eine mehrdimensionale Domonanzrelation verallgemeinert werden.
Die Behandlung der Positionsindizees unterscheidet sich geringfügig von der in [5], um später eine Übertragung auf den generelleren Fall von nicht-sequentiellen Eingabestrukturen zu erleichtern.

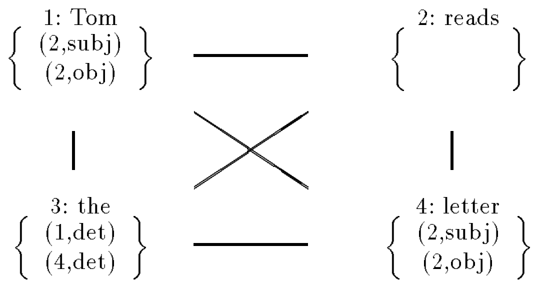


Abbildung 4.3: Der Inhalt eines Bedingungsnetzes vor der Anwendung von Anordnungsconstraints

Im Gegensatz zur Chart-Analyse ist Parsing als Bedingungsüberprüfung nicht mehr eine Prozedur, die der Zwischenergebnisstruktur streng monoton neue Teilergebnisse hinzufügt, sondern statt dessen die Menge der Strukturierungsmöglichkeiten monoton einschränkt.

Zusätzlich zu der Tatsache, daß sich Verfahren der Bedingungsüberprüfung ohne Schwierigkeiten parallelisieren lassen, verfügen sie noch über einen weiteren entscheidenden Vorteil: Bereits auf der Grundlage sehr einfacher formaler Parameter (z.B. der Größe der Wertemengen an den einzelnen Knoten des Bedingungsnetzes) wird eine Bewertung des aktuellen Analysezustands und -fortschritts möglich. Darüberhinaus stehen eine Reihe einfacher aber ziemlich leistungsfähiger Heuristiken zur Verfügung, um solche Bereiche im Bedingungsnetz auszuwählen, wo durch Constraint-Anwendung der Suchraum am wirkungsvollsten eingeschränkt werden kann.

Unter dem Gesichtspunkt einer Anytime-Analyse ist dieser Vorteil von entscheidender Bedeutung. Hiermit wird zum erstenmal eine dynamische Zuordnung der Verarbeitungskapazität einer Komponente ermöglicht. Es kann erreicht werden, daß sich die Analyse genau auf diejenigen Bereiche im Bedingungsnetz konzentriert, wo eine Disambiguierung am dringendsten erscheint. Das Scheduling kann allerdings noch weiter verbessert werden, wenn eine bestimmte Ordnung über der Constraint-Menge bezüglich ihrer Kapazität zur Suchraumbeschränkung angenommen wird. Auch hier stellt sich wieder ein zweidimensionales Entscheidungsproblem: Die Prozedur versucht eine optimale Reihenfolge von Constraint-Anwendungen zu finden, so daß der globale Zustand der Netzkonsistenz mit minimalem Aufwand erreicht wird. Im Gegensatz zur Tradition der Unifikationgrammatiken werden die verfügbaren Constraints nicht gleichzeitig aktiviert, sondern der Parser entscheidet selektiv, wo welche Constraints im aktuellen Analysezustand am wirkungsvollsten angewendet werden können

Für den Einsatz in einem interaktiven System zur Maschinellen Übersetzung schlägt Maruyama [5] das Prinzip einer Kerngrammatik vor. Die Analyse startet mit einer minimalen aber verhältnismäßig allgemeinen Constraint-Menge und fügt zu dieser nur dann stärker spezifische Constraints hinzu, falls dies für die Auflösung der verbleibenden Mehrdeutigkeiten unumgänglich ist.

Die verwendeten Constraints können syntaktischer, semantischer oder auch domänenspezifischer Natur sein, wobei keine feste Reihenfolge der Constraintanwendungen vorherbestimmt ist. Deshalb können domänenspezifische Constraints, die üblicherweise wesentlich restriktiver sind als die generellen Constraints der Grammatik, sofort zum Einsatz kommen, sobald alle ihrer Anwendungsbedingungen erfüllt sind. Falls die Disambiguierung in bestimmten Fällen bereits auf der Basis von rein domänenspezifischem Wissen erfolgreich war, kann auf die Anwendung allgemeiner syntaktischer Constraints ganz verzichtet werden. Dadurch werden bestimmte Arten von ungrammatischen Konstruktionen unter Umständen auch ohne zusätzlichen Aufwand akzeptiert. Hinzu kommt, daß Constraints nicht notwendigerweise statische Wissenskomponenten sind. Sehr spezifische Constraints können jederzeit auch von anderen Systemkomponenten (etwa einem dynamischen Domänen-, Diskurs- oder Nutzermodell) angefordert werden und sind daher im Hinblick auf interaktive Systemstrukturen besonders interessant (c.f. [7]).

4.4 Qualitätsoptimierung

Eine Prozedur zum Parsing durch Bedingungsüberprüfung geht stets von einer strukturellen Beschreibung mit maximaler Mehrdeutigkeit aus und versucht dann die Anzahl verschiedener Lesarten möglichst weitgehend zu reduzieren. Demnach bietet sich der Grad an verbleibender Mehrdeutigkeit als Maß zur Beurteilung des Analysefortschritts an. Dieses Maß kann durchaus zur Ermittlung einer geeigneten Reihenfolge von Constraint-Anwendungen herangezogen werden, um auf dieser Grundlage die Zeit bis zum Erreichen einer eindeutigen Interpretation zu minimieren. Andererseits sind aber erhebliche Zweifel angebracht, ob der Grad der verbleibenden Mehrdeutigkeit auch ohne weiteres als direktes Kriterium für die Ausgabequalität verwendet werden kann. Prinzipiell muß man natürlich davon ausgehen, daß nur weitgehend disambiguierte Strukturbeschreibungen als sinnvolle Grundlage für eine weitere Verarbeitung geeignet sind.

Selbst in Fällen von Zeitdruck oder beim Fehlen allgemeingültiger Constraints kann eine eindeutige Interpretation immer durch die Anwendung heuristischer Constraints oder in Extremfällen gar durch eine *ad hoc* Auswahl erreicht werden. Heuristische Constraints basieren etwa auf auf groben Annahmen, die den Suchraum wenigstens in den besonders häufigen Fällen reduzieren. So könnte man etwa eine Heuristik für die Erststellung des Subjekts im Deutschen auf die folgende Weise spezifizieren

$$cat(x) = \mathrm{V} \wedge mod(y) = x \wedge lab(y) = \mathrm{SUBJ} \rightarrow pos(y) < pos(x)$$

Dies ist ein verhältnismäßig rigides Constraint, das alle anderen Konstituenten eines Satzes unwiderruflich von der Möglichkeit zur Topikalisierung ausschließt.

Eine andere bekannte Heuristik ist das Prinzip des minimalen Attachments, das z.B. kürzere Dependenzbeziehungen gegenüber längeren präferiert. Heuristiken dieser Art beschränken nicht mehr den Spielraum für die Konsistenz individueller Strukturrepräsentationen, sondern definieren vielmehr *Präferenzen* auf der Basis eines Lesartenvergleichs. Daher erweist es sich auch als schwierig, diese Bedingungen in der üblichen Art als rein logische Constraints zu formulieren. Allerdings lassen sich solche Präferenzen durch (nichtmonotone) Regeln beschreiben, die den Suchraum direkt manipulieren und präferenzlose Modifikationsvarianten einfach aus den Wertemengen entfernen.

$$\begin{aligned} & mod(x) = y \wedge mod(x) = z \wedge y \neq z \wedge pos(y) < pos(z) < pos(x) \\ & \quad \Rightarrow \mathrm{DELETE}(mod(x) = y) \end{aligned}$$

"Modifiziert ein Nomen x zwie andere Knoten (y und z) gleichzeitig wird die Modifikation des weiter entfernten Knotens unterdrückt."

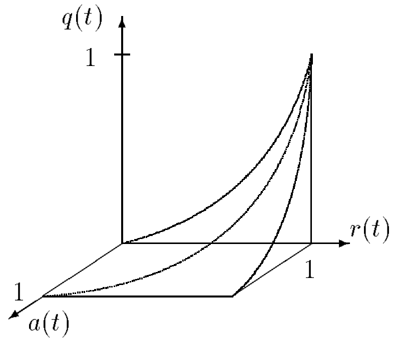


Abbildung 4.4: Qualitätsmaß für die Disambiguierung mit heuristischen Constraints

Das gleiche Resultat kann auch durch ein dynamisches Constraint erreicht werden, das eine obere Schranke für die Distanz zwischen Modifikator und Modifikandum definiert

$$\begin{aligned}&mod(x) = y \wedge mod(x) = z \wedge y \neq z \wedge pos(y) < pos(z) < pos(x)\\ &\quad\rightarrow pos(x) < pos(mod(x)) + n\end{aligned}$$

wobei n direkt proportional zur verbleibenden Zeit T sein sollte. Auf ganz ähnliche Weise können auch andere dynamische Heuristiken (z.B. die Unterordnung eines Artikels) erfaßt werden.

Heuristische Constraints lassen sich grob nach ihrer (geschätzten) Zuverlässigkeit einteilen. Damit wird das Auswahlproblem dreidimensional:

1. unterschiedliche Knoten im Bedingungsnetz, die einen unterschiedlichen Grad an Mehrdeutigkeit aufweisen,

2. unterschiedliche Constraints, für die ein unterschiedliches Potential zur Mehrdeutigkeitsreduktion vermutet wird und

3. unterschiedliche Constraints mit einem unterschiedlichen Grad an Zuverlässigkeit.

Um die Verarbeitung in einer möglichst optimalen Weise zu steuern, müssen alle drei Kriterien mit dem Nützlichkeitsprofil des Parsing-Moduls abgeglichen werden: So lange Zeitbeschränkungen noch als vernachlässigbar erscheinen, wird eine generelle Lösung für das Parsing-Problem angestrebt, wobei die restriktiveren Constraints präferiert zur Anwendung kommen. Erst ein wachsender Zeitdruck in Verbindung mit einem verhältnismäßig hohem Grad an restlicher Mehrdeutigkeit rechtfertigt die Aktivierung heuristischer Constraints, um dadurch die Analyse zu beschleunigen.

Ein Qualitätsmaß für einen schwachen Anytime-Vertragsalgorithmus sollte so definiert werden, daß es den grundsätzlichen Gegensatz zwischen der verbleibenden Mehrdeutigkeit und der Zuverlässigkeit der verwendeten Constraints widerspiegelt. Sei $a(t)$ die verbleibende Mehrdeutigkeit, normalisiert mit ihrem initialen Wert, und $r(t)$ der mittlere Grad an Zuverlässigkeit beim Parsing, die beide für das Intervall $\langle 0, 1\rangle$ definiert sind. Ein geeignetes Qualitätsmaß wäre dann etwa durch

$$q(t) = \frac{e^{r(t)[1-a(t)]} - 1}{e - 1}$$

gegeben. Diese Definition erfordert für Ergebnisse mit maximaler Qualität sowohl eine geringe Restmehrdeutigkeit als auch eine hohe Zuverlässigkeit. Demgegenüber ist eine geringe Qualität durch hohe Mehrdeutigkeit und/oder geringe Zuverlässigkeit charakterisiert.

Die Verwendung heuristischer Constraints im Verarbeitungsprozeß bringt den Vorteil mit sich, daß auch beim Vorliegen sehr strenger Zeitbeschränkungen noch partiell nützliche Resultate erzeugt werden können. Obwohl sich ein Qualitätsmaß kaum für individuelle Äußerungen angeben läßt, kann jedoch das gewünschte schwache Anytime-Verhalten zumindest in einem allgemeinen probabilistischen Sinne erwartet werden.

Allerdings treten in diesem Zusammenhang auch einige Probleme auf, die geeignet berücksichtigt werden müssen

1. Heuristische Constraints stehen mit Sicherheit im Widerspruch zu anderen Constraints, die bereits zuvor oder aber an anderer Stelle im Bedingungsnetz angewendet wurden. Daher ist die Verwendung von Heuristiken nur in engen lokalen Bereichen zur Lösung ganz spezieller Disambiguierungsprobleme möglich. Auf jeden Fall müssen globale Inkonsistenzen vermieden werden, da diese die Ermittlung kompletter Strukturbeschreibungen verhindern können.

2. Die Anwendung heuristischer Constraints ist grundsätzlich irreversibel. Heuristische Entscheidungen sollte daher nur dann getroffen werden, falls auf anderem Wege keine eindeutige Beschreibung mehr erreicht werden kann.

3. Der additive Qualitätszuwachs für zwei aufeinanderfolgende Zeitintervalle - eine fundamentale Eigenschaft gewöhnlicher Anytime-Algorithmen - ist bei der Verwendung heuristischer Constraints nicht mehr gegeben. Sei $\Delta q(\langle t_1, t_2 \rangle)$ der (meßbare) Qualitätszuwachs eines Vertragsalgorithmus im Zeitintervall $\langle t_1, t_2 \rangle$, wobei $t_1 \neq 0$ als Fortsetzung der Verarbeitung nach Ablauf der initialen Vertragszeit t_1 zu interpretieren ist. Für starke Anytime-Algorithmen gilt dann stets die folgende Gleichung

   $$\Delta q(\langle 0, t_1 \rangle) + \Delta q(\langle t_1, t_2 \rangle) = \Delta q(\langle 0, t_2 \rangle)$$

   Sie verliert jedoch ihre Gültigkeit, wenn zur Einhaltung der initialen Vertragszeit t_1 heuristische Constraints herangezogen werden mußten. Dann kann im allgemeinen Fall nur noch mit einem reduzierten Qualitätsanstieg im Anschluß an die Unterbrechung gerechnet werden und für hinreichend großes $t_2 - t_1$ gilt schließlich

   $$\Delta q'(\langle 0, t_1 \rangle) + \Delta q'(\langle t_1, t_2 \rangle) < \Delta q(\langle 0, t_2 \rangle).$$

   Diese Reduktion des Wachstums wird durch irreversible Einschränkungen in der initialen Verarbeitungsphase t_1 hervorgerufen, die verhindern, daß die Analyse jemals wieder eine nahezu optimales Niveau erreichen kann. In der Definition des oben angegebenen Qualitätsmaßes spiegelt sich diese Tatsache dadurch wieder, daß das Erreichen der maximalen Qualität mit einer mittleren Zuverlässigkeit $\bar{r} < 1$ unmöglich ist.

4.5 Parsing gesprochener Sprache

Als Mindestanforderung muß das Parsing gesprochener Sprache sicherlich zwei entscheidenden Phänomenklassen Rechnung tragen

- der inhärente Unsicherheit partieller Erkennungsergebnisse und
- dem Fehlen verläßlicher Grenzmarkierungen im Sprachsignal.

Angesichts der Unsicherheit an der Schnittstelle zwischen Spracherkennung und Sprachverarbeitung versucht man üblicherweise unzuverlässige Entscheidungen durch den Einsatz von Wortgittern zu vermeiden. Jede Worthypothese wird durch ihre zeitlichen Begrenzungen markiert und mit einem geschätzten Konfidenzwert versehen.

Hieraus ergibt sich als erste Konsequenz, daß alle Anordnungsconstraints in den oben angegebenen Beispielen von Positionsindizes auf Relationen über Zeitintervallen verallgemeinert werden müssen: Positionsanordnungen werden durch entsprechende Intervallpräzedenzen und Gleichheitsconstraints in Überlappungsbedingungen umgewandelt. Gleichzeitig werden zusätzliche Constraints erforderlich, die bestimmte abnorme Dependenzstrukturen ausschließen sollen. Weil Constraints ausschließlich über inkonsistenten Modifikationmöglichkeiten formuliert werden können, müssen Überlappungsbeschränkungen ebenfalls auf der Basis von Modifikationsrelationen angegeben werden:

$$\begin{aligned} & mod(a) = b \wedge mod(c) = d \rightarrow \neg(overlap(a,b) \\ & \quad \vee overlap(a,c) \vee overlap(a,d) \vee \ldots \vee overlap(c,d)) \end{aligned}$$

"Für zwei beliebige Modifikationsrelationen dürfen sich die davon betroffenen Knoten zeitlich nicht überlappen."[2]

Dieses Constraint verhindert jedoch nur, daß sich solche Knoten überlappen können, die durch höchstens zwei Dependenzrelationen miteinander verbunden sind. Offensichtlich ist diese Bedingung jedoch nicht hinreichend, denn Modifikation ist eine transitive Relation, so daß hier eigentlich die vollständige transitive Hülle einbezogen werden müßte. Um hier zu wirksamen Einschränkungen zu kommen, wird man (partielle) Linearisierungen von Dependenzstrukturen in Betracht ziehen müssen. Das erlaubt es dann, die Überprüfung von Überlappungsbedingungen von Einzelwortformen auf Teilbäume zu erweitern, bringt aber gleichzeitig das ernsthafte Risko einer extremen Ausweitung des Suchraums mit sich. Aus diesem Grunde sollte dies auch nur für bereits weitgehend disambiguierte Dependenzstrukturen in Betracht gezogen werden.

Genau hier liegt die entscheidende Schwierigkeit beim Wortgitterparsing mit Bedingungsüberprüfung: Wesentliche und wirksame Constraints können erst dann zur Anwendung kommen, wenn der Suchraum bereits bis zu einem gewissen Grad eingeschränkt worden ist. Man kann versuchen, diesed Problem durch den Rückgriff auf extrem restriktive (üblicherweise domänenspezifische) Constraints zu lösen, um dann einzelne "Inseln der Sicherheit" sukzessiv zu erweitern. Angesichts der gewaltigen Vielfalt von Modifikationsmöglichkeiten wird es jedoch die eindeutige Ausnahme bleiben,

[2]Drei wichtige Spezialfälle erhält man jeweils für $b = c$, $b = d$ bzw. $(a = c \wedge b = d)$ sowie der Annahme $overlap(x,x) = \text{FALSE}$.

daß eine Modifikationsrelation mit hoher Sicherheit allein auf der Grundlage von Verträglichkeitsbedingungen ausgeschlossen werden kann.

Als zusätzliche Informationquelle bieten sich naturgemäß probabilistische Schätzwerte, u.a. in den folgenden Bereichen, an:

- Bigram-Statistiken für Wortformsequenzen
- Wahrscheinlichkeitsschätzungen für Dominanzmöglichkeiten
- Konfidenzwerte für die zu analysierenden Worthypothesen

Anstelle einer binären Entscheidung, muß die Verträglichkeit von zwei Modifikationshypothesen nunmehr als Fuzzy-Wert beschrieben werden. Heuristische Constraints erhöhen dann die Bewertung für die präferierten Dominanzrelation, während der Wert für die eher unwahrscheinlichen Varianten reduziert werden muß. Modifikationshypothesen mit einer geringen Bewertung können dann unter Zeitdruck ausgeschlossen werden.

Die zweite wesentliche Besonderheit bei der Verarbeitung gesprochener Sprache ist das Fehlen geeigneter Grenzmarkierungen im Sprachsignal. Das Parsing muß deshalb als zeitsynchrone Analyseprozedur realisiert werden, so daß die Knotenanzahl im Bedingungsnetz - obwohl zu jedem beliebigen Zeitpunkt endlich - nicht mehr von vornherein bekannt ist. Das Netz wird durch neu hinzukommende Worthypothesen ständig erweitert und liefert als Resultat diejenigen Knoten, die den aktuelle gegebenen Zeithorizont verlassen. Für jeden neu hinzukommenden Knoten werden alle durch die unären Constraints lizensierten Modifikationsmöglichkeiten etabliert und die Bedingungsübertragung versucht erneut, die Zahl der konsistenten Lesarten auf eine eindeutige Interpretation zu reduzieren. Um der Anytime-Forderung Rechnung zu tragen, muß die eindeutige Interpretation für jeden Knoten spätestens erreicht werden, noch ehe das verfügbare Zeitintervall abgelaufen ist. Auch hierbei wird die Analyse wieder durch den verbleibenden Grad der Mehrdeutigkeit und den aktuellen zeitlichen Nachlauf gesteuert.

4.6 Schlußfolgerungen

Techniken der Bedingungsüberprüfung sind gut geeignet, um zumindest im Fall deterministischer Eingabedaten ein schwaches Anytime-Verhalten zu realisieren. Da der Ansatz eine explizites Abwägen zwischen den verfügbaren und den erforderlichen Ressourcen für die Disambiguierung unterstützt, werden Parsing-Prozeduren möglich, die sich dynamisch an externe Zeitrestriktionen anpassen können, wie sie für die Verarbeitung gesprochener Sprache typisch sind. Wissen aus sehr unterschiedlichen Quellen (Syntax, Semantik, Diskursgeschichte, Diskursdomäne, Nutzerwissen und -interessen, ...) kann in koordinierter Weise zusammengebracht werden. Das Verfahren beschränkt sich nicht auf statische (d.h. universell gültige) Constraints, sondern kann auch dynamische (d.h. nur lokal gültige) Wissenskomponenten einbeziehen. Das gilt auch in solchen Bereichen, in denen traditionell eher eine statische Sicht vorherrschend ist. So ist es beispielsweise möglich, die erhöhte Wahrscheinlichkeit für bestimmte Topikalisie

Eine relativ einfaches, lokal berechenbares Maß gestattet die Ermittlung derjenigen Teile des Bedingungsnetzes, für die eine weitere Disambiguierung besonders dringend erscheint. Kombiniert man dies mit einer Abschätzung der restriktiven Kapazität der einzelnen Constraints, wird eine dynamische Verteilung der Systemressourcen möglich. Die wirksamsten Constraints werden vorrangig angewendet, um so ein möglichst optimales Zeitverhalten zu erreichen. Probabilistische Verfahren können ohne Schwierigkeiten integriert werden.

Es gibt durchaus berechtigten Grund zu der Annahme, daß sich der Ansatz im Prinzip auch auf die Behandlung gesprochener Sprache ausdehnen läßt. Dies erfordert in erster Linie die Möglichkeit zum Parsing in dynamisch erweiterbaren Bedingungsnetzen. Derzeit dürfte wohl das Hauptproblem darin bestehen, daß eine Tradition zum Kodieren lingistischer Information als lokale Constraints völlig fehlt. Ein endgültiges Urteil über die Realisierbarkeit des Ansatzes wird daher nicht möglich sein, so lange nicht wenigsten ein nichttriviale Fragment einer natürlichen Sprache beschrieben und getestet worden ist, um zumindest die wichtigsten Fragen zu klären

- In welchem Maße kann sprachspezifisches Wissen durch streng lokale Constraints über Modifikationsmöglichkeiten dargestellt werden?
- Wie wirksam ist die restriktive Kraft der Grammatik angesichts der typischen Erkennungsunsicherheit, die sich in einem Wortgitter widerspiegelt?
- Läßt sich die restriktive Kraft eines einzelnen Constraints so zuverlässig abschätzen, daß ein wirksames Scheduling der Systemressourcen möglich wird?

Ungünstigerweise führt jedoch der generelle Trend in der derzeitigen Computerlinguistik genau in die entgegengesetzte Richtung. Im Rahmen von unifikationsbasierten Grammatiken werden Constraints mit ständig steigender Komplexität eingesetzt, um Kombinierbarkeitsbedingungen und strukturbildende Operationen mit Hilfe eines einheilichen Formalismus zu beschreiben. Gerade diese Komplexität ist es jedoch, die es für den Systementwickler sehr schwierig macht

- die restriktive Kraft von Constraints abzuschätzen,
- diejenigen Teile eines Constraints zu ermitteln, die für die Lösung eines speziellen Disambiguierungsproblems ausreichend sind,
- diejenigen Teile eines Constraints zu ermitteln, die bei Bedarf durch stärkere (heuristische) Constraints ersetzt werden können,
- verläßliche Heuristiken zur Ermittlung derjenigen Teilergebnisse zu finden, die im Hinblick auf das Endergebnis am vielversprechendsten erscheinen,
- Algorithmen für eine inkrementelle Analyse zu entwerfen, in denen partielle Constraints möglichst frühzeitig angewendet werden und die Teilergebnisse erst dann erweitert werden, wenn zusätzliche Daten vorliegen.

Da natürlich andererseits aber auch die Vorzüge des unifikationsbasierten Ansatzes für das Schreiben kompakter und verständlicher Grammatiken nicht in Frage gestellt werden sollen, wird eine der interessantesten Fragen darin bestehen, inwieweit es möglich ist, die für die Bedingungsüberprüfung erforderlichen lokalen Constraints (halb-)automatisch aus den komplexen Merkmalstrukturen abzuleiten, wie sie für die unifikationsbasierte Verarbeitung natürlicher Sprache üblich sind.

Literaturverzeichnis

[1] Mark Boddy and Thomas L. Dean. Solving time-dependent problems. In *Proceedings 11th International Joint Conference on Artificial Intelligence*, Detroit, MI, 1989.

[2] Mark Boddy and Thomas L. Dean. Deliberation scheduling for problem solving in time-constrained environments. *Artificial Intelligence*, 67(2):245–285, 1994.

[3] Ch. Boitet, P. Guillaume, and M Quezel-Ambrunaz. Implementation and conversational environment of ariane-78. In *Proc. Coling '82*, pages 19–28, Prague, 1982.

[4] Günther Görz and Marcus Kesseler. Anytime algorithms for speech parsing? In *Proceedings 15th International Conference on Computational Linguistics, Coling '94*, 1994.

[5] H. Maruyama. Structural disambiguation with constraint propagation. In *Proc. 28th Ann. Meeting of the ACL*, pages 31–38, 1990.

[6] Wolfgang Menzel. Parsing of spoken language under time constraints. In T. Cohn, editor, *Proceedings 11th European Conference on Artificial Intelligence*, pages 560–564, 1994.

[7] C. Pyka. Management of hypotheses in an integrated speech-language architecture. In *Proc. ECAI '92*, pages 558–560, Vienna, 1992.

[8] S. J. Russell and S. Zilberstein. Composing real-time systems. In *Proc. IJCAI '91*, pages 212–217, Sidney, 1991.

[9] W. Wahlster. Verbmobil: Translations of face-to-face dialogs. In *Proc. MT Summit IV*, pages 127–135, Kobe, 1993.

Kapitel 5

Die inkrementelle Bildung von Diskursrepräsentationsstrukturen über einer Chart

Ingrid Fischer
Lehrstuhl für Programmiersprachen und ihre Compiler
Universität Erlangen–Nürnberg
Martensstraße 3
D–91058 Erlangen
email: *idfische@informatik.uni-erlangen.de*

Zusammenfassung

In diesem Beitrag wird die Implementierung eines Parsers beschrieben, der Syntax und Semantik eines Satzes inkrementell und parallel konstruiert. Als syntaktische Komponente wird ein PATR–2 ähnlicher Formalismus eingesetzt, die semantische Basis bildet die Diskursrepräsentationstheorie (DRT). Für die Konstruktion der Diskursrepräsentationsstrukturen (DRS) wird ein besonderer Algorithmus verwendet, der auf dem λ–Kalkül basiert (siehe [9], [1] und [11]) . Nach der Anpassung dieses Algorithmus an die parallele Konstruktion von Syntax und Semantik, können verschiedene Phänomene des Deutschen wie Tempora, ambige Sätze, Negation und Verben als Verbvalenzen analysiert werden.

Abstract

This paper presents an implementation of a parser, which constructs syntax and semantics of a sentence incrementally in parallel. The parser consists of a syntactic component similar to PATR–2 and a semantic component which uses Discourse Representation Theory (DRT). To construct the Discourse Representation Structures (DRS) a special algorithm based on λ–calculus is used ([9], [1], [11]). After adapting this algorithm to the parallel construction of syntax and semantics, the analysis of different phenomena in German like tempora, ambigous sentences, negation and verbs as valences of verbs is possible.

5.1 Einleitung

Anfang der Achtziger Jahre wurde von H. Kamp die DRT entwickelt, deren grundlegende Datenstruktur, die Diskursrepräsentationsstruktur (DRS), aus zwei Teilen besteht: einer Menge von Diskursreferenten, dem Universum der DRS, und einer Menge von Bedingungen. Die Diskursreferenten können sowohl von Nominalphrasen als auch von Verbalphrasen eingeführt werden, die Bedingungen sind entweder Prädikationen über den Diskursreferenten oder enthalten wiederum DRSen. In dem von Kamp vorgeschlagenen Konstruktionsalgorithmus werden diese DRSen aus einem Phrasenstrukturbaum top–down konstruiert, was voraussetzt, daß die syntaktische Analyse eines Satzes vor der semantischen durchgeführt wird.

Ein häufiger Kritikpunkt an Kamps Algorithmus ist, daß die Konstruktion der DRSen nicht kompositional ist. Andere Formalismen überwinden das Problem der Kompositionalität, indem sie den λ–Kalkül zur Konstruktion verwenden ([9], [1], [11]). Aber auch diese Verfahren erwarten als Eingabe einen Phrasenstrukturbaum oder eine Merkmalsstruktur, also wiederum die syntaktische Analyse vor der semantischen. Die Nachteile dieser Trennung von Syntax und Semantik sind offensichtlich. Zum einen können Ergebnisse der Semantik nicht frühzeitig zur Disambiguierung auf syntaktischer Ebene herangezogen werden, zum anderen entspricht sie nicht dem menschlichen Verarbeitungsmodell.

Da die Anwendung des λ–Kalküls aber die kompositionale Bildung von DRSen erlaubt, ist es möglich, die Konstruktionsalgorithmen an die parallele Bildung von syntaktischen und semantischen Strukturen anzupassen. Ein einfacher Schritt erlaubt dann sogar die inkrementelle konstruktion der DRSen.

Im folgenden wird zuerst das zugrundeliegende syntaktische System näher beschrieben, anschließend die Verbindung von λ–Kalkül und DRT. Danach wird das aus der parallelen Verarbeitung resultierende Grammatikformat dargestellt. Zuletzt wird die Verarbeitung einiger Phänomene des Deutschen und daraus resultierende Erweiterungen des Systems erläutert.

5.2 Der zugrundeliegende Parser

Ein PATR–2 ähnliches, merkmalsbasiertes System bildet die syntaktische Basis. Die Konstituenten– und Merkmalsstruktur eines Satzes werden mit Hilfe von Regeln gebildet, die aus einem kontextfreien Teil und aus Constraint–Gleichungen über Merkmalsstrukturen bestehen. In den Gleichungen werden die unterschiedlichen Merkmalsstrukturen der Konstituenten durch Projektionen angesprochen. Mit Hilfe der Unifikation werden die Gleichungen ausgewertet. Eine ausführliche Beschreibung dieses Systems findet sich in [2].

Als Grundlage dient ein inkrementell arbeitender aktiver Chartparser (siehe [4]). Die Kanten der Chart sind zum einen mit Kategoriesymbolen und zum anderen mit Merkmalsstrukturen markiert. Die Ausdehnung einer aktiven Kante über eine inaktive erfolgt, wenn sowohl die zugehörigen Kategoriesymbole der angewandten Regel entsprechen, als auch die Gleichungen über den Merkmalsstrukturen anwendbar sind.

5.3 Die Verbindung von λ–Kalkül und DRT

Die Kritik an Kamps Konstruktionsalgorithmus für DRSen hat zur Entwicklung anderer, kompositional arbeitender Algorithmen geführt. Die meisten dieser Ansätze verbinden den λ–Kalkül mit der DRT.[1]

Grundlage für die Benutzung des λ–Kalküls sind λ–*Abstraktion* und λ–*Konversion*. Die Beschreibung der λ–Abstraktion stimmt bei den verschiedenen Ansätzen im wesentlichen überein:

- Man spricht von einer **prädikativen DRS**, wenn über einen Diskursreferenten r abstrahiert wird, der in einer Gleichung der DRS, aber in keinem ihrer Universen auftritt. Ein Beispiel ist die λ–DRS zum Nomen *„Mann“*:

  $$\lambda x \; \boxed{mann(x)}$$

- Bei einer **partiellen DRS** wird über eine DRS abstrahiert, deren Inhalt mit dem einer zweiten Struktur vereinigt werden soll. Die Vereinigung sei durch $+$ ausgedrückt. Das folgende Beispiel zeigt die λ–DRS zum indefiniten Artikel:

  $$\lambda Q \lambda R \; \boxed{x} \; +Q(x)+R(x)$$

Zur Durchführung der λ–Konversion existieren folgende voneinander verschiedene Ansätze:

- Reyle bildet in [11] DRSen aus Merkmalsstrukturen der LFG. Zunächst wird jedes Blatt der Merkmalsstruktur mit einer (prädikativen oder partiellen) DRS assoziiert. Die λ–Konversion wird dann mit Hilfe der *funktionalen Applikation* durchgeführt. Auch bei Reyle dient das Ergebnis der syntaktische Analyse als Eingabe für die Semantikkomponente.

- Asher beschreibt in seinem Buch ([1]) ausführlich die Anwendung der DRT auf verschiedene linguistische Phänomene. Grundlage seiner Ausführungen sind dabei die Arbeiten von Reyle ([11]). Zusätzlich existieren sogenannte *linking rules*. Diese Regeln beschreiben die zugrundeliegenden syntaktischen Strukturen näher und legen fest, auf welche der DRSen und Referenten, über die abstrahiert wird, die λ–Konversion angewandt werden soll.

- Bei Pinkal ([9], [10]) werden DRSen durch Merkmalsstrukturen dargestellt. Die λ–Konversion wird als *funktionale Komposition* mit Hilfe der Unifikation durchgeführt. Allerdings greift Pinkal auf eine eingeschränkte Definition der funktionalen Komposition zurück:

  $$\phi * \psi = \lambda\vec{\sigma}.(\lambda\nu.\psi(\nu)(\vec{\sigma}))$$

  Wesentlich ist hierbei die Einstelligkeit der Funktion ψ. Praktisch bedeutet dies, daß bei der Durchführung der Komposition nur genau ein Element der Abstraktionsliste des Arguments gesättigt wird.

  Wie auch bei Reyle arbeiten bei Pinkal Syntax und Semantik sequentiell, allerdings ist die Semantikkomponente hier nicht an einen bestimmten Syntaxformalismus gebunden.

[1] Es gibt auch Ansätze, die DRSen ohne Benutzung des λ–Kalkül kompositional konstruieren, siehe [13] und [5].

Für die Anbindung an den oben beschriebenen Parser eignet sich Pinkals Ansatz am besten. Zum einen sind Merkmalsstrukturen und Unifikation bereits implementiert, zum anderen ist Pinkals Ansatz nicht abhängig von einem syntaktischen Formalismus. Im folgenden Abschnitt wird daher die von Pinkal so benannte λ–DRT näher beschrieben.

5.4 Die λ–DRT

Die Merkmalsstrukturen zur Darstellung der λ–DRSen enthalten auf oberster Ebene neben den Merkmalen *lambda* und *drs* auch das Merkmal *bdr* als Abkürzung für **bindungsfähige Diskursreferenten**. Der Wert dieses Merkmals ist eine Liste, die alle durch Pronomina eingeführten Diskursreferenten enthält. Diese Liste wird zur Auflösung von Anaphern verwendet.
Das folgende Beispiel zeigt die λ–DRSen des Nomens *„Mann“* und des indefiniten Artikels *„ein“* als Merkmalsstruktur:[2]

$$
\begin{bmatrix}
\text{lambda:} & \langle \boxed{1} \rangle \\
\text{drs:} & \begin{bmatrix} \text{dr:} \ \langle\rangle \\ \text{con:} \ \left\langle \begin{bmatrix} \text{pred: } \mathit{Mann} \\ \text{arg: } \langle \boxed{1} \rangle \end{bmatrix} \right\rangle \end{bmatrix} \\
\text{bdr:} & \langle\rangle
\end{bmatrix}
\quad
\begin{bmatrix}
\text{lambda:} & \left\langle \begin{bmatrix} \text{lambda:} \langle \boxed{6} \rangle \\ \text{drs:} \begin{bmatrix} \text{dr: } \langle \boxed{2} \rangle \\ \text{con: } \langle \boxed{3} \rangle \end{bmatrix} \end{bmatrix} \begin{bmatrix} \text{lambda:} \langle \boxed{6} \rangle \\ \text{drs:} \begin{bmatrix} \text{dr: } \langle \boxed{4} \rangle \\ \text{con: } \langle \boxed{5} \rangle \end{bmatrix} \end{bmatrix} \right\rangle \\
\text{drs:} & \begin{bmatrix} \text{dr: } \langle \boxed{6}\ \boxed{2}\ \boxed{4} \rangle \\ \text{con: } \langle \boxed{3}\ \boxed{5} \rangle \end{bmatrix} \\
\text{bdr:} & \langle\rangle
\end{bmatrix}
$$

Die funktionale Komposition wird als Unifikation von Teilen der Merkmalsstrukturen des Funktors und des Arguments realisiert:

- Das Argument wird mit dem ersten Element der λ–Liste des Funktors unifiziert.
- Die Reste der λ–Listen des Funktors und des Arguments bilden die neue λ–Liste. Dabei erscheint zuerst die Restliste des Arguments.
- Die neue DRS entspricht der DRS des Funktors. Durch die Unifikation sind die benötigten Variablen belegt worden.
- Zusätzlich werden die Listen der bindungsfähigen Diskursreferenten vereinigt.

Da im hier beschriebenen System noch keine Anaphern aufgelöst werden, wird im folgenden auf die Liste der bindungsfähigen Diskursreferenten verzichtet.

[2] Symbole wie $\boxed{1}$ entsprechen Variablen, $\langle, \rangle$ bezeichnen Sequenzen.

5.5 Die parallele Konstruktion von Syntax und Semantik

Es existieren zwei verschiedene Ansätze zur parallelen Konstruktion semantischer und syntaktischer Strukturen:

- Im **integrierten Ansatz** werden die syntaktische und semantische Analyse mit Hilfe derselben Datenstrukturen und Operationen konstruiert. Prominentestes Beispiel dieses Ansatzes ist die HPSG.[3] Diese erlaubt eine enge Kommunikation, führt aber auch zu einer zu starken Vermischung zwischen Syntax und Semantik. Ein weiterer Nachteil ist die komplexe Verarbeitung bei dieser Methode, was die Entwicklung großer Systeme erschwert.

- Im **co–descriptiven Ansatz** werden Syntax und Semantik parallel, aber mit Hilfe unterschiedlicher Formalismen konstruiert. Dies erscheint als die beste Möglichkeit, das menschliche Verarbeitungsmodell zu simulieren und effizient auf den Computer zu übertragen. Durch die unterschiedlichen Verarbeitungsformalismen kann besser auf die jeweiligen Aspekte von Syntax und Semantik eingegangen werden.

Im hier beschriebenen System wird also das zweite Verfahren angewandt: Syntax und Semantik eines Satzes werden parallel mit Hilfe von PATR–2 und λ–DRT gebildet.

Gemeinsames Gerüst beider Formalismen bildet in der Grammatik der kontextfreie Regelteil von PATR–2. Die dort verwendeten Kategorien erscheinen nicht nur als Projektionen in den syntaktischen Gleichungen, sondern auch in Operationen über DRSen. Grundlage dieser semantischen Operationen sind der einstellige Operator *id* für die Identitätsfunktion und der zweistellige Operator *compose*, der mit zwei λ–DRSen als Funktor und Argument die funktionale Komposition durchführt. Das folgende Beispiel zeigt eine solche Regel:

```
(S → NP VP

 ((NP cat)        =  np
  (VP cat)        =  vp
  (S cat)         =  s
  (NP head agr)   =  (VP head agr)
  (S head)        =  (VP head)
  (S head subj)   =  (NP head))

 ((compose NP VP)))
```

Im Gegensatz zu den Regeln ist das Lexikon zweigeteilt. Das syntaktische Lexikon enthält Merkmalsstrukturen und Kategorien eines Wortes. Die Merkmalsstrukturen wiederum enthalten als Wert zum Attribut *sem* einen Verweis auf den zugehörigen Eintrag im semantischen Lexikon. Diese Zweiteilung wurde vorgenommen, da viele syntaktische Einträge eine gemeinsame λ–DRS haben.

Die Kanten der Chart sind nun nicht nur mit den Kategorien und syntaktischen Merkmalsstrukturen markiert, sondern auch mit λ–DRSen. Soll eine aktive Kante ausgedehnt werden, muß auch die Durchführbarkeit der funktionalen Komposition überprüft werden.

[3] Für einen Versuch, LFG und ein λ–DRT ähnliches System in einem integrierten Ansatz zu verarbeiten, siehe [7].

5.6 Behandelte Phänomene

Mit Hilfe der oben beschriebenen λ–DRT und ihrer Anbindung an einen PATR–2 ähnlichen Parser wird nun versucht, verschiedene Phänomene des Deutschen zu analysieren.

Ohne Probleme lassen sich DRSen für einfache Sätze mit Subjekt, Verb, Objekten und Präpositionalphrasen sowie Nominalphrasen mit indefiniten Artikeln und generalisierten Quantoren bilden. Die Konstruktion der λ–DRSen folgt dabei vor allem [9] und [1]. Interessant ist unter anderem die Behandlung von Tempora, Verben als Verbvalenzen, Negation und skopusambigen Sätzen, die im folgenden dargestellt wird.

5.6.1 Tempus

Eine ausführliche Beschreibung zur Behandlung der Tempora in der DRT findet sich in [6]. Grundlage dieser Darstellung ist die Behandlung der Zeiten nach Reichenbach. Hier wird nun versucht, Reichenbachs Ideen in die λ–DRT zu übertragen.

Reichenbachs Darstellung beruht auf drei verschiedenen Punkten auf der Zeitachse: dem Sprechzeitpunkt n, dem Ereigniszeitpunkt e und dem Referenzzeitpunkt t, der angibt, aus welcher Sicht auf ein bestimmtes Ereignis Bezug genommen wird. Diese drei Punkte werden als neue Diskursreferenten in eine DRS eingeführt. Die verschiedenen Tempora werden dann durch Gleichungen über diese Referenten beschrieben.[4]

Das folgende Beispiel zeigt die λ–DRS zu *„Ein Hund wird geschlafen haben.“*:

$$\boxed{\begin{array}{l} x\ e\ t\ n \\ \\ t > n \\ e < t \\ hund(x) \\ \\ e\!:\ \boxed{schlafen(x)} \end{array}}$$

Da die λ–DRSen parallel zur Syntax konstruiert werden, entsteht bei der Übertragung dieser Darstellung folgendes Problem: Die Gleichungen und Diskursreferenten einer bestimmten Zeitform müssen auf die verschiedenen λ–DRSen der Wörter verteilt werden, die zur Konstruktion dieser Form nötig sind.

Einfach ist dies bei Präsens und Präteritum, zu deren Erkennung nur eine Wortform nötig ist. Perfekt, Plusquamperfekt und FuturI bestehen dagegen bereits aus zwei Wörtern. Einheitlich ist für diese Zeiten, daß der Ereigniszeitpunkt mit dem Vollverb eingeführt wird. Der Ereigniszeitpunkt wird zu einem Referenzzeitpunkt, über den dann abstrahiert wird, in Beziehung gesetzt. Der Referenzzeitpunkt wird zusammen mit dem Sprechzeitpunkt und den dazugehörenden Gleichungen durch die λ–DRS des Hilfsverbs eingeführt.

Das folgende Beispiel zeigt die Bildung der Phrase *„hat geschlafen“* aus den zum Hilfsverb *„hat“* und zum PartizipII *„geschlafen“* gehörenden λ–DRSen:

[4] Auf die übliche Unterscheidung zwischen Ereignis– und Zustandsverben wird hier der Einfachheit halber verzichtet.

$$\left\{ \quad \lambda Q \boxed{\begin{array}{l} t\ n \\ n \circ t \end{array}} \; + \; Q(t) \qquad \lambda t \lambda x \boxed{\begin{array}{l} e \\ e < t \\ e{:}\ \boxed{schlafen(x)} \end{array}} \quad \right\} \Rightarrow$$

$$\lambda x \boxed{\begin{array}{l} e\ t\ n \\ n \circ t \\ e < t \\ e{:}\ \boxed{schlafen(x)} \end{array}}$$

Für die Bildung des FuturII sind sogar drei Wörter nötig. Neben dem finiten Hilfsverb besteht der Infinitiv Perfekt aus einem weiteren Hilfsverb und dem eigentlichen Vollverb (*„wird geschlafen haben“*). Sowohl dem finiten Hilfsverb *„wird“* als auch dem PartizipII *„geschlafen“* wurden zur Darstellung des Perfekt bzw. FuturI bereits λ–DRSen zugeordnet. Wendet man auf diese DRSen die funktionale Komposition an, entsteht die gewünschte Darstellung des FuturII. Daraus folgt, daß das Hilfsverb des Infinitiv Perfekt (*„haben“*) keine weitere Bedeutung bei der Bildung der λ–DRS des FuturII hat. Die zugehörige Merkmalsstruktur im syntaktischen Lexikon enthält deshalb kein Merkmal *sem*, um auf einen Eintrag im semantischen Lexikon zu verweisen. Fehlt dieser Verweis auf das semantische Lexikon, wird intern die leere λ–DRS als als Semantik zugeordnet.

5.6.2 Verben als Verbvalenzen

Nicht jede Valenzstelle von Verben wird durch Nominalphrasen gefüllt, auch der Eintrag von weiteren Verben ist möglich. Das folgende Beispiel zeigt die Bildung der Phrase *„glaubte zu schlafen“*:[5]

$$\left\{ \quad \lambda Q \lambda x \boxed{\begin{array}{l} s\ t\ n \\ t \circ n \\ s \circ t \\ s{:}\ \boxed{glauben(x,\ Q(t,x))} \end{array}} \qquad \lambda t \lambda x \boxed{\begin{array}{l} e \\ t \circ e \\ e{:}\ \boxed{schlafen(x)} \end{array}} \quad \right\}$$

$$\Rightarrow \quad \lambda x \boxed{\begin{array}{l} s\ t\ n \\ t \circ n \\ s \circ t \\ s{:}\ \boxed{glauben(x, \boxed{\begin{array}{l} e \\ e \circ t \\ e{:}\ \boxed{schlafen(x)} \end{array}})} \end{array}}$$

Die λ–Liste zu *„schlafen“* enthält zwei Elemente: Gesucht wird ein vom Subjekt eingeführter Diskursreferent und ein Referenzzeitpunkt, zu dem der Ereignisreferent

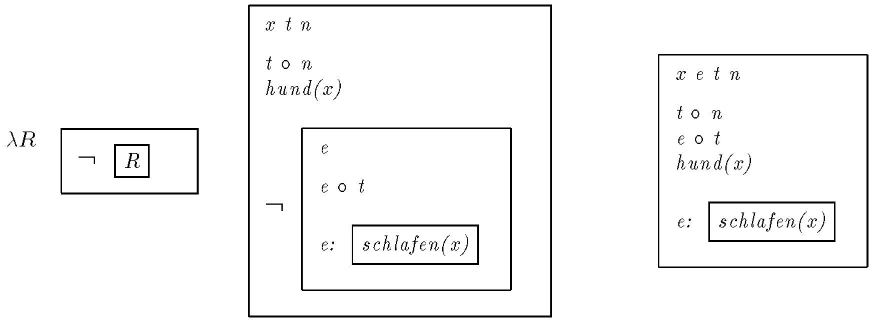
[5] Dem Wort *„zu“* wird dabei keine Semantik zugeordnet.

des Verbs in Beziehung gesetzt werden kann. Diese gesuchten Referenten finden sich beide in der λ–DRS zu *„glauben“*. Für beide Elemente der λ–Liste des Arguments kann also in einem Schritt die λ–Konversion durchgeführt werden.

Dies ist mit der funktionalen Komposition Pinkals nicht möglich, es könnte nur für das erste Element der λ–Liste die Komposition durchgeführt werden. Deshalb wird im weiteren anstelle der Pinkalschen Komposition die allgemeine funktionale Komposition verwendet.

5.6.3 Negation

Das folgende Beispiel zeigt die λ–DRS des Negationspartikels *„nicht“*, sowie die DRS des Satzes *„Ein Hund schläft nicht.“* und des zugehörigen positiven Satzes.



Die Gleichungen und Diskursreferenten zur Darstellung des Präsens müssen alle mit Hilfe der λ–DRS zum finiten Verb *„schläft“* eingeführt werden. Geschieht dies in *einer* λ–DRS, so kann mit Hilfe der funktionalen Komposition zwar der positive, aber nicht mehr der negierte Satz konstruiert werden.

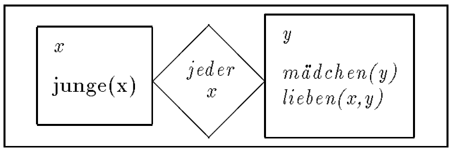
Eine Lösung dieses Problems ist, Verben im Präsens (und auch im Präteritum) nicht nur eine, sondern zwei λ–DRSen zuzuordnen, die zum einen das eigentliche Verb und den Ereignisreferenten einführen und zum anderen die für das Tempus wesentlichen Teile. Damit ähnelt die Darstellung des Präsens der zuvor eingeführten Darstellung des Perfekt, das aus zwei Wörtern zusammengesetzt wurde. Die beiden äquivalenten DRSen für das Präsens werden nun einem Wort zugeordnet.

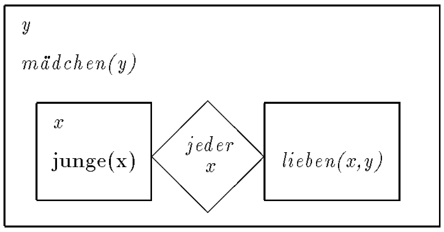
Übertragen auf den zugrundeliegenden Parser heißt das, daß der einzelnen Kante einer Chart nicht nur eine λ–DRS, sondern eine Liste von λ–DRSen zugeordnet wird, die über verschiedene Projektionen angesprochen und verarbeitet werden können. Ein Satz ist erfolgreich geparst, wenn am Ende die satzübergreifende Kante nur eine DRS mit leerer λ–Liste enthält.

5.6.4 Ambige Sätze

Für den Satz *„Jeder Junge liebt ein Mädchen.“* ergeben sich auf Grund der verschiedenen Skopusmöglichkeiten der Artikel zwei verschiedene Lesarten:[6]

[6]Durch die Raute werden generalisierte Quantoren gekennzeichnet.





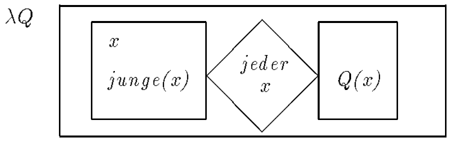
Zum einen können alle Jungen in unterschiedliche Mädchen verliebt sein, es ist aber auch möglich, daß sie nur ein bestimmtes Mädchen verehren. Im ersten Fall ist *„Mädchen“* im Skopus von *„Jungen“* zu finden, im zweiten Fall hat *„Mädchen“* einen weiteren Skopus als *„Jungen“*.

Reyle bildet in [11] diese verschiedenen Lesarten mit Hilfe des *ausgezeichneten Diskursreferentens*, der jeder λ–DRS zugeordnet wird. Mit einer speziellen Operation, der *Argumentidentifikation* werden die Argumente des Prädikats durch die ausgezeichneten Diskursreferenten der λ–DRSen des Subjekts bzw. der Objekte ersetzt. Eine andere Möglichkeit, die verschiedenen Lesarten zu konstruieren ist das *Cooper Storage* (siehe [8]).

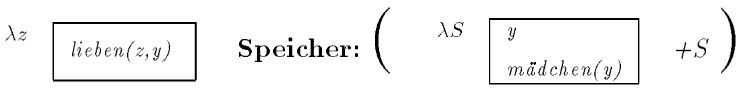
Im vorliegenden Parser wurden zur Bildung ambiger Sätze zwei weitere Operatoren eingeführt. Mit Hilfe des zweistelligen Operators *store* wird eine λ–DRS in einem der DRS zugeordneten Zwischenspeicher abgelegt.

Das folgende Beispiel zeigt die Anwendung des Operators *store* auf die λ–DRSen des Nomens *„Mädchen“* und des Verbs *„lieben“* des obigen Beispielsatzes. Die λ–DRS zum Subjekt ist davon noch nicht betroffen.

NP =



VP =



Um zu gewährleisten, daß der richtige Diskursreferent in die Gleichung der λ–DRS des Verbs eingesetzt wird, wurde die zugehörige Konversion bereits durchgeführt. Die verbleibende λ–DRS des Nomens wird zwischengespeichert. In einem zweiten Schritt kann genauso mit der λ–DRS des Subjekts verfahren werden.

Der Operator *empty-store* wendet dann alle im Speicher enthaltenen λ–DRSen in jeder beliebigen Reihenfolge auf die angegebene λ–DRS an. So entsteht ebenfalls jede mögliche Lesart eine Satzes.

5.7 Schlußbemerkungen und Ausblick

In diesem Beitrag wurde die Verbindung eines PATR–2 ähnlichen Parsers mit einem auf dem λ–Kalkül basierenden Konstruktionsalgorithmus für DRSen beschrieben. Aufbauend auf einer Chart, deren Kanten mit Kategoriesymbolen, Merkmalsstrukturen und einer Liste von λ–DRSen markiert sind, werden die syntaktische und semantische Repräsentation eines Satzes parallel ind inkrementell konstruiert. Die Regeln der zugehörigen Grammatik bestehen aus drei Teilen, wobei im dritten Abschnitt zur Behandlung der λ–DRSen vier Operatoren möglich sind: *id* steht für die Identität, *compose* führt die funktionale Komposition aus und *store* bzw. *empty–store* dienen zur Bildung ambiger Sätze. Zu jeder DRS gehört ein Zwischenspeicher, der vom Operator *store* benötigt wird. Die DRSen werden intern als Merkmalsstrukturen dargestellt, wobei zur Durchführung der funktionalen Komposition die Unifikation benutzt wird. Mit Hilfe dieses Systems lassen sich verschiedene Konstruktionen des Deutschen wie gezeigt behandeln. Die folgenden Punkte sind noch nicht bearbeitet worden:

1. Die freie Wortstellung im Deutschen wurde nicht implementiert. Mit dem bisherigen Verfahren zur λ–Konversion, bei dem die λ–Liste sequentiell abgearbeitet wird, ist eine feste Wortstellung vorgegeben, die sich aus der Reihenfolge der λ–Liste ergibt. Eine Möglichkeit, diese Reihenfolge zu varieren ist in [1] gezeigt.

2. Wesentlicher Bestandteil der DRT ist die Behandlung von Pronomina. Im vorhandenen System ist allerdings die Auflösung von Anaphern und Kataphern noch nicht möglich. Die Auflösung sollte parallel zur Konstruktion der Merkmalsstrukturen und DRSen erfolgen.

Literaturverzeichnis

[1] Asher, Nicolas: *Reference to Abstract Objects in Discourse*, Kluwer Academic Press, 1993

[2] Bröker, Norbert: *Erweiterung eines Systems zur kontextfreien Strukturanalyse natürlicher Sprachen um einen Unifikationsformalismus*, Studienarbeit, Universität Erlangen–Nürnberg, 1990

[3] Fischer, Ingrid: *Die kompositionelle Bildung von Diskursrepräsentationsstrukturen über einer Chart*, Diplomarbeit, Universität Erlangen–Nürnberg, 1993

[4] Görz, Günther: *Ein Verarbeitungsmodell zum maschinellen Verstehen gesprochener und geschriebener Sprache*, Addison–Wesley, 1988

[5] Johnson, M.; Klein, E.:*Discourse, Anaphora and Parsing*, Proceedings of the 11th Conference on Computational Linguistics, 1986

[6] Kamp, H.; Reyle, U.:*From Discourse to Logic*, Kluwer Academic Press, 1993

[7] Kasper, Walter: *The Construction of Semantic Representation from F-Structures* in Bes, Grabriel: *The Construction of a Natural Language and Graphics Interface. Results and Perspectives from the ACORD Project*, Springer Verlag, Heidelberg, 1992

[8] Keller, William R.: *Nested Cooper Storage: The Proper Treatment of Quantification on Ordinary Noun Phrases*, in Reyle, U. and Rohrer, C. *Natural Language Parsing and Linguistic Theory*, D. Reidel Publishing Company, 1988

[9] Millies, S., Pinkal, M.: *An Autonomous Syntax–Sensitive Module for Semantic Interpretation*, unveröffentlichtes Manuskript, Universität Saarbrücken, 1993

[10] Millies, S., Pinkal, M.: *Linking One Semantic System to different Syntactic Formalisms*, unveröffentlichtes Manuskript, Universität Saarbrücken, 1993

[11] Reyle, Uwe: *Zeit und Aspekt bei der Verarbeitung natürlicher Sprache*, Dissertation, Institut für Linguistik der Universität Stuttgart, 1986

[12] Stürmer, Thomas: *Modellhafte Implementation von Analysealgorithmen für natürliche Sprache*, Studienarbeit, Universität Erlangen–Nürnberg, 1989

[13] Zeevat, Henk: *A Compositional Approach to Discourse Representation Theory*, Linguistics and Philosophy 12, 1989

Kapitel 6

Ein dreischichtiger Ansatz für die Dialogverarbeitung in Verbmobil

Norbert Reithinger, Elisabeth Maier, Jan Alexandersson
DFKI GmbH, Stuhlsatzenhausweg 3
D-66123 Saarbrücken
{reithinger,maier,alexandersson}@dfki.uni-sb.de

Zusammenfassung

In diesem Beitrag wird die Dialogkomponente des VERBMOBIL Systems für die Übersetzung spontansprachlicher Dialoge vorgestellt. Im Gegensatz zu herkömmlichen Dialogsystemen vermittelt dieses System zwischen den Dialogteilnehmern. Es verarbeitet dabei maximal 50% eines Dialogs vollständig. Die anderen Dialogbeiträge werden nur oberflächlich anhand von Schlüsselwörtern verfolgt. Anforderungen wie Robustheit und Verarbeitungseffizienz führten zur Entwicklung einer dreischichtigen hybriden Architektur für die Dialogkomponente, die sich aus den Moduln Statistik, Automat und Planer zusammemsetzt. Während der Verarbeitung wird außerdem ein Dialoggedächtnis aufgebaut.

6.1 Einleitung

Das System VERBMOBIL, das gesprochene Sprache übersetzt, stellt besondere Anforderungen an die Dialogkomponente. Dialogverarbeitung in VERBMOBIL unterscheidet sich von interaktiven Systemen wie z.B. SUNDIAL [Bilange, 1991, Mast, 1993] insofern, als das System lediglich zwischen zwei Diskursteilnehmern vermittelt und keine aktive Rolle übernimmt. Eine Ausnahme bilden hier lediglich Klärungsdialoge, die aktiviert werden, wenn Verarbeitungsschwierigkeiten auftreten (z.B. ein Dialogteilnehmer spricht zu laut). Eine weitere Besonderheit bei VERBMOBIL ist, daß beide Dialogpartner passive Kenntnisse des Englischen besitzen, welches als Zwischensprache verwendet wird. Übersetzungen werden “on demand” erstellt mit der Konsequenz, daß lediglich Teile eines Dialogs verarbeitet werden. Solange keine Übersetzungsanforderung vorliegt verfolgt ein Schlüsselworterkenner – ein sogenannter *Key Word Spotter* – den Dialog, indem anhand von Signalwörtern bestimmte Dialogzustände erkannt werden.

In diesem Artikel wird die Dialogkomponente mit ihren einzelen Submoduln vorgestellt. Im Anschluß daran werden zukünftige Erweiterungen diskutiert.

6.2 Aufgaben der Dialogkomponente

Die Dialogkomponente von VERBMOBIL hat vier Hauptaufgaben, die das Design der Dialogkomponente wesentlich beeinflußten:

- Sie unterstützt die Analysekomponenten mit *Top-down Erwartungen*, um Suchräume einzuschränken und damit die Analyse schneller und robuster ablaufen zu lassen [Young et al., 1989, Andry, 1992, Niedermair, 1992]. Vorhersagen von Folge-Sprechhandlungen können beispielsweise eingesetzt werden um die Menge der mögliche Worte einzuschränken, die im unmittelbar nächsten Dialogschritt auftreten können. Dieses Verfahren wird bereits im Rahmen der Spracherkennung ausgenutzt, wo adaptive Sprachmodelle (*Adaptive Language Models*) verwendet werden [Niedermair, 1992]. Top-down Erwartungen können außerdem benutzt werden, um abhängig vom Dialogzustand die Menge der anwendbaren Grammatikregeln auf eine Subgrammatik zu reduzieren. Top-down Erwartungen sind in VERBMOBIL von besonderer Bedeutung, da das System unter Realzeitbedingungen arbeiten muß.

- Sie versorgt die Übersetzungskomponente mit Informationen über den Kontext; ein häufig zitiertes Beispiel ist die Äußerung *"Geht es bei Ihnen?"*, was mit *"Does it suit you?"* übersetzt wird, wenn der Kontext ergibt, daß über Termine gesprochen wird, oder mit *"How about your place?"*, wenn es um Orte geht. Es wird ein Dialoggedächtnis aufgebaut, das von allen VERBMOBIL-Komponenten verwendet werden kann, wenn Kontextinformation benötigt wird.

- Sie folgt dem Dialog, wenn beide Partner Englisch reden und eine tiefe Analyse nicht durchgeführt wird. In diesem Fall steuert die Dialogkomponente einen Schlüsselworterkenner an, der abhängig vom nächsten erwarteten Dialogschritt eine vorher bestimmte geringe Menge von Schlüsselwörtern erkennt. Die Dialogkomponente bestimmt die wahrscheinlichste Folgesprechhandlung, für welche dann die typischen Schlüsselworte bestimmt werden.

- Sie führt Klärungsdialoge mit dem Benutzer, falls die Verarbeitung einer Äußerung nicht durchgeführt werden kann.

6.3 Die Architektur

Die im vorherigen Kapitel angeführten Anforderungen können nicht mithilfe einer einzigen Verarbeitungsmethodik geleistet werden: einerseits lassen sich aus strukturorientierten Wissensquellen wie z.B. Plänen oder Dialoggrammatiken keine Top-down Erwartungen ableiten; aus einer Wissenquelle, die lediglich über statistisch unbewertete Informationen verfügt, lassen sich üblicherweise nur Mengen von "gleichwertigen" Folgesprechhandlungen ableiten [Nagata und Morimoto, 1993]. Andererseits ist auch ein rein plan-basierter Ansatz ungünstig, insbesondere dann wenn ein Dialog nur teilweise verarbeitet wird. Aus diesen Gründen wurde eine hybride Architektur entwickelt, die aus drei geschichteten Verarbeitungseinheiten besteht (siehe Abbildung 6.1). Die

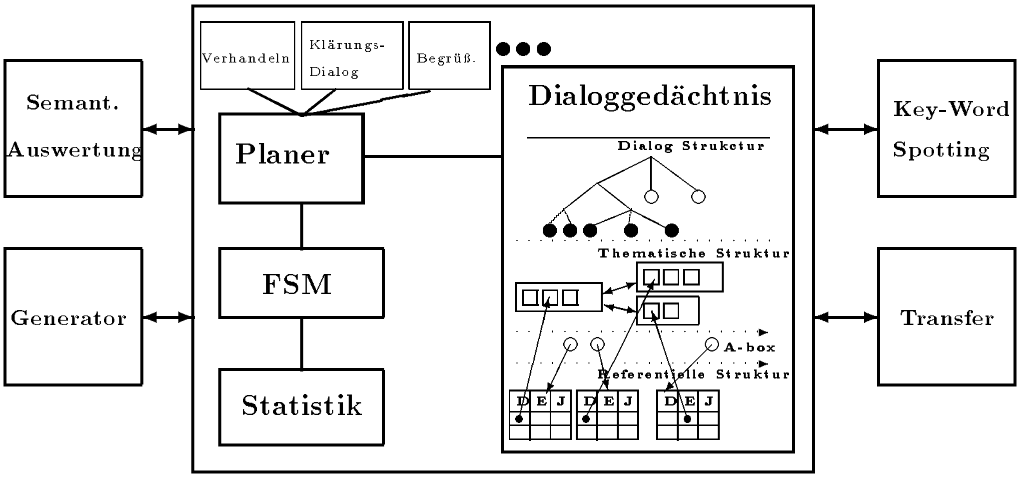


Abbildung 6.1: Architektur der Dialogkomponente

drei Einheiten unterscheiden sich jeweils im Bezug auf Verarbeitungsgeschwindigkeit und linguistische Kompetenz:

Statistikmodul Auf der Basis von Daten, die in einer Trainingsphase gewonnen wurden, sagt dieser Modul die jeweils nächste Sprechhandlung voraus.

Endlicher Automat (Finite State Machine (FSM)) Ausgehend von einer Beschreibung von möglichen Sprechhandlungssequenzen testet der endliche Automat, ob eine Eingabeäußerung im Bezug auf ein zugrundeliegendes Dialogmodell (siehe Abbildung 6.2) konsistent ist.

Planer Der hierarchische Planer als wissensintensivster Modul innerhalb der Dialogkomponente erzeugt eine hierarchische Beschreibung der Dialogstruktur und baut das Dialoggedächtnis auf. Dazu wird kontextuelles und pragmatisches Wissen ausgewertet.

Das Dialoggedächtnis gliedert sich in drei Teile: (a) die *Dialogstruktur*, die die intentionale Struktur des Dialogs wiedergibt, (b) die *Thematische Struktur* für den attentionalen Zustand und (c) die *Referentielle Struktur*, die konzeptuelle und sprachliche Daten für Äußerungen speichert.
Als Ein- und Ausgabeformat zwischen semantischer Konstruktion/Auswertung und der Dialogkomponente bzw. zwischen der Dialogkomponente und dem Transfermodul werden DRS-artige Merkmalsstrukturen verwendet [Bos et al., 1994]. Der Schlüsselworterkenner kommuniziert mit dem Dialgmodul einerseits über ein Wörtergitter, das die erkannten Schlüsselworte enthält, und andererseits über eine Liste, die angibt, welche Wörter innerhalb der nächsten Äußerung erwartet werden können. Schließlich liefert die Generierungskomponente noch Angaben über die übersetzte Äußerung an die Dialogkomponente.
Die Ausgabe der Dialogkomponente wird an die Komponenten weitergegeben, die Informationen über die vorangehenden Dialogschritte benötigen, also zum Beispiel an das Transfermodul und an die Komponente für die Semantische Auswertung. Außerdem wird der Schlüsselworterkenner mit den Signalwörtern versorgt, die in der nächsten Äußerung erwartet werden können.

6.4 Das Dialogmodell

Wie bereits andere Ansätze zur Modellierung von Interaktion in Sprachverarbeitenden Systemen (z.B. [Bilange, 1991], [Mast, 1993]) basiert auch unser Ansatz auf der Annahme, daß sich ein Dialog mithilfe von Sprechakten, oder *Sprechhandlungen*, beschreiben läßt. Als Ausgangspunkt dienten zunächst *Sprechakte* [Austin, 1962, Searle, 1969] und *Illokutionäre Akte* [Sitter und Stein, 1992]. Dann wurde anhand zufällig ausgewählter Dialoge unseres VERBMOBIL Korpus untersucht ob diese Sprechhandlungen auftauchen und ob zusätzlich neue Sprechhandlungen eingeführt werden müssen.
Das Dialogmodell (vgl. Abbildung 6.2) beschreibt das Potential der möglichen Sequenzen von Sprechhandlungen bei Terminverhandlungen. Die Kanten des in der Abbildung gezeigten Modells entsprechen dabei den oben erwähnten Sprechhandlungen. Zur Zeit enthält unser Dialogmodell 17 Sprechhandlungen [Maier, 1994]. Zu den domänenunabhängigen Sprechhandlungen gehören Sprechakte wie z.B. AKZEPTANZ und ABLEHNUNG. Außerdem wurden einige domänenabhängige Sprechhandlungen speziell für die Modellierung von Terminverhandlungsdialogen eingeführt, wie z.B. INIT_TERMINABSPRACHE und BESTAETIGUNG. Während INIT_TERMINABSPRACHE verwendet wird um den Gegenstand der Verhandlung einzuführen, also in unserem Fall zu vereinbarende Termine, beschreibt die Sprechhandlung BESTAETIGUNG Äußerungen, die ein gegenseitiges Einverständnis zum Gegenstand haben.
Ein Terminverhandlungsdialog besteht aus drei Phasen: die erste Phase besteht aus einer Einleitung, in welcher sich die Dialogteilnehmer gegenseitig begrüßen, sich vorstellen und Angaben über ihre berufliche Position machen. Danach wird der Verhandlungsgegenstand genannt und zur eigentlichen Verhandlungsphase übergegangen. Dabei machen die Dialogpartner wiederholt Vorschläge und Gegenvorschläge, die dann angenommen oder zurückgewiesen werden oder sie fordern weitere Vorschläge und Stellungnahmen des jeweils anderen Dialogteilnehmers an. Sobald ein Termin angenommen wurde und gegenseitiges Einverständnis besteht, wird in die Schlußphase übergegangen und der Dialog mit Dankes- und Verabschiedungsfloskeln beendet.
Ein Dialogmodell, das auf Sprechhandlungen aufbaut, ist besonders geeignet im Hinblick auf die Maschinelle Übersetzung und den Transfer: während in monolingualen Diskursen (d.h. Texten) üblicherweise ganze Sätze als elementare Verarbeitungseinheiten verwendet werden, kann diese Annahme nicht für spontansprachliche Diskurse übernommen werden, in welchen oft unvollständige oder grammatikalisch unkorrekte Äußerungen auftreten. Aus diesem Grund wurden für VERBMOBIL Sprechhandlungen als atomare Einheiten bei der Dialogverarbeitung gewählt.
Die Sprechhandlungen, die bei unserem Ansatz in ein sequentielles Modell zur Beschreibung von Interaktionen eingebettet sind, können zusätzlich anhand der in [Bunt, 1989] beschriebenen Taxonomie von Dialogkontrollfunktionen klassifiziert werden. Sprechhandlungen wie BEGRUESSUNG und VERABSCHIEDUNG können beispielsweise als Dialogkontrollfunktionen betrachtet werden, die das Interaktionsmanagement zwischen den Dialogpartnern betreffen. Feinere taxonomische Unterscheidungen wie in [Bunt, 1994] vorgeschlagen, z.B. CONFIRM und CONFIRM/WEAK, werden in VERBMOBIL durch *pragmatische Features* abgedeckt, wie z.B. *suitability* und *possibility*. Diese pragmatischen Features werden in der DRS-Beschreibung einer Äußerung spezifiziert, die dann als Eingabe für die Dialogkomponente dient.

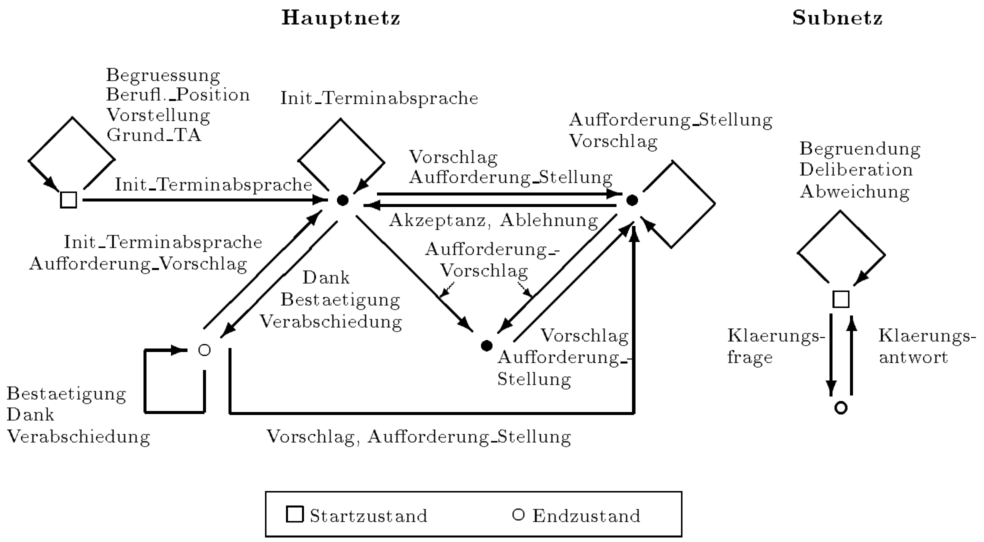


Abbildung 6.2: Das Dialogmodell zur Beschreibung von Terminverhandlungsdialogen

6.5 Kurzbeschreibung der Dialogmodule

Im Folgenden werden die einzelnen Subkomponenten beschrieben, aus welchen sich die Dialogkomponente zusammensetzt. Alle drei Teilkomponenten basieren auf den Sprechhandlungen wie sie in Abschnitt 6.4 eingeführt wurden.

6.5.1 Der endliche Automat

Der endliche Automat stellt eine (rechenzeit-) effiziente Umsetzung des Dialogmodells dar. Der Automat verarbeitet die Sprechhandlungen der Eingabe und überprüft ihre Kompatibilität mit dem Dialogmodell. Außerdem überführt der Automat den Dialog in einen neuen definierten Zustand. Ist eine Eingabe nicht verträglich mit dem Dialogmodell, dann übergibt der Automat die Kontrolle an den Dialogplaner, der Methoden zur "Reparatur" des Dialogs anwendet. Der Automat verarbeitet auch Phänomene, die an jeder Stelle im Dialog auftreten können, wie z.B. Begründungen oder Deliberationen. Klärungsdialoge können dabei beliebig tief verschachtelt sein.

6.5.2 Planbasierte Verarbeitung

Um Constraints in die Dialogverarbeitung miteinbeziehen zu können und um es zu ermöglichen, daß bestimmte Entscheidungen in der Dialogverarbeitung Folgeaktionen nach sich ziehen, wurde unter anderem auch ein planbasierter Ansatz realisiert. Dieser Ansatz wurde an Verfahren aus der Textgenerierung angelehnt, wo Planoperatoren für die Erzeugung kohärenter Textsegmente eingesetzt werden [Hovy, 1988, Moore und Paris, 1989]. Die Anwendung eines Planoperators hängt davon ab, ob im aktuellen Zustand vorspezifizierte Constraints erfüllt werden können. Diese Constraints beziehen sich im Wesentlichen auf kontextuelle und pragmatische Eigenschaften des Diskurses, z.B. auf den Bekanntheitsgrad der Diskursteilnehmer, auf deren Vorwissen, oder auf das Auftreten bestimmter Sprechhandlungen im unmittelbaren Kontext.

```
begin-plan-operator GENERIC-OPERATOR
   goal [SCHEDULE-MEETING]
   constraints nil
   actions nil
   subgoals (sequence [GREET-INTRODUCE-TOPIC]
                      iterate [NEGOTIATE]
                      [FINISH])
end-plan-operator

begin-plan-operator OFFER-OPERATOR
   goal [OFFER]
   constraints nil
   actions (retrieve-theme)
   subgoals primitive
end-plan-operator
```

Abbildung 6.3: Beispiele von Planoperatoren für die Dialogverarbeitung

Unsere Planoperatoren enthalten zusätzlich Angaben zu auszuführenden Folgeaktion im Fall, daß der Planoperator zur Anwendung kommen sollte. Actions werden für die Interaktion mit anderen Dialogprozessen eingesetzt. Im Sub-Plan `OFFER-OPERATOR` (siehe Abbildung 6.3), der für die Planung einer Sprechhandlung vom Typ VORSCHLAG zuständig ist, veranlaßt die Aktion `(retrieve-theme)` und damit die Extraktion der temporalen Information (Angaben zu Monat, Woche, Tag) aus der aktuellen Äußerung. Mit dieser Information wird die thematische Struktur des Dialoggedächtnisses (siehe Abschnitt 6.6) aktualisiert.
Während der Planung werden automatisch baumartige Strukturen aufgebaut, die die Struktur des Dialogs wiedergeben. Das Dialoggedächtnis besteht aus drei Ebenen: (1) einer *intentionalen Struktur*, die Dialogphasen und Sprechhandlungen repräsentiert, (2) eine *thematische Struktur*, die Zeitangaben speichert und Informationen darüber besitzt, wie diese Angaben von den Dialogpartnern bewertet wurden, (3) eine *referentielle Struktur*, die eine Beziehung zwischen erwähnten Konzepten und ihrer Oberflächenrealisierung herstellt.
Die Planung des Dialogs erfolgt Top-down, d.h. abstrakte Ziele werden in Subziele zerlegt, die in einer bestimmten Reihenfolge erfüllt werden müssen. In unserem Fall ist das abstrakteste Ziel, das mit der Durchführung eines Dialogs erfüllt wird `SCHEDULE-MEETING` (siehe Abbildung 6.3). Es besteht aus drei Unterzielen, von welchen jedes eine Dialogphase beschreibt: die Einleitungsphase (`GREET-INTRODUCE-TOPIC`), die Verhandlungsphase (`NEGOTIATE`) und die Schlußphase (`FINISH`). Das Schlüsselwort `iterate` zeigt an, daß Verhandlungsphasen wiederholt auftreten können.
In unserer Hierarchie von Planoperatoren entsprechen die Blätter, d.h. die spezifischsten Planoperatoren, den einzelnen Sprechhandlungen wie in Abbildung 6.2 definiert.
Der endliche Automat, die plan-basierte und die statistische Teilkomponente interagieren auf verschiedene Art und Weise: in Fällen, in denen Lücken im Dialog auftreten – zum Beispiel wenn VERBMOBIL für eine gewisse Zeit nicht aktiviert wurde oder wenn die Spracherkennung kein Ergebnis liefern kann – kann die Statistik-Komponente dazu beitragen, die fehlenden Sprechhandlungen zu bestimmen. Für den Fall, daß der

endliche Automat eine fehlerhafte Eingabe entdeckt, kann der Planer sogenannte Reparaturoperatoren aktivieren.

6.5.3 Die statistische Ebene

Auf der statistischen Verarbeitungsebene werden Ansätze aus der Language Modellierung auf die Ebene der Dialogverarbeitung übertragen [Jellinek, 1990, Nagata und Morimoto, 1993], wobei die Einheiten, die verarbeitet werden, nicht Wörter sondern Sprechhandlungen sind. Das Verarbeitungsziel ist dabei die Vorhersage einer Sprechhandlung anhand der Sprechhandlungen der vorangegangenen Äußerungen. Diese Daten können z.B. benutzt werden, um den Schlüsselworterkenner anzusteuern oder die Sprechhandlung des folgenden Satzes zu desambiguieren. Die Basis des Verfahrens sind Unigramm-, Bigramm- und Trigrammhäufigkeiten, die aus einem Trainingskopus berechnet werden. Um die s_n-te Sprechhandlung vorherzusagen muß das Maximum der bedingten Wahrscheinlichkeiten

$$s_n = \max_s P(s|s_{n-1}, s_{n-2}, s_{n-3}, \ldots)$$

berechnet werden. Die Wahrscheinlichkeiten können mit der Interpolationsformel

$$P(s_n|s_{n-1}, s_{n-2}) = q_1 f(s_n) + q_2 f(s_n|s_{n-1}) + q_3 f(s_n|s_{n-1}, s_{n-2}) \qquad (\sum \mathrm{s_i} = 1)$$

berechnet werden [Jellinek, 1990], wobei $f(s_n|S_{n-1}, \ldots)$ jeweils Unigramm-, Bigramm- und Trigramm-Häufigkeiten sind.
Die folgende Tabelle zeigt die Performanz des Ansatzes für drei Beispielexperimente:

Pred.	*TS1*	*TS2*	*TS3*
1	44,24 %	37.47 %	40.28 %
2	66,47 %	56.50 %	59.62 %
3	81,46 %	69.52 %	71.93 %

Für alle Experimente wurden 41 deutsche Dialoge (mit 2472 Sprechhandlungen) aus dem VERBMOBIL Korpus als Trainingsdaten herangezogen. TS1 and TS2 verwenden die selben 81 Dialoge als Testdaten. Der Unterschied zwischen beiden Experimenten ist, daß in TS1 nur Sprechhandlungen aus dem Hauptnetzwerk des Dialogmodells herangezogen wurden, während TS2 auch die Abweichungen berücksichtigt. Aus den Daten ergibt sich, daß – wie zu erwarten – die Trefferquote sinkt, wenn nicht vorhersehbare Abweichungen mit berücksichtigt werden. TS3 zeigt die Trefferquoten für alle derzeit mit Sprechhandlungen annotierten Dialoge (über 200 mit 7197 Sprechhandlungen).

6.6 Ein annotiertes Beispiel

Um einen Einblick in die Verarbeitung eines Dialogs mithilfe der oben beschriebenen Dialogkomponente zu geben zeigen wir hier die Verarbeitung von drei Äußerungen, die einen Ausschnitt aus einem Dialog darstellen (siehe Abbildung 6.4), der aus insgesamt 25 einzelnen Dialogbeiträgen besteht. Der Dialog wurde aus unserem Korpus von annotierten Terminverhandlungsdialogen entnommen, der momentan 200 Dialoge umfaßt.
Vor der Äußerung `DE004` initialisierte der Dialogpartner `EL` den Dialog indem er anzeigte, einen Termin für eine Dienstreise ausmachen zu wollen[1].

```
DE004:   #oh ja, gut, nach meinem Terminkalender <Pause>, wie w"ars im
         Oktober?# (VORSCHLAG)
VM005:   just lookin at my diary,  I would suggest October. (VORSCHLAG)
DE006/1: <Pause> I propose from  Tuesday the fifth/-
DE006/2: <Pause> no, Tuesday the fourth to Saturday the eighth <Pause>,
         those five days? (VORSCHLAG)
EL007:   oh, that's too bad, I'm not free right then. (ABLEHNUNG) <Pause>
         I could fit it into my schedule <Smack> the week after, from
         Saturday to Thursday, the thirteenth. (VORSCHLAG)
```

Abbildung 6.4: Mit Sprechhandlungen annotierter Beispieldialog aus dem VERBMOBIL-Korpus.

Bei der Verarbeitung des Dialogfragments mittels Automat und Statistikkomponente erhält man bei zwei Vorhersagen pro Dialogschritt die folgenden Daten:

```
EL003:     INIT_TERMINABSPRACHE
           Prediction: (VORSCHLAG AUFFORDERUNG_VORSCHLAG)
DE004:     VORSCHLAG
           Prediction: (AKZEPTANZ VORSCHLAG)
DE006/1:   VORSCHLAG
           Prediction: (AKZEPTANZ VORSCHLAG)
DE006/2:   VORSCHLAG
           Prediction: (AKZEPTANZ VORSCHLAG)      ***Failed***
EL007/1:   ABLEHNUNG
           Prediction: (VORSCHLAG AUFFORDERUNG_STELLUNG)
EL007/2:   VORSCHLAG
           Prediction: (AKZEPTANZ ABLEHNUNG)
```

Während der Automat die Sprechhandlungssequenzen ohne Fehler verarbeitet, sind die Vorhersagen, die von der Statistik erstellt werden, für die Äußerung `DE006/2` nicht korrekt. Die vier besten Vorhersagen zusammen mit den jeweiligen Auftretenswahrscheinlichkeiten sind `AKZEPTANZ (28.09%)`, `VORSCHLAG (26.93%)`, `ABLEHNUNG (21.67%)` und `AUFFORDERUNG_STELLUNG (9.7%)`. Im Vergleich zur vierten Vorhersage weisen die ersten drei Vorhersagen ähnliche Werte auf. Aus diesem Grund kann das Fehlschlagen der Statistikkomponente an dieser Stelle im Dialog als ein "near miss" betrachtet werden. Die Gesamtvorhersagewerte für den ganzen Dialog sind 56.52 %, 82,60%, und 95.65% bei jeweils einer, zwei oder drei Vorhersagen.

Da der Dialog vom Automaten fehlerfrei verarbeitet werden kann sind keine Reparaturoperationen notwendig. Die einzige Aufgabe des Planers ist aus diesem Grund der Aufbau des Dialoggedächtnisses (siehe Abbildung 6.5).

Der Planer fügt die aktuelle Sprechhandlung in die intentionale Struktur des Dialoggedächtnisses ein, speichert die Zeitangaben, die im Laufe der Verhandlung zur Sprache kommen, und verwaltet die verschiedenen Oberflächenrealisierungen der im Dialog erwähnten Referenten. Auf diese Art und Weise werden Phänomene wie lexikalische Variationen und referentielle Ausdrücke im Dialoggedächtnis repräsentiert. Außerdem beinhaltet das Dialoggedächntnis auch Verweise zwischen den im Text erwähnten Objekten und ihrer Repräsentation in BACK [Hoppe et al., 1993]. In Abbildung 6.5 geben

[1] `DE` bezeichnet den deutschsprachigen Dialogteilnehmer, `EL` den englischsprechenden. `VM` steht für die Äußerungen, die von VERBMOBIL übersetzt wurden. `#` zeigt an, wann der Knopf gedrückt wurde, mit dem eine Übersetzung angefordert und VERBMOBIL aktiviert wird.

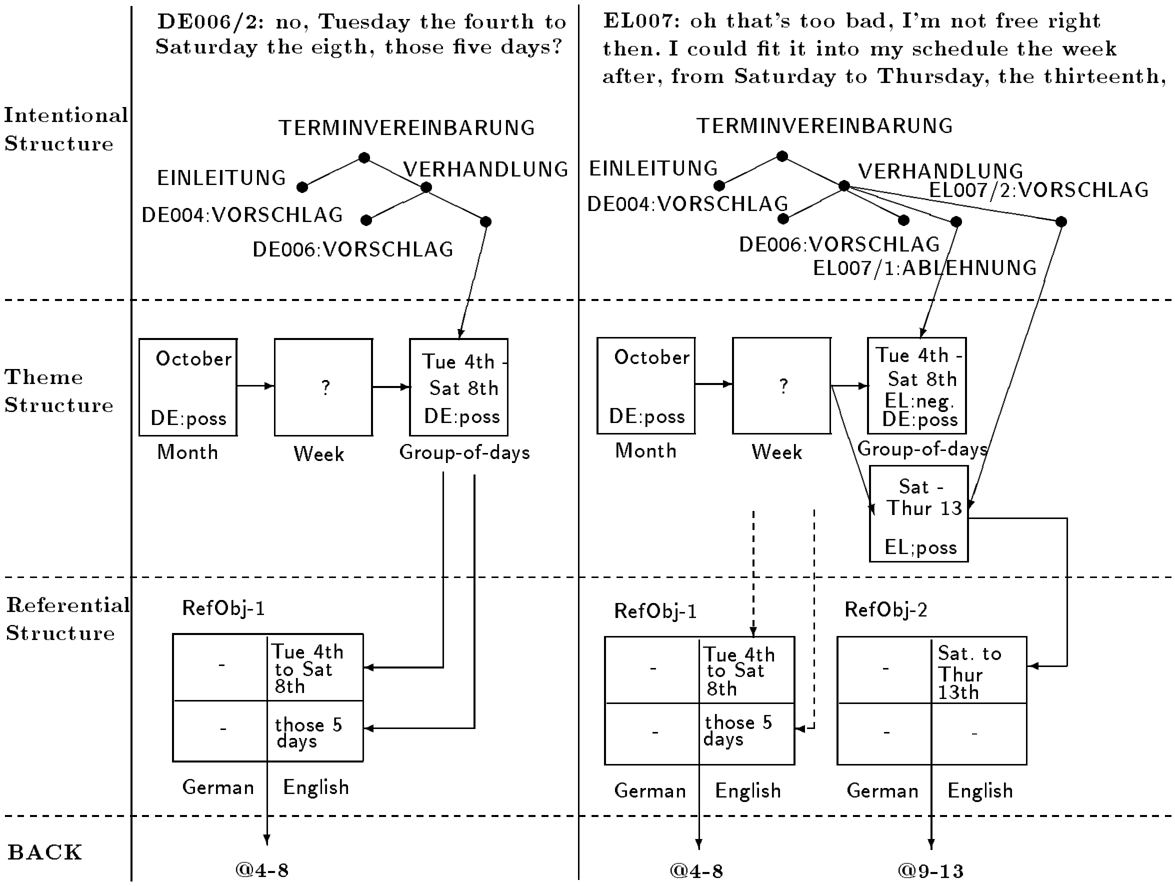


Abbildung 6.5: Der Zustand des Dialoggedächtnisses nach der Verarbeitung der Äusßerungen DE006/2 und EL007

wir ein Beispiel für den inkrementellen Aufbau des Dialoggedächntnisses bei der Verarbeitung der Äußerungen DE006/2 und EL007.

6.7 Zusammenfassung und Ausblick

Unser Ansatz unterscheidet sich von anderen Ansätzen zur Dialogverarbeitung in sprachverstehenden Systemen wie z.B. [Mast, 1993, Bilange, 1991] insofern als diese für Mensch-Maschine Dialoge entwickelt wurden während in VERBMOBIL sowohl Mensch-Mensch Dialoge mit der Maschine (Übersetzung) als auch Mensch-Maschine Dialoge (Klärungsdialoge) verarbeitet werden müssen. Das System nimmt dabei eine vermittelnde Stellung ein. Zudem muß das System in Phasen, in denen keine Übersetzungsleistung benötigt wird, die Interaktion oberflächlich verfolgen können. Die Verarbeitung mit drei unterschiedlichen Prozessen – wobei wenn möglich effiziente Methoden (Statistik, endlicher Automat) aufwendigen aber andererseits verlässlichen Methoden (Planung) vorgezogen werden – garantiert eine robuste und den Anforderungen angemessene Leistung der Dialogkomponente. Eine wichtige Aufgabe der Dialogkomponente ist dabei die Bestimmung von Top-Down-Erwartungen, die anderen VERBMOBIL-Moduln zur Verfügung gestellt werden. Diese Erwartungen werden verwendet um den Suchraum der Spracherkennungskomponenten einzuschränken. Eine weitere Aufgabe der Dialogkomponente ist der Aufbau eines Dialoggedächntnisses, das kontextuelle Informationen enthält, die im Wesentlichen von der Transfer- und der Generierungskomponente benötigt werden. Die inkrementelle Konstruktion des Dialoggedächntnisses leistet dabei die Planungskomponente. Die mehrschichtige Verarbeitung von Dialogen durch die VERBMOBIL-Dialogkomponente zeichnet sich durch Effizienz und Robustheit aus. Momentan werden alle 200 Dialoge unseres Korpus von annotierten Terminverhandlungsdialogen verarbeitet.
Die Dialogkomponente wurde in CommonLISP implementiert und umfasst ca. 450 KB Programmcode. Für die Kommunikation mit anderen VERBMOBIL-Moduln wird das PVM/ICE-System verwendet [Auerswald und Amtrup, 1994].
Zukünftige Erweiterungen des Systems betreffen die folgenden Punkte:

- die Behandlung sogenannter *multipler* Sprechhandlungen:
  In der aktuellen Implementation wird pro Äußerung lediglich eine Sprechhandlung verarbeitet. Da jedoch angenommen werden muß, daß mehr als eine Sprechhandlung pro Äußerung auftreten kann (eine Sprechhandlung vom Typ INIT_TERMINABSPRACHE kann beispielsweise gleichzeitig als VORSCHLAG fungieren) muß dies in unserem Dialogmodell berücksichtigt werden. Die verschiedenen Sprechhandlungen sollen außerdem mit Wahrscheinlichkeiten annotiert sein.

- die Verwendung von Wahrscheinlichkeiten in Planoperatoren:
  die Planoperatoren werden mit statistischen Angaben versehen, so daß der im gegebenen Kontext wahrscheinlichste Planoperator zuerst getestet werden kann. Eine solche Vorgehensweise kann die Effizienz des Planungsprozesses weiter steigern.

Literaturverzeichnis

[Andry, 1992] Francois Andry. 1992. Static and dynamic predictions: a method to improve speech understanding in cooperative dialogues. In *Proceedings of the International Conference on Spoken Language Processing*, Seiten 639–642, Banff.

[Auerswald und Amtrup, 1994] Marko Auerswald and Jan Amtrup. Kommunikationsarchitektur für den Demonstrator. Verbmobil Technisches Dokument 15, DFKI Kaiserslautern, Universität Hamburg, Dezember 1994.

[Austin, 1962] John Austin. 1962. *How to do things with words.* Oxford: Clarendon Press.

[Bilange, 1991] Eric Bilange. 1991. A task independent oral dialogue model. In *Proceedings of the Fifth Conference of the European Chapter of the Association for Computational Linguistics (EACL-91)*, Seiten 83–88, Berlin.

[Bos et al., 1994] Johan Bos, Elsbeth Mastenbroek, Scott McGlashan, Sebastian Milles, und Manfred Pinkal. 1994. The VERBMOBIL semantic formalism. Arbeitsbereich Computerlinguistik der Universität des Saarlandes, Saarbrücken.

[Bunt, 1989] Harry C. Bunt. 1989. Information Dialogues as Communicative Action in Relation to Partner Modelling and Information Processing. In M. M. Taylor, F. Néel, and D. B. Bouwhuis, editors, *The Structure of Multimodal Dialogue*, pages 47–73. Elsevier Sience Publishers, North-Holland.

[Bunt, 1994] Harry C. Bunt. 1994. Context and Dialogue Control. *Think*, 3:19–31, May.

[Hoppe et al., 1993] Thomas Hoppe, Carsten Kindermann, J. Joachim Quantz, Albrecht Schmiedel, and Martin Fischer. 1993. BACK V5 Tutorial & Manual. KIT - REPORT 100 ISSN 0931-0436, TU Berlin, März.

[Hovy, 1988] Eduard H. Hovy. 1988. Planning coherent multisentential text. In *Proceedings of the 26th ACL Conference*, pages 179–186, Buffalo.

[Jellinek, 1990] Fred Jellinek. 1990. Self-organized language modeling for speech recognition. In A. Waibel und K.-F. Lee, editors, *Readings in Speech Recognition*, Seiten 450–506. Morgan Kaufmann.

[Maier, 1994] Elisabeth Maier. 1994. Dialogmodellierung in VERBMOBIL – Festlegung der Sprechhandlungen für den Demonstrator. Technical Report Verbmobil–Memo 31, DFKI Saarbrücken, Juli. Mit Beiträgen von Folker Caroli, Susanne Jekat-Rommel, Walter Kasper, Elisabeth Maier, Ilona Maleck, Marion Mast, Bärbel Ripplinger, Nina Ruge, Birte Schmitz.

[Mast, 1993] Marion Mast. 1993. *Ein Dialogmodul für ein Spracherkennungs- und Dialogsystem.* Dissertation, Universität Erlangen.

[Moore und Paris, 1989] Johanna D. Moore and Cécile L. Paris. 1989. Planning text for advisory dialogues. In *Proceedings of ACL, US Chapter*, Vancouver.

[Nagata und Morimoto, 1993] Masaaki Nagata und Tsuyoshi Morimoto. 1993. An experimental statistical dialogue model to predict the speech act type of the next utterance. In *International Symposium on Spoken Language*, Waseda University.

[Niedermair, 1992] Gerhard Th. Niedermair. 1992. Linguistic Modelling in the Context of Oral Dialogue. In *Proceedings of International Conference on Spoken Language Processing (ICSLP'92)*, volume 1, pages 635–638, Banff, Canada.

[Searle, 1969] John R. Searle. 1969. *Speech Acts.* Cambridge/GB: University Press.

[Sitter und Stein, 1992] Stefan Sitter und Adelheit Stein. 1992. Modelling the illocutionary aspects of information-seeking dialogues. *Information Processing and Management*, 28(2):165 – 180.

[Young et al., 1989] Sheryl R. Young, Wayne H. Ward, und Alexander G. Hauptmann. 1989. Layering predictions: flexible use of dialog expectation in speech recognition. In *Proceedings of IJCAI '89, Detroit.*

Kapitel 7

Inkrementelle Generierung in Verbmobil

Wolfgang Finkler, Anne Kilger
DFKI GmbH
Deutsches Forschungszentrum für Künstliche Intelligenz
Stuhlsatzenhausweg 3, D – 66123 Saarbrücken
Tel: (0681) 302 5269|5255, Fax: (0681) 302 5341

E–mail: finkler|kilger@dfki.uni–sb.de

Zusammenfassung

Es werden besondere Aspekte inkrementeller Verarbeitung natürlicher Sprache beschrieben, die damit zu tun haben, daß ein inkrementeller Modul ohne Wissen über die vollständige Eingabe, die er bei einem Aufruf bearbeiten soll, zu jedem Zeitpunkt Entscheidungen treffen muß, die im Konfliktfall evtl. nur schwer rückgängig gemacht werden können.

7.1 Inkrementalität

Da die Akzeptanz von natürlichsprachlichen Systemen bei Mensch–Maschine Schnittstellen stark davon abhängt, wie gut menschliches Sprachverhalten nachgebildet ist, sollten neben Grammatikalität und inhaltlicher Adäquatheit auch Eigenschaften des zeitlichen Verlaufs der Sprachproduktion berücksichtigt werden. Psycholinguistische Untersuchungen auf diesem Gebiet führen den Begriff inkrementelle Generierung in einem Modell der spontansprachlichen Satzgenerierung ein:

> "An important characteristic of spontaneous speech is that overt pronunciation of a sentence can be initiated before the speaker has completely worked out the conceptual content he is going to express in that sentence.

[1]Die vorliegende Arbeit wurde im Rahmen des Verbundvorhabens Verbmobil vom Bundesministerium für Forschung und Technologie (BMFT) unter dem Förderkennzeichen 01IV101K/1 gefördert. Die Verantwortung für den Inhalt dieser Arbeit liegt bei den Autoren.

> Apparently, the speaker is able to build up a syntactically coherent utterance out of a series of syntactic fragments each rendering a new part of the conceptual content."
>
> [Kempen & Hoenkamp 82, S. 151]

Mit dem Begriff ‚inkrementelle Verarbeitung' wird im wesentlichen das Eingabe–Ausgabeverhalten einer Komponente charakterisiert. Es handelt sich um einen verzahnten Ablauf von Eingabekonsumierung und Ausgabeproduktion (vgl. [Kilger & Finkler 95]):

> Jedes als Teileingabe spezifizierte Element muß seine unmittelbare Verarbeitung auslösen, so daß sich insgesamt folgendes Verhalten ergibt:
>
> - Die Verarbeitung beginnt, bevor die Eingabe vollständig ist,
> - erste Teile der Ausgabe werden erzeugt, bevor die Verarbeitung abgeschlossen ist und sogar
> - bevor die Eingabe vollständig vorliegt.

In der Informatik wurde der Begriff inkrementelle Verarbeitung im Kontext des Übersetzerbaus eingeführt (vgl. [Lock 65], [Katzan Jr. 69]), wobei herausgearbeitet wurde, daß kleine Erweiterungen in der Eingabe nicht unbedingt ein vollständiges Neucompilieren nach sich ziehen sollten. In diesen Arbeiten wurden überwiegend Fälle betrachtet, in denen die Inkremente in der Eingabe unabhängig voneinander bearbeitet werden konnten und das Ergebnis sich unmittelbar aus den Einzelergebnissen für die Inkremente ergab (kumulative Verarbeitung). Dies ist für die Verarbeitung natürlicher Sprache nicht gültig. Während der Formulierung müssen komplexe Einflüsse einzelner Entscheidungen auf Realisierungsmöglichkeiten weiterer Eingabeelemente berücksichtigt werden. So sind beispielsweise bei der inkrementellen Generierung der Aufbau der syntaktischen Struktur sowie die Positionierung einzelner Elemente gemäß der Wortstellungsregeln einer Sprache auch davon abhängig, wie vorher bearbeitete Teilstrukturen realisiert, und ob sie eventuell sogar bereits artikuliert wurden. Dies bedeutet gegenüber inkrementellen Ansätzen zur syntaktischen oder semantischen Analyse (siehe z.B. [Haddock 87], [Wirén 92]) die zusätzliche Anforderung an die Verarbeitung, bereits erzeugte Teilergebnisse in der weiteren Verarbeitung möglichst fortzusetzen (vgl. [Finkler & Schauder 92]).

7.2 Vorteile inkrementeller Generierung

Inkrementelle Generierung bietet die Chance, einen globalen Laufzeitgewinn zu erzielen, da die Verarbeitung offensichtlich parallelisiert werden kann. Ein inkrementeller Modul kann bereits Teile seiner Aufgabenstellung bearbeiten, während seine Eingabespezifikation von einem anderen Modul zur selben Zeit vervollständigt wird. Es ist sogar möglich, daß mit der Konzeptualisierung der nächsten Äußerung begonnen wird, bevor die Formulierung der aktuellen Äußerung abgeschlossen wurde.
In einem inkrementellen Generierungssystem wird gegenüber einem seriellen System eine Verkürzung der initialen Verzögerung vor Äußerungsbeginn erreicht. Das Merkmal menschlicher Sprachproduktion, daß bereits mit dem Reden angefangen werden kann, bevor der gesamte Inhalt einer Äußerung im Detail geplant ist, ist in kommunikativen Situationen mit mehreren menschlichen Dialogpartnern wichtig, um etwa in

einer Diskussion das Wort zu ergreifen. Auch in einer Dialogsituation zwischen Computer und menschlichem Benutzer führt ein promptes Antworten des Systems zu einer höheren Akzeptanz[1]. Prompte Systemreaktion ist erst recht wichtig, wenn das Computersystem mittels angeschlossener Sprachsynthesekomponente gesprochene Sprache ausgeben soll. In solchen Anwendungen ist vom menschlichen Benutzer ständige Aufmerksamkeit verlangt, damit er keine Systemäußerung verpaßt.

Die Fähigkeit menschlicher Sprachverwendung, Sprache flüssig zu produzieren, wird ebenfalls in einem System zur inkrementellen Sprachproduktion nachvollziehbar. In einem solchen Verarbeitungsmodell ist es plausibel, daß signifikant lange Pausen nicht zwischen einzelnen Sätzen auftreten, wie es etwa in einem seriellen Modell bei einem Satzgenerator zu erwarten ist. Hingegen können bei einem inkrementellen Generator Pausen innerhalb einzelner Sätze auftreten. Sie können konzeptuelle Ursachen haben und sind auch in spontaner Sprachproduktion des Menschen beobachtbar (vgl. [Maclay & Osgood 59], [Goldman-Eisler 68]). Sie resultieren beispielsweise aus der zeitweiligen Unvollständigkeit der inkrementellen Eingabespezifikation für die Generierungskomponente.

Ein weiterer wichtiger Vorteil, der sich aus der Fähigkeit ergibt, fragmentarische Eingabe zu verarbeiten, ist die hohe Flexibilität eines inkrementellen Systems. Wenn ein solches System in der Lage ist, nicht nur auf Additionen von Eingabespezifikationen sondern sogar auf Ersetzungen oder Löschungen vorheriger Eingabespezifikationen zu reagieren, so erlaubt dies eine neue Qualität an Interaktion mit der aufrufenden Komponente. Die konzeptuelle Komponente des Generators, die den Inhalt einer Äußerung festlegt, kann die Verbalisierung von Änderungen nach dem Start des inkrementellen Generators ansteuern und sogar, nachdem der Anfang der Äußerung artikuliert wurde. Solche konzeptuellen Änderungen treten etwa beim Simultandolmetschen auf, wenn Ausgabeproduktion in einer sich für den Dolmetscher unerwartet ändernden Situation schritthaltend erzeugt werden muß.

7.3 Der Aspekt der Unsicherheit bei inkrementeller Verarbeitung und Folgen für die Generierung

Aufgrund der temporären Unvollständigkeit der Eingabe arbeitet ein inkrementelles System ständig in Unsicherheit darüber, was weiterhin als Eingabe spezifiziert werden wird. Bei der Verarbeitung der ersten Eingabeelemente muß darauf geachtet werden, daß Fortsetzungen der Eingabe im Zwischenergebnis integrierbar bleiben. Da inkrementelle Ausgabe in zeitlicher Verzahnung mit der Eingabe erzeugt werden soll, müssen Entscheidungen getroffen werden, die bestimmte Eigenschaften möglicher Fortsetzungen der Eingabe erfordern. Ein solches Problem ergibt sich bei der Übersetzung vom Japanischen ins Englische. Im Japanischen wird Negation am Satzende spezifiziert, während sie beim Englischen vor dem finiten Verb stehen muß. Kann mit der Übersetzung aber nicht bis zum Ende des Satzes gewartet werden, so muß als Arbeitshypothese einer der Fälle +Negation oder −Negation angenommen werden. Solche Annahmen (vgl. [Harbusch et al. 94]) können im Widerspruch zu später tatsächlich eingegebenen Elementen stehen. Dann kommt es zu suboptimalen Ergebnissen, eventuell sogar zu offenen Reparaturen in der Ausgabe ([Finkler & Schauder 92]).

[1] Siehe [Rubinstein & Hersh 94] für Untersuchungen zur Mensch–Maschine Kommunikation.

Aus dieser Diskussion ergibt sich die Forderung nach adäquaten Strategien für die folgenden Aufgaben bei der inkrementellen Generierung:

- *Entscheidungen auf der Basis unvollständiger Eingaben* erfordern Strategien für die Auswahl alternativer Arbeitshypothesen, z.B. die Ausnutzung interner Inferenzmöglichkeiten wie das Schließen auf die wahrscheinlichsten Fortsetzungen aus dem Kontext.

- Mit zunehmender Eingabe steigt die Verläßlichkeit einer produzierbaren Ausgabe. Allerdings muß dieser Vorteil gegen das Problem abgewogen werden, daß eine zu lange Verzögerung innerhalb einer Äußerung inadäquat ist. Neben den in der Literatur zu Generierungssystemen diskutierten Problemstellungen des What–to–Say, Festlegung des Inhalts eines Gesprächsbeitrags, und How–to–Say, Realisierung der intendierten Bedeutung, (vgl. [Thompson 77]) spielt *When–to–Say*, die Entscheidung über Äußerungszeiten von Satzfragmenten eine entscheidende Rolle bei der inkrementellen Generierung.

- Will man inkrementelle Ausgabe produzieren, muß man selbst im einfachen Fall der Spezifikation zusätzlicher Eingabeinkremente in der Lage sein, Reparaturen zu realisieren, wenn später eingegebene Inkremente in den bereits geäußerten Satzanfang integriert werden müßten. Das Problem wird noch verschärft, wenn man mit Reparaturen wie Löschungen oder Ersetzungen in der Eingabe zu rechnen hat. Adäquate *Reparaturstrategien*, die zu akzeptablen und verständlichen Äußerungen führen, sind deswegen ein Kernpunkt bei der Produktion inkrementeller Ausgabe (vgl. [Finkler 95]).

7.4 Inkrementelle Generierung in Verbmobil

Im Rahmen des VERBMOBIL–Projektes (vgl. [Wahlster & Engelkamp 92]) wird ein inkrementeller Generator für natürlichsprachliche Äußerungen entwickelt. Ziel des Gesamtprojektes ist die Entwicklung eines mobilen Systems zur Übersetzung von Verhandlungsdialogen in face–to–face Situationen. Jede quellsprachliche Äußerung der Dialogpartner wird analysiert und von einer Transferkomponente in eine semantische Repräsentation für die Äußerung in der Zielsprache Englisch überführt, welche die Verhandlungssprache der Dialogpartner darstellt. Der Generator formuliert die zielsprachliche Äußerung, eine Sprachsynthesekomponente artikuliert sie.
Inkrementalität als Grundprinzip des VERBMOBIL–Generators läßt sich mit den oben beschriebenen Effizienz– und Flexibilitätsvorteilen und den Rahmenbedingungen der zeitkritischen Anwendung Dolmetschen motivieren. Als Konsequenz dieser Verarbeitungsstrategie müssen die einzelnen Moduln des inkrementellen Systems stark interagieren, um im Konfliktfall oder statt Anwendung von Defaultregeln Rückfragen an die jeweils aufrufende Komponente zu ermöglichen. Der VERBMOBIL–Generator besteht aus Komponenten zur Mikroplanung, zur Wortwahl und zur syntaktischen Generierung, die paarweise bidirektional verbunden sind (vgl. Abbildung 7.1). In einem Durchlauf ohne Rückwirkung soll in der Mikroplanung auf der Basis einer übergebenen ‘Botschaft’ (semantische und pragmatische Information zu der zu generierenden Äußerung) unter anderem die grobe Satzstruktur festgelegt werden (Satztyp, Tempus, Modus, …). Weiterhin wird über die Realisierung komplexer Konzepte als Verbalphrase oder in nominalisierter Form entschieden.

In der Wortwahlkomponente werden konzeptuelle Eingaben in Lemmata der Zielsprache überführt. Wortwahl findet ebenfalls auf der Grundlage semantischer und pragmatischer Information sowie mit den Vorgaben der Mikroplanung (z.B. bez. der Kategorie des zu wählenden Wortes) statt. Hier werden nicht nur situationsadäquate Wörter sondern auch idiomatische Ausdrücke gewählt.
Mikroplanung und Wortwahl sind voneinander abhängig. Die Mikroplanung kann die Wahl eines Wortes einer bestimmten Kategorie fordern, die Wortwahl kann eine solche Forderung als nicht erfüllbar zurückweisen oder als schlecht einstufen und andere Vorgaben verlangen. Diese bidirektionale Abhängigkeit ist durch den Doppelpfeil in Abbildung 7.1 verdeutlicht.

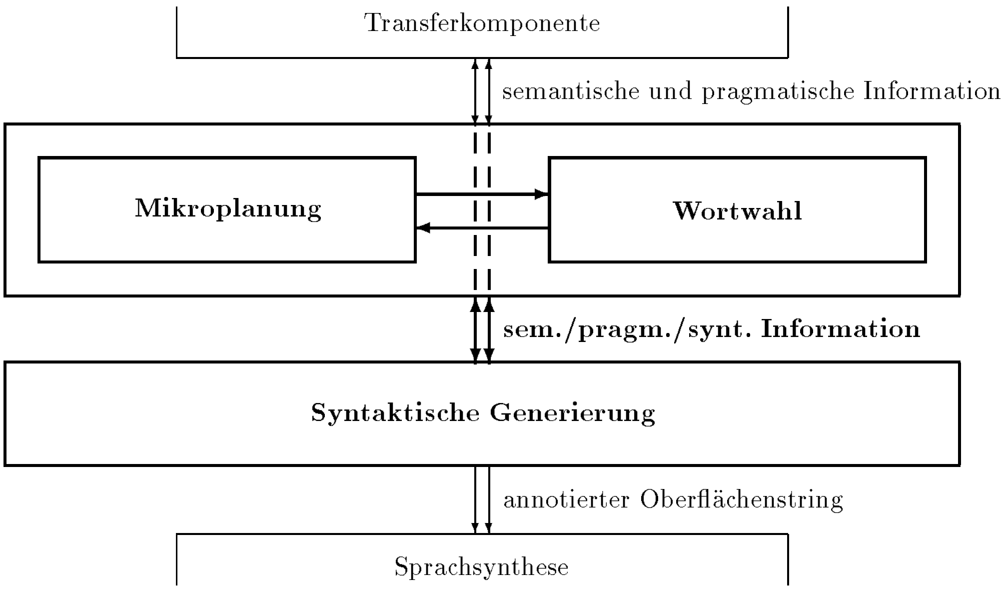


Abbildung 7.1: Grobdarstellung der Systemarchitektur von VM–GEN

Die Ergebnisse der Mikroplanung ergeben ein syntaktisches Schema, das um die Ergebnisse der Wortwahl angereichtert und zusammen mit semantischer und pragmatischer Information an den *syntaktischen Generator* übergeben wird. Die semantische Information betrifft z.B. semantische Beziehungen zwischen den Wörtern und Phrasen. Sie wird beim Aufbau der globalen hierarchischen Struktur für die Äußerung in entsprechende syntaktische Beziehungen umgesetzt (z.B. wird ,,agent" in der Regel ,,subject"). Semantische und pragmatische Information wird für die Auswahl der geeignetsten Strategie zur Bestimmung der Wortstellung verwendet. Im syntaktischen Teil des Generators von VERBMOBIL wird eine hierarchische Oberflächenstruktur für die Äußerung aufgebaut, die Terminale werden gemäß Wortstellungsregeln des Englischen angeordnet, und die inkrementelle Ausgabe fertiggestellter Äußerungsteile wird durch ihre Übermittlung an eine Sprachsynthesekomponente veranlaßt.
Rückflüsse von Information zwischen den Moduln entstehen aus verschiedenen Gründen. Durch den modularen Aufbau des Systems sind die einzelnen Komponenten

nicht in der Lage, Konsequenzen lokal getroffener Entscheidungen auf die nachfolgende Komponente zu antizipieren. Arbeitshypothesen in einer Komponente können während der späteren Verarbeitung falsifiziert werden. Eine entsprechende Rückmeldung muß an die höhere Komponente ergehen und eine Neuberechnung auslösen. Die zweite Ursache für Rückwirkungen liegt in der Eigenschaft inkrementeller Verarbeitung, Entscheidungen auf der Basis unvollständiger Information zu treffen. Führen solche Entscheidungen zur Äußerung von Satzteilen, die nicht kompatibel mit später eintreffenden Daten sind, so bestehen grundsätzlich zwei Reaktionsmöglichkeiten: Der syntaktische Generator, der die inkrementelle Artikulation ansteuert, kann entweder versuchen, lokal eine Reparatur der Äußerung zu produzieren (z.B. ,,Herr Maier möchte dem Händler einen PC ... äh ... von dem Händler einen PC kaufen") oder darüberliegende Komponenten auffordern, eine Alternative zu liefern, die die aktuellen syntaktischen Restriktionen erfüllt (z.B. ,,Herr Maier möchte dem Händler einen PC ... abkaufen").

Während wir derzeit an den Konzepten für Mikroplanung und Wortwahl arbeiten, konnten wir mit der Realisierung des inkrementellen syntaktischen Generators VM–GEN, der auf eigene Vorabeiten im Projekt WIP (vgl. [Wahlster et al. 93]) aufbaut, bereits wichtige Erfahrungen im Bereich inkrementeller natürlichsprachlicher Generierung sammeln. Die flexible Eingabeschnittstelle zu VM–GEN erlaubt die Spezifikation von lexikalischen Elementen und semantischen Relationen zwischen ihnen in beliebiger Reihenfolge und mit beliebigen Verzögerungen. Die technische Möglichkeit, beliebige Permutationen der Eingabepakete zu übergeben, wird im Rahmen eines umgebenden Systems wahrscheinlich nicht ausgenutzt werden. Vielmehr sind eingeschränkte Varianten zu erwarten, z.B. die Übergabe des Satztyps vor der Spezifikation einzelner lexikalischer Einheiten ([Clark & Clark 77]).
VM–GEN ist in der Lage, auf der Grundlage der inkrementellen Eingabe syntaktisch korrekte Äußerungen inkrementell zu produzieren, sofern sich Probleme (siehe oben) lokal korrigieren lassen. Die Basis von VM–GEN bildet ein verteiltes paralleles Modell aktiver kooperierender Objekte, welche jeweils für die Verbalisierung eines Eingabepakets, z.B. (ENTITY NP-1 (HEAD "Termin") (NUM sg)), verantwortlich sind. Parallelität wird hier nicht verwendet zur Realisierung von Wettbewerb zwischen konkurrierenden Varianten sondern zur Ermöglichung einer unabhängigen und simultanen Bearbeitung separater Teilstrukturen (z.B. können die Anbindungen verschiedener Artikel oder Attribute an verschiedene Nomen zeitgleich erfolgen). Das System läuft sowohl mit simulierter Parallelität auf einer Einprozessoranlage unter LISP als auch echt parallel in einem Rechnernetz. Die Objekte kommunizieren per Nachrichtenaustausch und bewältigen auf diese Weise ihre Aufgaben, die globale hierarchische Struktur virtuell zusammenzubauen, die Terminale anzuordnen und fertige Ausgabeteile an einen Artikulator zu übergeben. Die Vernetzung der Objekte in einer Hierarchie von Regenten und Dependenten, die festlegt, wer als Adressat von Nachrichten dienen kann, wird dazu verwendet, in speziellen Fällen, z.B. bei der Ausgabe von Äußerungsteilen, die Arbeit der Objekte zu synchronisieren.
Die syntaktischen Regeln für VM–GEN sind deklarativ in einer Erweiterung des Formalismus Tree Adjoining Grammar (siehe beispielsweise [Joshi 85]) um Unifikation und gesonderte Linearisierungsspezifikationen repräsentiert (vgl. [Kilger 95]). Die Grammatik ist lexikalisiert, d.h. lexikalische Elemente sind mit syntaktischen Regeln verbunden, in denen etwa Bedingungen ihrer obligatorischen Argumentpositionen formuliert sind. Lexikalische Elemente in der inkrementellen Eingabe werden zunächst in einem bottom–up Schritt zum Zugriff auf syntaktische Regeln verwendet. Die Grammatik ist als Vererbungshierarchie organisiert und erlaubt so bei unvollständiger Eingabe die

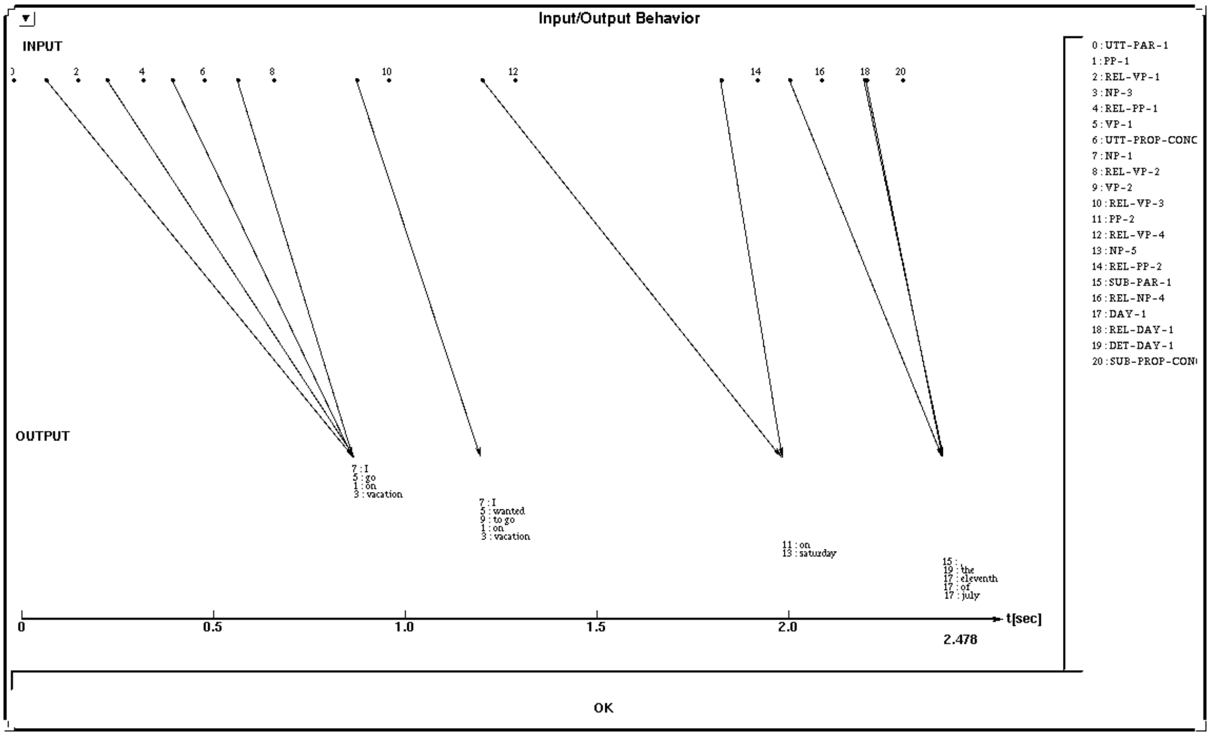


Abbildung 7.2: Bildschirmabzug eines Systemlaufs zur Verdeutlichung der Verzahnung von Eingabe– und Ausgabezeitpunkten

Auswahl unterspezifizierter Strukturen. Beim verteilten Aufbau der globalen hierarchischen Struktur der zu generierenden Sätze werden je nach Operation in der Grammatik Kopien von Adjunktionsknoten, Wurzel– und Fußknoten auxiliarer Bäume (Adjunktionsoperation) bzw. Kopien von Substitutionsknoten und Wurzeln von Substitutionsbäumen (Substitution) zwischen den beteiligten Objekten verschickt, um nicht nach dem Zusammenbau zweier Strukturen die Parallelität zu verlieren[2] (vgl. [Kilger 94]).

Die in Abschnitt 7.3 angesprochenen besonderen Aspekte inkrementeller Verarbeitung werden in VM–GEN auf die folgende Weise behandelt: *Entscheidungen auf der Basis unvollständiger Information* werden durch ein besonderes Grammatikdesign unterstützt, das in der Definition kleinster Strukturen besteht, z.B. werden zu lexikalischen Elementen nur die Valenzrahmen bis zu den syntaktischen Funktionen der Ergänzungen spezifiziert. Zur Frage des *When–to–Say* müssen erst Untersuchungen angestellt werden. Derzeit werden jeweils an der aktuellen Ausgabeposition mögliche Fortsetzungen der Äußerung unmittelbar an den Artikulator übergeben. Um inkrementelle Ausgabe zu ermöglichen, sind einfache *Reparaturstrategien* implementiert worden.

[2]Details zur Verarbeitung im syntaktischen Generator können in [Kilger & Finkler 95] weiterverfolgt werden.

Sie beschränken sich allerdings auf die Wiederholung der zu korrigierenden (Teil–) Äußerungen, z.B. „Herr Maier möchte dem Händler einen PC ... Herr Maier möchte von dem Händler einen PC kaufen". Eine feinere Untergliederung der Reparaturfälle wird zur Zeit bearbeitet.

Literaturverzeichnis

[Clark & Clark 77] H.H. **Clark** and E.V. **Clark**. *Psychology and Language.* New York: Harcourt Brace Jovanovich, 1977.

[Finkler & Schauder 92] W. **Finkler** and A. **Schauder**. *Effects of Incremental Output on Incremental Natural Language Generation.* In: B. Neumann (ed.), Tenth European Conference on Artificial Intelligence, pp. 505–507, Vienna, Austria, August 1992. Wiley.

[Finkler 95] W. **Finkler**. *Automatische Selbstkorrektur bei der inkrementellen Generierung natürlicher Sprache unter Realzeitbedingungen.* Technical report, German Research Center for Artificial Intelligence (DFKI), 1995. (to appear).

[Goldman-Eisler 68] F. **Goldman-Eisler**. *Psycholinguistics: Experiments in Spontaneous Speech.* New York: Academic, 1968.

[Haddock 87] N. **Haddock**. *Incremental Interpretation and Combinatory Categorial Grammar.* In: Tenth International Joint Conference on Artificial Intelligence, pp. 661–663, Milan, Italy, 1987.

[Harbusch et al. 94] K. **Harbusch**, G. **Kikui**, and A. **Kilger**. *Default Handling in Incremental Generation.* In: Proceedings of the 15th International Conference on Computational Linguistics (COLING–94), pp. 356–362, Kyoto, Japan, August 1994.

[Joshi 85] A.K. **Joshi**. *How Much Context–Sensitivity is Necessary for Characterization Structural Descriptions – Tree Adjoining Grammar.* In: D. Dowty, L. Karttunen, and A. Zwicky (eds.), Natural Language Processing – Theoretical, Computational and Psychological Perspective. New York: Cambridge University Press, 1985.

[Katzan Jr. 69] H. **Katzan Jr.** *Batch, conversational, and incremental compilers.* In: American Federation of Information Processing Societies (AFIPS): Proc. Spring Joint Computer Conference, May 14-16, Boston, Massachusetts, volume 34, pp. 47–56, Montvale, New Jersey, USA, 1969. AFIPS Press.

[Kempen & Hoenkamp 82] G. **Kempen** and E. **Hoenkamp**. *Incremental Sentence Generation: Implications for the Structure of a Syntactic Processor.* In: Nineth International Conference on Computational Linguistics. North–Holland Publishing Company, 1982.

[Kilger & Finkler 95] A. **Kilger** and W. **Finkler**. *TAG–Based Incremental Generation.* Technical report, German Research Center for Artificial Intelligence (DFKI), 1995. (to appear). submitted to Computational Linguistics.

[Kilger 94] A. **Kilger**. *Using UTAGs for Incremental and Parallel Generation.* Computational Intelligence, 10(4), 1994.

[Kilger 95] A. **Kilger**. *TAGs with Context–Dependent Disjunctive Linearization Rules.* Technical report, German Research Center for Artificial Intelligence (DFKI), 1995. (to appear).

[Lock 65] K. **Lock**. *Structuring Programs for Multiprogram Time–Sharing On–Line Applications.* In: American Federation of Information Processing Societies (AFIPS): Proc. Fall Joint Computer Conference, volume 27, Part 1, pp. 457–472, Montvale, New Jersey, USA, 1965. AFIPS Press.

[Maclay & Osgood 59] H. **Maclay** and C. **Osgood**. *Hesitation Phenomena in Spontaneous English Speech.* Word, 15:19–44, 1959.

[Rubinstein & Hersh 94] R. **Rubinstein** and H. **Hersh**. *The Human Factor: Designing Computer Systems for People.* Digital Press, 1994.

[Thompson 77] H. **Thompson**. *Strategy and Tactics in Language Production.* In: W. A. Beach, S. E. Fox, and S. Philosoph (eds.), Papers from the Thirteenth Regional Meeting of the Chicago Linguistics Society, Chicago, April 1977.

[Wahlster & Engelkamp 92] W. **Wahlster** and J. **Engelkamp**. *Wissenschaftliche Ziele und Netzpläne für das VERBMOBIL–Projekt.* Dokument, German Research Center for Artificial Intelligence (DFKI), Saarbrücken, Germany, April 1992.

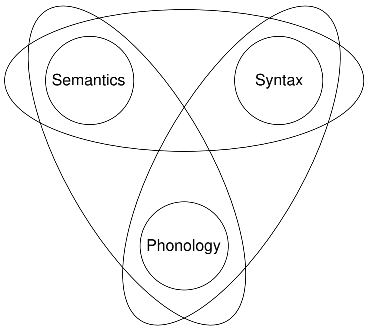
[Wahlster et al. 93] W. **Wahlster**, E. **André**, W. **Finkler**, H.-J. **Profitlich**, and T. **Rist**. *Plan-based Integration of Natural Language and Graphics Generation.* Artificial Intelligence, 63:387–427, 1993.

[Wirén 92] M. **Wirén**. *Studies in Incremental Natural–Language Analysis.* PhD thesis, Department of Computer and Information Science, Linköping, Sweden, 1992. Dissertation No. 292.

Kapitel 8

Inkrementbasierte Verarbeitung von Informationsstrukturen im Sprachproduktionssystem SYNPHONICS

Carsten Günther, Claudia Maienborn, Andrea Schopp, Soenke Ziesche
Universität Hamburg
FB Informatik, WSV
Vogt–Kölln–Str. 30, 22527 Hamburg
`E--Mail: guenther, schopp, ziesche@informatik.uni--hamburg.de`
`Claudia=Maienborn@rz.hu-berlin.de`



[1]Das Sprachproduktionssystem SYNPHONICS (**syn**tactic und **phon**ological realization of **i**ncrementally generated **c**onceptual **s**tructures) wird im Rahmen eines von der DFG unter dem Förderzeichen HA–1237/4–2 geförderten Projekts entwickelt. Wir danken Ingo Schröder für die LaTeX–Aufbereitung.

8.1 Einleitung

Im Rahmen des Projekts SYNPHONICS erfolgt eine Modellierung des Sprachproduktionsprozesses von der konzeptuellen Inhaltsplanung bis zur lautsprachlichen Realisierung, die drei linguistische Herangehensweisen verknüpft: Es werden Resultate der Psycholinguistik berücksichtigt, um nicht irgendwie Sprache zu produzieren, sondern die diesbezüglichen Vorgänge beim Menschen zu simulieren, nicht zuletzt deshalb, weil sich diese durch Effizienz und Robustheit auszeichnen. Zudem werden die aktuellen Forschungen der theoretischen Linguistik integriert, was sich u.a. darin manifestiert, daß eine deklarative Repräsentation des grammatischen Wissens nach der HPSG–Theorie (vgl. Pollard & Sag (1987, 1994)) vorliegt. Schließlich handelt es sich um einen computerlinguistischen Ansatz mit dem Ziel einer teilweise bereits realisierten Implementation. Die Verarbeitung findet dabei auf der Grundlage eines Unifikationsformalismus statt, der auf getypte Merkmalsstrukturen angewandt wird.
Die zugrundeliegende Systemarchitektur ist gekennzeichnet durch die globalen Eigenschaften:

- Modularität
- Inkrementalität
- Rückkopplungsfreiheit

Die SYNPHONICS zugrundeliegende Modularitätsannahme betrifft verschiedene Dimensionen der Sprachverarbeitung: Sie schlägt sich zum einen nieder in einer strikten Unterscheidung von deklarativem und prozeduralem Wissen und führt zum anderen zur Identifikation und Abgrenzung unterschiedlicher Verarbeitungsebenen beim Aufbau sprachlicher Strukturen. Abb. 8.1 zeigt eine schematische Darstellung des Architekturmodells von SYNPHONICS:
Wir nehmen die globalen Ebenen der Konzeptualisierung, Formulierung und Artikulation für die außersprachliche, sprachliche bzw. artikulatorische Verarbeitung an (vgl. Levelt (1989)), auf denen die im folgenden sehr verkürzt dargestellten Abläufe stattfinden[1]:
Der *Konzeptualisierer* greift auf eine Referenzobjekte (RefO) enthaltende konzeptuelle Wissensbasis zu und erzeugt zwei Ausgabestrukturen: inkrementell die zu äußernde konzeptuelle Struktur CS sowie eine für die Äußerung relevante Kontextstruktur CT. Diese werden vom *Formulator* übernommen, der die semantische, syntaktische und phonologische Kodierung durchführt. Zunächst erzeugt der *Semantische Enkodierer* aus CS und CT eine einzelsprachspezifische semantische Repräsentation SEM, wobei u.a. Berechnungen zur Informationsgliederung durchgeführt werden. Diese Ausgabe SEM ist dreigeteilt: Die Referenz– und die Kerninformation fließen in den *Lemma–Selektor*, der der Lemma–Partition des Lexikons ein geeignetes Lemma entnimmt und dieses vom *Lizensierer* lizensieren läßt. Simultan konstruiert der *Schema–Selektor* bereits abstrakte syntaktische Strukturen, basierend auf der Einbettungsinformation der SEM und Informationen einer Schema–Wissensbasis. Der Output von Schema–Selektor und Lemma–Selektor wird daraufhin im *Integrator* zusammengeführt, in das sogenannte aktuelle Äußerungsfragment integriert und einer Wohlgeformtheitskontrolle des *Lizensierers* unterzogen. Das *Floodgate* übernimmt ein vom Integrator angebotenes aktuelles Äußerungsfragment dann, wenn es für dessen Zwecke hinreichend spezifiziert

[1] Für eine ausführliche Beschreibung siehe Günther (ed., 1994) und Abb et al. (ersch.).

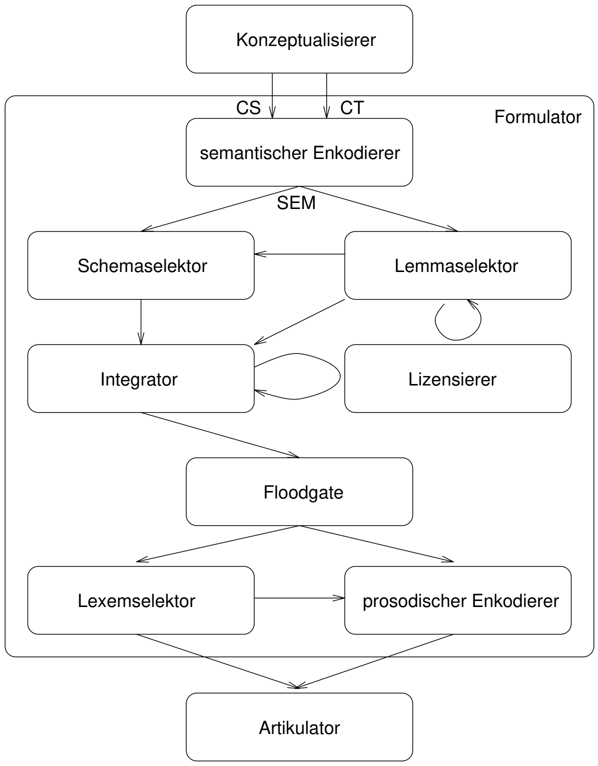


Abbildung 8.1: Das Sprachproduktionssystem SYNPHONICS

ist, und leitet es an den Lexem–Selektor und den prosodischen Enkodierer weiter. Der *Lexem–Selektor* selektiert aus der Lexem–Partition des Lexikons ein Lexem, dessen segmentale Merkmale direkt an den Artikulator gereicht werden, indes die prosodischen Merkmale in den *prosodischen Enkodierer* einfließen. In diesem prosodischen Enkodierer wird daraufhin die makroprosodische Planung realisiert. Abschließend wird von den Untermodulen des *Artikulators* die phonetisch–artikulatorische Enkodierung vorgenommen, die sich in Form akustischer, von einem *Sprachsynthetisator* ausgegebener Sprachsignale manifestiert.
Im folgenden wird die inkrementelle Verarbeitung von zu verbalisierenden Informationsfragmenten zunächst allgemein und dann konkret anhand der Berechnung des weiten Fokus während der semantischen Enkodierung dargestellt.

8.2 Einführung in die inkrementelle Verarbeitung

Den Schwerpunkt der weiteren Ausführungen bilden Überlegungen zur Inkrementalität: Im Rahmen eines kognitiv orientierten Sprachproduktionssystems gilt es zum einen, Bedingungen hinsichtlich Aufbau, Größe, Vernetzung und Reihenfolge von Inkrementen zu bestimmen und zum anderen, unter diesen Vorgaben eine effiziente, adäquate Informationsverarbeitung zu gewährleisten.
Jede Prozeßkomponente des Sprachproduktionsmodells verfügt über ein eigenes Re-

pertoir an Regeln und Prinzipien und richtet dementsprechend spezifische Wohlgeformtheitsbedingungen an Form und Umfang der zu verarbeitenden Wissensstrukturen. Erfüllt ein eintreffendes Informationsfragment (Inkrement) die jeweiligen modulspezifischen Auflagen zur Weiterverarbeitung nicht, so wird es zur weiteren Informationsanreicherung durch nachfolgende Inkremente in einem Schnittstellenpuffer (z.B. das Floodgate, s. Abb. 8.1) zwischengespeichert, bis der charakteristische Eingabe vorliegt. Diese Annahme bildet eine wesentliche Voraussetzung für die Entwicklung eines Verarbeitungsmodells, das auf den Einsatz von Rückkopplung verzichten kann.
Im Rahmen der SYNPHONICS–Konzeption wird Inkrementalität damit als ein Phänomen aufgefaßt, das dynamisch veränderbare, durch die jeweilige Prozeßumgebung determinierte Verarbeitungseinheiten hervorbringt. Inkrementelle Verarbeitung bezieht sich also nicht auf fragmentarische Information, die im gesamten Verlauf des Sprachproduktionsprozesses vom Umfang her stabil und jederzeit zusammenhängend lokalisierbar ist, sondern involviert zeitabhängig variierende und ggf. über parallel angeordnete Verarbeitungskomponenten verteilte, d.h. nicht notwendigerweise lokal fixierte, Datenstrukturen. Die zu versprachlichende Information, die mit einem Referenzobjekt zu einem bestimmten Zeitpunkt assoziiert ist, verändert sich nicht nur aufgrund der Weitergabe an eine nachgeordnete Verarbeitungskomponente, die ein entsprechend anderes Datenformat aufweist, sondern kann sich auch innerhalb einer Komponente wandeln: Bei der parallel verlaufenden grammatisch–strukturellen und lexikalischen Aufbereitung im Formulator etwa verteilt sich ein Inkrement auf zwei Verarbeitungszweige (Lemma– und Schema–Selektor, s. Abb. 8.1), wobei der Zusammenhang aber über die identische referentielle Spezifikation jederzeit gewährleistet ist und zu einem späteren Verarbeitungszeitpunkt wiederhergestellt wird. Die Identifizierbarkeit von RefOs über Referenzindizes sowie ihre Vernetzung mittels einbettender relationaler Information (beispielsweise Information darüber, welche thematische Rolle ein Objektreferent in einer Situation einnimmt) bilden die Grundlage für die inkrementelle Verarbeitung partieller Datenstrukturen in SYNPHONICS.
Um der angestrebten Robustheit des Systems entsprechend dem menschlichen Vorbild Rechnung zu tragen, sehen wir vor, daß sowohl die referentielle als auch die relationale Verankerung eines Inkrements zu verschiedenen Verarbeitungszeitpunkten unterbestimmt sein kann. Dies unterstützt eine schnelle und flexible Versprachlichung konzeptueller Strukturen. Voraussetzung für einen reguläre sprachliche Strukturbildung ist hierbei die kohärente Vernetzung der Inkremente. Ist diese Kohärenz nicht gegeben, so erfolgt wiederum orientiert am menschlichen Vorbild die Ausgabe einer nicht–wohlgeformten sprachlichen Struktur oder ggf. sogar der Abbruch der Äußerung.
Die Inkrementreihenfolge wird in SYNPHONICS als konzeptuell gesteuert angenommen. Abfolgevarianzen von Inkrementen werden durch unterschiedliche konzeptuelle Ausgangsbedingungen ausgelöst und bedingen ihrerseits eine Umsetzung durch geeignete sprachliche Strukturmittel (z.B. Topikalisierung, Scrambling, Rechtsextraposition, Passivierung). Inkremente sind somit nicht beliebig permutierbar; vielmehr wird ihre sequentielle Anordnung auf der konzeptuellen Ebene durch konzeptuelle und kontextuelle Prominenzverhältnisse festgelegt, die eine mehrdimensionale Strukturierung der Information (Topik/Kommentar–Gliederung, Fokus/Hintergrund–Gliederung) bewirken. Der Zusammenhang zwischen Inkrementalität und den verschiedenen Ebenen der Informationsstrukturierung erweist sich als äußerst aufschlußreicher Gegenstandsbereich für die Modellierung eines Sprachproduktionssystems: Zum einen nimmt Informationsgliederung Einfluß auf Größe und Reihenfolge von Inkrementen (Topik/Kommentar), zum anderen sind Vorkehrungen dafür zu treffen, daß die Berech-

nung globaler, nämlich satzbezogener, Informationsstrukturen (Fokus/Hintergrund) und ihre prosodische Umsetzung unter den Bedingungen der Inkrementalität des Verarbeitungsprozesses vollzogen werden kann.

8.3 Illustration der inkrementellen Verarbeitung anhand der Berechnung des weiten Fokus

Die inkrementelle Verarbeitung soll im folgenden konkret an einem Phänomen demonstriert werden, das ohne die Annahme eines Überblicks über alle Äußerungsinkremente zu einem Verarbeitungszeitpunkt zunächst problematisch erscheint. Wir betrachten die Berechnung des weiten Fokus[2], einer Einteilung im Rahmen der Fokus–Hintergrund–Gliederung, die wiederum eine Ebene der Informationsgliederung bildet. Ein Inkrement wird mit dem Merkmal „weit fokussiert“ versehen, wenn mehrere Inkremente einer Äußerung gemeinsam zur Fokussierung vorgesehen werden. Diese Berechnung findet im Semantischen Enkodierer (s. Abb. 8.1), einem an den Verbalisator von Bierwisch & Schreuder (1992) angelehnten Modul, statt und wird erst bei der prosodischen Realisierung der Fokus–Hintergrund–Gliederung während der phonologischen Enkodierung wieder aufgegriffen, wodurch die von der SYNPHONICS–Konzeption postulierte Semantik–Phonologie–Schnittstelle manifestiert wird. Prosodisch wird Fokus im Deutschen durch Akzentuierung markiert, wobei im Falle des engen Fokus genau die fokussierte Konstituente akzentuiert wird. Im Falle von weitem Fokus wird unter der Maßgabe normaler linearer Ordnung der Konstituenten nur ein Fokusakzent realisiert, wobei über Fokusprojektionsregeln innerhalb der Fokusdomäne eben diese akzenttragende Konstituente, der Fokusexponent, bestimmt werden muß (s. Jacobs (1991)). Der Fokusexponent wird in Abhängigkeit von seiner syntaktischen Umgebung bestimmt. Somit wird die Akzentrealisierung von der gegebenen Fokusinformation beschränkt, die auf die Ausdehnung der Fokusdomäne rekurriert.
Während der semantischen Berechnung der Fokusinformation tritt das Problem auf, ohne Vorausschau auf folgende Inkremente erkennen zu können, ob für ein Inkrement, das als zu fokussierend identifiziert wurde, noch mindestens ein weiteres solches unter den übrigen Inkrementen der aktuellen Äußerung auftritt. Eine Lösung bietet die zweite Eingabestruktur neben den zu äußernden Inkrementen in den Semantischen Enkodierer, nämlich die vom Konzeptualisierer aufbereitete Auswahl der relevanten Kontextinformation. Diese Struktur drückt mittels Unterspezifiziertheit an bestimmten Stellen die der Äußerung zugrundeliegende explizite oder implizite Frage aus, basierend auf dem Quaestio–Ansatz von Klein & von Stutterheim (1987), und repräsentiert damit den informationellen Bedarf. Ist die Unterspezifiziertheit derart, daß dieser durch ein einzelnes Inkrement befriedigt werden kann, so wird das entsprechende Inkrement mit dem Merkmal „eng fokussiert“ versehen. Dagegen werden Teile einer Äußerung mit dem Kennzeichen „weit fokussiert„ versehen, wenn der Informationsbedarf nur durch mehrere Inkremente gedeckt werden kann. Genau dies ist aber durch Überprüfung des Ausmaßes der Unterspezifiziertheit der bereits mit dem ersten Inkrement vorliegenden Kontextstruktur erkennbar, so daß zu äußernden Inkrementen, die als zu fokussierend identifiziert wurden, auch direkt die Information beigefügt werden kann, ob sie allein die Fokusdomäne bilden (enger Fokus) oder Bestandteil einer größeren Fokusdomäne sind (weiter Fokus).
Illustriert wird dieses Vorgehen anhand der konzeptuellen Struktur in Abb. 8.2 von

[2] Für eine ausführliche Beschreibung siehe Günther et al. (1994) und Ziesche (1994).

Hans [$_F$schlägt PETER].

$$\begin{bmatrix} \textit{object-refo} & \\ \textsc{Concpred} & \{\textit{hans}\} \\ \textsc{R-Pointer} & \textit{r1} \\ \textsc{Rel-Set} & \left\{ \begin{bmatrix} \textit{obj-sit-rel-elt} & \\ \textsc{Addr} & \textit{s1} \\ \textsc{Rel} & \textit{agent-taker} \end{bmatrix} \right\} \end{bmatrix}$$

$$\begin{bmatrix} \textit{sit-refo} & \\ \textsc{Concpred} & \{\textit{schlagen}\} \\ \textsc{R-Pointer} & \textit{s1} \\ \textsc{Rel-Set} & \left\{ \begin{bmatrix} \textit{sit-obj-rel-elt} & \\ \textsc{Addr} & \textit{r1} \\ \textsc{Rel} & \textit{agent-giver} \end{bmatrix}, \begin{bmatrix} \textit{sit-obj-rel-elt} & \\ \textsc{Addr} & \textit{r2} \\ \textsc{Rel} & \textit{exp-giver} \end{bmatrix}, \right\} \end{bmatrix}$$

$$\begin{bmatrix} \textit{object-refo} & \\ \textsc{Concpred} & \{\textit{peter}\} \\ \textsc{R-Pointer} & \textit{r2} \\ \textsc{Rel-Set} & \left\{ \begin{bmatrix} \textit{obj-sit-rel-elt} & \\ \textsc{Addr} & \textit{s1} \\ \textsc{Rel} & \textit{exp-taker} \end{bmatrix} \right\} \end{bmatrix}$$

Abbildung 8.2: zu äußernde Struktur

Hans schlägt Peter. vor einem Kontext aus Abb. 8.3, der die implizite oder explizite Frage *Was macht Hans?* repräsentiert und somit einen weiten Fokus auf *schlägt Peter* zur Folge hat. Dagegen hätte dieser Satz beispielsweise als Antwort auf die Frage *Was macht Hans mit Peter?* eine enge Fokussierung auf *schlägt*.
Die Äußerung *Hans schlägt Peter.* wird also durch drei die einzelnen Module sukzessive durchlaufenden Inkremente aus Abb. 8.2 repräsentiert, die auf der außersprachlichen Ebene durch die in Abschnitt 8.1 genannten RefOs dargestellt sind und auch in einer anderen Reihenfolge erscheinen könnten. Dabei handelt es sich um ein ObjektrefO mit dem Referenzindex *r1*, das durch die Prädikation (concpred) *hans* gekennzeichnet ist und dessen Relationenmenge (rel–set) angibt, daß es als Agent (agent-taker) in einer

Was macht Hans?

$$\begin{bmatrix} \textit{context} & \\ \textsc{Knowledge} & \left\{ \begin{bmatrix} \textit{object-refo} & \\ \textsc{Concpred} & \{\textit{hans}\} \\ \textsc{R-Pointer} & \textit{r1} \\ \textsc{Rel-Set} & \left\{ \begin{bmatrix} \textit{obj-sit-rel-elt} & \\ \textsc{Addr} & \textit{s1} \\ \textsc{Rel} & \textit{agent-taker} \end{bmatrix} \right\} \end{bmatrix}, \begin{bmatrix} \textit{sit-refo} & \\ \textsc{Concpred} & \{\} \\ \textsc{R-Pointer} & \textit{s1} \\ \textsc{Rel-Set} & \left\{ \begin{bmatrix} \textit{sit-obj-rel-elt} & \\ \textsc{Addr} & \textit{r1} \\ \textsc{Rel} & \textit{agent-giver} \end{bmatrix} \right\} \end{bmatrix} \right\} \end{bmatrix}$$

Abbildung 8.3: Kontext

Situation *s1* fungiert. Ferner liegt ein SituationsrefO *s1* vor, das über die Prädikation *schlagen* verfügt und in Relation zu ObjektrefOs *r1* und *r2* steht, die die Rolle des Agenten (agent–giver) und des Experiencers (exp–giver) einnehmen. Schließlich gibt es das ObjektrefO *r2* mit der Prädikation *peter* und einer Relationenmenge, die ausdrückt, daß es Experiencer (exp–taker) in der Situation *s1* ist.
Der zugehörige Kontext in Abb. 8.3 drückt die implizite oder explizite Frage *Was macht Hans?* mittels zweier RefOs aus: Das den Menschen Hans repräsentierende RefO *r1* ist bereits vollständig spezifiziert und ist demzufolge als Agent in der zu versprachlichenden Situation *s1* angesiedelt. Das die Schlagenssituation repräsentierende RefO *s1* ist unterspezifiziert, und zwar derart, daß die Prädikationenmenge leer ist und die Relationenmenge nur angibt, daß *r1* als Agent auftritt. Dagegen ist das den Menschen Peter repräsentierende RefO *r2* kontextuell nicht spezifiziert.
Wenn nun vor diesem Kontext die Inkremente aus Abb. 8.2 den Semantischen Enkodierer passieren, wird das RefO *r1* mit einer Hintergrund–Markierung versehen, da es in der gleichen relationalen Einbettung schon kontextuell gegeben ist. Dagegen werden die beiden weiteren Inkremente zur Fokussierung vorgesehen, da das RefO *s1* unterspezifiziert und das RefO *r2* gar nicht kontextuell gegeben ist. Zudem kann an der Kontextstruktur aber auch auf folgende Weise pro Inkrement abgelesen werden, daß eine mehr als ein Inkrement umfassende Fokusdomäne vorliegen muß: Die Relationenmenge des zu äußernden SituationsrefOs *s1* ist von den Relationsarten (agent–giver und exp–giver) her eine echte Obermenge der Relationsarten (agent–giver) der Relationenmenge des korrespondierenden kontextuellen SituationsrefOs, d.h. neben diesem *s1* muß mindestens noch das den Experiencer repräsentierende ObjektrefO fokussiert werden. Was das ObjektrefO *r2* anbelangt, ist auch an diesem separat erkennbar, daß es zu einem weiten Fokus gehört, da es nicht im Kontextwissen enthalten ist und zudem das dortige SituationsrefO unterspezifiziert ist, also mindestens diese beiden zur Fokussierung vorgesehen werden müssen. Dieses Gefüge partieller Fokusinformation erlaubt es, den beiden RefOs *s1* und *r2* das Merkmal „weit fokussiert“ zuzuweisen, und zwar auf inkrementelle Weise.
Zusammenfassend läßt sich sagen, daß Informationsstrukturen im Rahmen des Sprachproduktionssystems SYNPHONICS an der Schnittstelle zwischen konzeptueller und sprachlicher Verarbeitung auf der Grundlage eines Vergleichs eines aktuellen konzeptuellen Fragments mit dem jeweils relevanten Kontext der Äußerung durch den Semantischen Enkodierer berechnet und durch prosodische Merkmale realisiert werden. Der Semantische Enkodierer hat dabei zu keinem Zeitpunkt einen Überblick über mehr als das aktuell zu äußernde Inkrement. Derartigen Restriktionen unterliegt die Verarbeitung auf allen Ebenen des SYNPHONICS–Modells, da nur so eine kognitive Adäquatheit hinsichtlich der Geschwindigkeit der Sprachproduktion gewährleistet ist.

8.4 Literatur

Abb, B.; Günther, C.; Herweg, M.; Lebeth, K.; Maienborn, C.; Schopp, A. (1995):] Incremental grammatical encoding — An outline of the SYNPHONICS formulator. In: Extended Proc. of 4th European Language Generation Workshop. Pisa.

Bierwisch, M. & R. Schreuder (1992): From concepts to lexical items. Cognition 42: 23–60.

Günther, C. (ed., 1994): Hamburger Arbeitspapiere zur Sprachproduktion VI. Graduiertenkolleg Kognitionswissenschaft, Universität Hamburg.

Günther, C., Maienborn, C., Schopp, A. (1994): The Processing of Information Structure in SYNPHONICS. In: P. Bosch & R. van der Sandt (eds.), Proceedings of the Interdisciplinary Conference in Celebration of the 10th Anniversary of the Journal of Semantics „Focus & Natural Language Processing", 12.–15. 6. 1994, Schloß Wolfsbrunnen, Vol. 1, 71–81.

Jacobs, J. (1991): Focus Ambiguities. Journal of Semantis 8, 1–36.

Klein, W. & C. von Stutterheim (1987): Quaestio und referentielle Bewegung in Erzählungen. Linguistische Berichte, 109: 163–183.

Levelt, W. J. (1989): Speaking: From Intention to Articulation. Cambridge, Mass.: The MIT Press.

Pollard, C. & I. Sag (1987): Information–based Syntax and Semantics. Vol. I: Fundamentals. Stanford: Center for the Study of Language and Information. Lecture Notes No. 13.

Pollard, C. & I. Sag (1994): Head–driven Phrase Structure Grammar. Chicago: Chicago University Press, CSLI.

Ziesche, S. (1994): Vorgehensweise des semantischen Enkodierers. In: C. Günther (ed.): Hamburger Arbeitspapiere zur Sprachproduktion VI, Graduiertenkolleg Kognitionswissenschaft, Universität Hamburg.

Kapitel 9

Prozeduralität, Lexikon und Systemarchitektur

Gunter Gebhardi
Humboldt-Universität zu Berlin
Philosophische Fakultät II, Computerlinguistik
Unter den Linden 6, 10 099 Berlin
e-mail: gebhardi@compling.hu-berlin.de

Zusammenfassung

Dieser Beitrag gehört zum Tagungsband eines Workshops *„Prozedurale Anforderungen an die maschinelle Sprachverarbeitung“* und fokussiert auf das Lexikon. Daß es mannigfaltige Aspekte zwischen *Prozeduralität und Lexikon* gibt und diese sich nicht im engeren Sinn auf den Lexikonzugriff beziehen, wird im ersten Abschnitt gezeigt. Es werden Architekturaspekte, insbesondere in Bezug auf die Vorschläge in [6], diskutiert. Der Modularisierungsgedanke führt im zweiten Abschnitt zum Ansatz des **virtuellen Lexikons**, der am Beispiel der Flexionsbehandlung erörtert wird. Die Flexion soll hierbei vor allem als Vehikel dienen, das einen einfachen Einstieg in die teilweise sehr maschinennahen Untersuchungen ermöglicht. Im dritten Abschnitt sollen die Ergebnisse in Verbindung zu einigen anderen linguistischen Erscheinungen gebracht werden. In diesem Abschnitt werden auch Fragen der Integration eines solchen Lexikons diskutiert. Abschließend wird ein Ausblick gegeben.
Die Betrachtung steht vor dem Hintergrund der nicht–numerischen Computerlinguistik und computerlinguistischer Techniken in der Tradition der in den letzten Jahren entwickelten und untersuchten unifikationsbasierten Systeme. Dennoch sollten die Ergebnisse nicht uninteressant sein auch aus der Sicht der (wieder modernen) numerischen Sprachverarbeitungssysteme, obwohl die Strukturen im Lexikon dort bei weitem nicht dieselbe Komplexität haben.

[1]Das diesem Beitrag zugrundeliegende Vorhaben wurde mit Mitteln des Bundesministeriums für Forschung und Technologie unter dem Förderkennzeichen 01 IV 101 G gefördert. Die Verantwortung für den Inhalt dieses Beitrags liegt beim Autor.

9.1 Ausgangspunkt

9.1.1 Technische Aspekte des Lexikons

Ausgangspunkt sind Systeme zur Verarbeitung natürlicher Sprache, gleich ob zur Analyse oder zur Generierung. Die Funktionsweise dieser Systeme wird durch eine Menge von Regeln und Fakten im weitesten Sinne beschrieben, die den Verlauf der Verarbeitung bestimmen.
So unterschiedlich auch die Systeme und ihre Verarbeitungsstrategien sind, ihn ist allen die Annahme eines Lexikons gemein. Die Lexikoneinträge haben dabei die Funktion von Axiomen. Das bedeutet, daß das, was dem Lexikon entnommen wird, als Fakt zu werten ist und damit keiner weiteren Fundierung bedarf. Die Aufgabe der Auswahl der entsprechenden Fakten, obwohl auch ein durch Regeln zu beschreibender Vorgang, soll hier nicht weiter betrachtet werden.
Nun lassen sich theoretisch zwei Extreme zeichnen:

- Jede Äußerung, die vom System zu verarbeiten ist, wird im Lexikon abgelegt. Damit erreicht der Aufwand zur Verarbeitung der Informationen aus dem Lexikon ein Minimum, der Umfang der Informationen im Lexikon ein Maximum.

- Jede Äußerung wird soweit wie möglich zerlegt. Der Aufwand zur Verarbeitung der Informationen erreicht damit ein Maximum, der Umfang der Informationen im Lexikon ein Minimum

Beide Extreme besitzen praktisch kaum Relevanz, sieht man davon ab, daß ein System entsprechend der ersten Variante für bestimmte, sehr eingeschränkte Aufgaben nutzbar ist.[1]
Die beiden Extreme lassen sich technisch wie folgt charakterisieren:

- minimaler Berechnungsaufwand, daher sehr schnell, aber: maximaler Speicheraufwand

- maximaler Berechnungsaufwand, folglich langsamer, aber: minimaler Speicheraufwand

Der gegenwärtige Stand der Technik läßt noch keine Optimierung der Verarbeitungsverfahren entsprechend der einen oder anderen Art aufgrund bestimmter erwarteter Kostenvorteile zu,[2] vielmehr geht es um die Machbarkeit der Systeme für komplexe Anwendungen überhaupt.
Daher resultiert der Wunsch, einerseits zur effizienten Lösung komplexer Aufgaben ein umfangreiches und weitgehend vorverarbeitete Informationen umfassendes Lexikon zu nutzen und andererseits ein kleines, weil dadurch kostengünstig abzuspeicherndes.
Vor 20 Jahren bestand das Problem, ein größeres Lexikon in den damals verfügbaren Speichern überhaupt ablegen zu können. In dieser Art existiert das Problem heute nicht mehr. Bei der gegenwärtig erreichten Beschreibungskomplexität, die durchaus als Erfolg gewertet werden kann, erfordert die Verarbeitung nicht zu vernachlässigenden Zeitaufwand. Um diesen zu minimieren werden möglichst viele Berechnungen vor dem Systemlauf ausgeführt und deren Ergebnisse abgelegt – zum großen Teil auch im

[1] Bestimmte Einzelworterkenner kann man den Systemen zuordnen, bei denen jede vom System zu verarbeitende Äußerung im Lexikon abgelegt ist: Eine Wortform wird als vollständige Äußerung betrachtet und sämtliche Informationen bezüglich der Wortform sind im Lexikon abgelegt.

[2] Auch hier soll von eingeschränkten speziellen Systemen abgesehen werden.

Lexikon. Dies führt zum Anwachsen des Speicherumfangs des Lexikons und damit zu unwirtschaftlichen Speichergrößen.
Damit sind wieder Untersuchungen notwendig, die Lexikongröße zu minimieren. Der auch hier verfolgte Ansatz besteht darin, das Lexikon zu komprimieren. Die Frage ist, wie dies geschehen soll.
Eine naheliegende, aber nicht befriedigende Lösung ist, die Größe des Lexikons mit einem der bekannten universellen Kompressionsalgorithmen einzuschränken. Als erstes Problem ergibt sich, daß die meisten der bekannten Kompressionsalgorithmen genau dann besonders effizient sind, wenn möglichst große Datenmengen bearbeitet werden. Dies ist aber hier nicht der Fall, da nicht das Lexikon *insgesamt*, sondern *jeder* Eintrag *für sich* komprimiert werden soll. Das nächste Problem tritt beim Zugriff auf, wenn der verwendete Zugriffsschlüssel nicht ausschließlich die geforderten Lexikoneinträge selegiert, mit anderen Worten: wenn zu viele Lexikoneinträge gefunden werden. In einer solchen Situation müssen zunächst sämtliche potentiellen Lexikoneinträge expandiert und danach bezüglich der geforderten Informationen bewertet werden. Dabei werden Lexikoneinträge expandiert, deren Einbeziehung in die Verarbeitung anschließend wieder verworfen wird. Eine solche Situation tritt auf, wenn die Anzahl möglicher Schlüssel nicht stark eingeschränkt werden kann und deshalb allgemeinere Schlüssel verwendet werden, die bezüglich des eigentlich benötigten Schlüssels überselegieren. Eine Einschränkung der Schlüssel sollte aber a priori nicht geschehen. Ein weiteres Problem bei der Verwendung universeller Kompressionsalgorithmen besteht darin, daß damit zusätzlicher Berechnungsaufwand in das System gebracht wird.
Um die Probleme bei der Nutzung universeller Kompressionsalgorithmen zu vermeiden, muß eine aus Sicht des Verarbeitungssystems transparente Kompression jedes einzelnen Eintrags mit möglichst geringem zusätzlichen Berechnungsaufwand erfolgen. Dies kann dadurch erreicht werden, daß die Kompression mit anderen notwendigen Systemoperationen verknüpft wird. Die im weiteren verfolgte Idee ist daher ein *Modul Lexikon*, welches nach außen weitgehend vorverarbeitete Informationen liefert, intern äußerst kompakt ist, zur Abbildung von der internen auf die äußere Darstellung möglichst wenig zusätzlichen Berechnungsaufwand erfordert und daher bestimmte systemnotwendige Operationen einbezieht und zur Kompression ausnutzt.

9.1.2 Modularisierung

Nochmals Jahre zurück: Markante Systeme Ende der siebziger Jahre auf dem Gebiet der automatischen Sprachverarbeitung waren Systeme basierend auf *Augmented Transition Networks* (ATN) [1][3]. Eine der Erkenntnisse aus der Arbeit mit diesen Systemen war, daß die Komplexität der Beschreibung mit Hilfe dieser Systeme ein Maß erreichte, das nur noch schwer beherrschbar war. Als ein Ausweg schienen die über einem einheitlichen Datentyp – Merkmalstrukturen – operierenden unifikationsbasierten Formalismen [5] geeignet, da sie eine Beschreibungsmächtigkeit besitzen, die es erlaubte, komplexe Sachverhalte elegant[4] zu beschreiben. Die Einführung von Typen brachte eine zusätzliche Ausdrucksstärke (*Turing*–mächtig).
[6] faßt nach fast zehnjähriger Entwicklung die Erfahrungen mit den unifikationsbasierten Systemen dann aber wie folgt zusammen (p. 86): *The use of one single data structure did not result in a homogenous, elegant formalism, but only in unnecessarily complex and unefficient procedures in some parts of the system. ... The one and only,*

[3] [1] ist ein Sammelband mit Aufsätzen, der die Ergebnisse der dabei verfolgten unterschiedlichen Ansätze darstellt.
[4] Was ist eigentlich Eleganz?

universal, elegant and efficient formalism suitable for all problems in NLP does not exist. So we'd better stop searching for it. (NLP Natural Language Processing).
So wird in [6] die Schlußfolgerung gezogen: *.., I want to argue for a variety of task specific formalisms that are well–integrated.* Besser ist das nicht zu formulieren.
Danach werden eine Reihe von Ansätzen diskutiert, in welcher Richtung ein solcher Formalismus zu verändern ist. Zwei Richtungen sollen hier hervorgehoben werden:

- die Anreicherung um weitere Beschreibungsmittel für jeweils spezielle Aufgaben
- Interfaces zu externen Systemen und Modulen, welche sich in ihrer jeweiligen Funktionalität in das entworfene Konzept des *constraint solving* einbinden lassen

Beide Ansätze für Veränderungen scheinen sinnvoll. Widersprechen sich die Ansätze aber nicht? Einerseits sollen Beschreibungsmittel hinzugefügt werden, andererseits wird doch aber gerade damit die Notwendigkeit für externe Systeme und Module geschwächt; die Ansätze widersprechen sich.
Nimmt bei Erweiterung eines solchen Formalismus die Komplexität ab und damit die Effizienz zu, wie zuerst gefordert? Nun zielen weitere Ideen in [6] direkt auf eine Verbesserung der Effizienz hin. Müssen diese Ansätze nicht aber zunächst den zu erwartenden Effizienzverlust durch die Erweiterungen kompensieren? Und was bleibt dann an Effizienzgewinn? Die Eingangsfrage – Komplexitäts- *und* Effizienzzunahme – ist sicherlich aus Sicht eines Formalismus mit Nein zu beantworten, aus Sicht eines Grammatikautors könnte dies anders sein. Die Anwort auf die letzte Frage ist offen.
Beide Ansätze sollen hier vereint und noch radikaler gestellt werden — damit ist man genau wieder an der Ausgangsposition für solche Veränderungen, wie sie oben von [6] erhoben wurden: Für jeden Teilbereich sollen geeignete effiziente Lösungen enwickelt werden mit adäquaten Beschreibungsmitteln. Diese sind in ein gemeinsames System zu integrieren.
Die Konzeption von [6] sieht im Mittelpunkt einen *Constraint Solver*. Die Menge der Constraints, die zu einem bestimmten Zeitpunkt zu lösen sind, wird durch die Beschreibungszustände von Teilmodulen, ganz gleich ob technisch in der Sprache des Constraint Solvers beschrieben oder aber als echte Teilmodule, realisiert. Die Aufgabe des Solvers ist es, genau die Teilmodule zu bestimmen und zu aktivieren, die durch Übergang in einen Folgezustand (sprich: Weiterrechnen) der Lösung der Gesamtaufgabe am besten dienen. Der Constraint Solver ist Scheduler des Systems.
Dem soll keine andere Architektur entgegengestellt werden, vielmehr soll diese Architektur konsequent zu Ende geführt werden: Der Scheduler des Systems ist ein eigenständiges Modul, welches für diese Aufgabe spezialisiert ist; das Modul für die Verarbeitung der Grammatik wird genau für diese Aufgabe optimiert, ein Kommunikationsmodul dient der Interaktion zwischen allen Modulen, weitere Module lösen andere Aufgabe und sind dafür optimiert. Genau auf diese Art und Weise lassen sich effiziente Formalismen für die jeweilige Teilaufgabe realisieren und ein Overhead, der sich in Ineffizienz niederschlägt, wird vermieden.
Bei einer solchen Architektur ist die Abstimmung der Module und deren Verbindung zum Datenaustausch ein zentrales Problem. [6] kommt daher in seinen Überlegungen zu den Aspekten einer Modularisierung zu der Vermutung: *„And I expect, that sometimes the design and implementation of the interface will be more complicated than the tiny little formalism for a simple task.“*.
Einige Fragen, die dagegengestellt werden sollen und diese Vermutung als zu einfachen Schluß zeigen, sind:

- Welche Module gibt es? Welche Aufgaben müssen in einem Modul gelöst werden und welche Aufgaben lassen sich auf Module verteilen?
- Wie sind die Kommunikationsanforderungen dieser Module?
- Sind alle softwaretechnologischen Möglichkeiten – etwa der objektorientierten Analyse und Synthese – auf Anwendbarkeit geprüft und dann verworfen worden?

Weitere Fragen lassen sich anschließen. Statt einen Formalismus noch mächtiger zu machen, sollten die für eine bestimmte Aufgabe angemessenen Formalismen optimiert und die Möglichkeit ihrer Integration untersucht werden. Spezielle Formalismen oder deren Konzepte sollten nicht *in einen* universellen Formalismus aufgenommen, sondern *miteinander* verbunden werden.

9.2 Konsequenzen für das Lexikon – das virtuelle Lexikon

In Abschnitt 9.1.1 wurde das technische Dilemma bei der Gestaltung des Lexikons aufgezeigt. Im vorigen Abschnitt wurden Ansätze für eine Modularisierung diskutiert. Welche Möglichkeiten bietet ein *Modul Lexikon*?
Der hier vorgestellte Ansatz hat einen zentralen Gedanken: **das virtuelle Lexikon**. Das eigentliche Lexikon – es soll zur Unterscheidung zum virtuellen Lexikon als reelles bezeichnet werden – wird in eine sehr kompakte Darstellung gebracht, um der Anforderung nach geringem Speicherumfang (es geht hier um Größen von mehreren Megabyte) gerecht zu werden. Um dieses Lexikon wird – quasi wie eine Schale – ein Verarbeitungssystem gelegt, welches die Lexikoneinträge effizient expandiert, so daß das Lexikon aus Sicht der Grammatik virtuell aus sehr vielen und vollständigen Lexikoneinträgen zu bestehen scheint. Damit kann der Grammatik Berechnungsaufwand genommen werden, der im Modul Lexikon (möglichst reduziert) wieder in Erscheinung tritt, dort aber zusätzlich weitere Aufgaben erfüllt.
Die Frage, der im weiteren nachgegangen werden soll, ist, welche Möglichkeiten existieren, Verarbeitungsschritte innerhalb einer Grammatik so zu nutzen, daß sie gleichzeitig Funktionen in der Verarbeitungshülle des Lexikons tragen können. Dies wird zunächst am Beispiel von Flexionsinformationen diskutiert.

9.2.1 Speicherplatz *versus* Verarbeitungszeit

Wortformen

Zunächst sollen die Ergebnisse gezeigt werden, die bei der Kodierung von Wortformen erreichbar sind. Dabei wurden bewußt möglichst einfache Lösungen gewählt, die in ihren Eigenschaften bekannt sind. Um einen Bezug zur unmittelbaren Anwendung zu bekommen und insbesondere auch um das Laufzeitverhalten abschätzen zu können, wurden experimentell maschinennahe Programmfragmente erstellt[5]. Pro Befehl wird

[5]Benutzt wurde eine Assemblersprache, angelehnt an die der Prozessoren in [2] und [3], die weitgehend kompatibel zu Typen der Firmen Zilog und Intel sind. Auf diese Weise lassen sich unmittelbar Taktzyklen auszählen. Da die Prozessoren eine typische CISC–Architektur haben und damit nicht den gegenwärtig zunehmend dominierenden Prozessoren mit RISC–Architektur entsprechen, wurde auf die Verwendung komplexer Mehrzyklen/–bytebefehle verzichtet

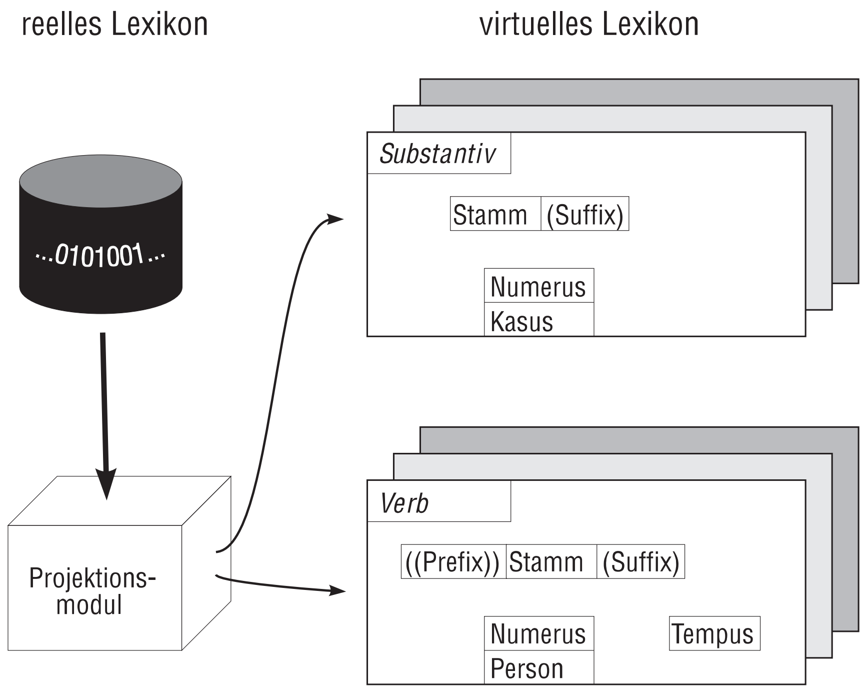


Abbildung 9.1: Prinzipielle Vorgehensweise bei der Abbildung der Flexionsinformationen aus dem reellen Lexikon in das virtuelle

eine Länge von einem Taktzyklus T angesetzt[6]. Angenommen wurde eine Länge der jeweiligen Wortform von 10 Zeichen.
Folgende Werte wurden ermittelt:

Variante	Speicherbedarf	Kodierung	Zugriffszeit
1	10 Byte	Byte Kodierung	< 20 T
2	6 ... 7 Byte	5–bit Kodierung	< 35 T
3	um 5 Byte	*Huffman*–Kodierung	< 120 T
4	um 2 Byte[7]	ähnlich einem Automaten	< 200 T

Für die Varianten 1 – 3 muß zusätzlich noch ein Aufwand zur Bestimmung des Schlüssels eingerechnet werden und für alle Varianten gegebenenfalls noch eine Referenz auf den eigentlichen Lexikoneintrag.

Flexionsinformationen

Entsprechend dem Konzept des Moduls Lexikon wurde untersucht, welchen Speicheraufwand die Unterbringung von Flexionsinformationen im reellen Lexikon erfordert und welcher Berechnungsaufwand zur Abbildung dieser Informationen auf die Strukturen des virtuellen Lexikons nötig ist. In Abbildung 9.1 wird das prinzipielle Vorgehen

[6] Dies entspricht nicht den echten Taktzyklen.
[7] bei hinreichender Größe des Gesamtlexikons

skizziert. Die Flexionsinformationen sind in gepackter Form im Lexikon verankert und werden durch ein als Projektionsmodul bezeichnetes Programm auf das virtuelle Lexikon abgebildet. Als Flexionsinformationen wurden die Informationen zur Beschreibung der Veränderung der Wortform (mit Ausnahme von Stammveränderungen über die Umlautung hinausgehend) berücksichtigt, sowie weiter bei Substantiven Numerus und Kasus und bei Verben Numerus, Person und Tempus. Adjektive wurden nicht einbezogen; der zu erwartende notwendige Aufwand zu ihrer Behandlung liegt zwischen dem der Substantive und Verben.
Zur Abspeicherung dieser Informationen zur Flexion sind zu den oben ermittelten Kosten folgende zu addieren (für ein großes Gesamtlexikon (> 40 000 Einträge) überschlagen):

Variante	Speicherbedarf	Zugriffszeit
1	1 Byte	< 30 T
3	um 4 Bit	< 100 T
4	um 2 Bit	< 100 T

Die Angaben zum Speicherplatz berücksichtigen nur den für die Informationen im Lexikon erforderlichen zusätzlichen Speicherplatz, jedoch nicht den vom Dekompressionsprogramm und dessen Daten benötigten. Bezogen auf die angenommene Lexikongröße liegt dieser Zusatzbedarf jedoch unter einem Bit je Eintrag.
In Abhängigkeit vom gewählten Zugriff auf die Informationen kann gegebenenfalls noch etwas zusätzlicher Speicheraufwand erforderlich sein. Bei der überschlägigen Berechnung wurde von einem modifizierten Schlüssel über die verkürzte Wortformlänge ausgegangen, der zu einem häufigeren *Re–Hash* beim Zugriff führt.
Ein interessanter Seiteneffekt ergibt sich bei dem untersuchten Verfahren: Unbekannte Wortformen können als hypothetische Lexikoneinträge angenommen werden. Bei deren Projektion auf das virtuelle Lexikon werden sie mit den im Dekompressionsprogramm verankerten Flexionsinformationen verbunden. In der Sicht auf das virtuelle Lexikon stehen diese Einträge – stellvertretend für die unbekannten Wortformen – mit den möglichen, für die weitere Verarbeitung notwendigen Flexionsinformationen zur Verfügung.

9.2.2 Diskussion

Bei Einbeziehung von Flexionsinformation in das Lexikon benötigt dieses bei hinreichender Größe nur geringen zusätzlichen Speicheraufwand. Ein bezüglich der Flexion als Vollformenlexikon zu betrachtendes Lexikon wird in seinem Umfang also keinesfalls mehrfach so groß wie ein Lexikon, in welches jede Wortform lediglich einmal aufgenommen wurde. Diese Beobachtung ist sehr wichtig, denn eine intensive Verfolgung dieses Gedankens der Ablage von Vollformen im Lexikon kann zu Verfahren der Abspeicherung weiterer Informationen im Lexikon führen – von Informationen, die anderenfalls erst durch das Verarbeitungsverfahren zu berechnen sind. Damit gewinnt das gesamte Verarbeitungssystem an Effizienz.
Die angegebenen Taktzeiten dienen lediglich dem Nachweis der jeweiligen Formen im Lexikon. Ein echtes Kopieren aus dem Lexikon verursacht zusätzlichen Aufwand, wobei dieser bei allen Verfahren ähnlich ist. Wenn für T ein Wert von einigen Nanosekunden angenommen wird,[8] beansprucht der Zugriff auf das Lexikon mit dem Kopieren der Informationen aus diesem nicht zu vernachlässigende Zeit.

[8] Prozessoren jüngeren Entwicklungsdatums

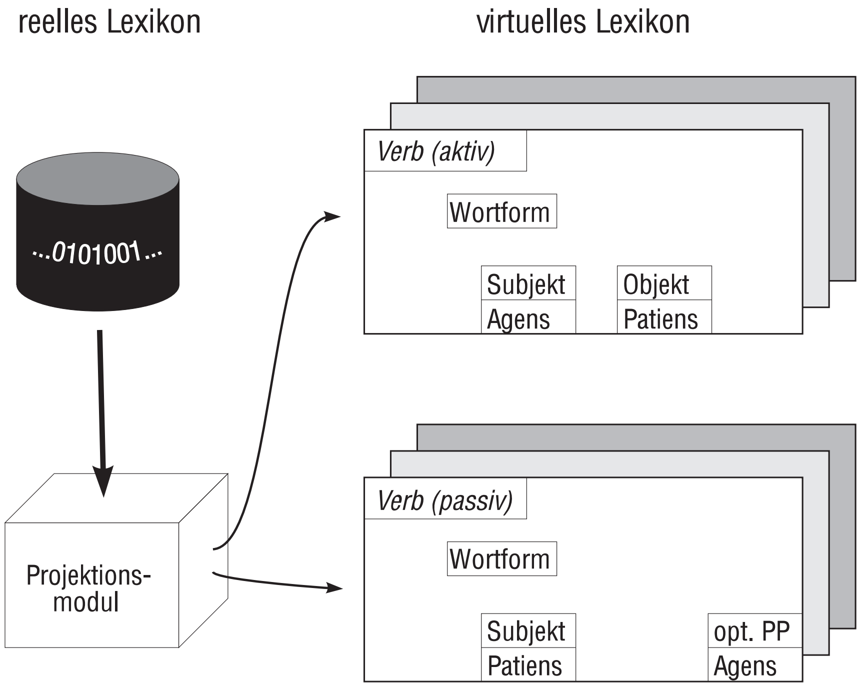


Abbildung 9.2: Prinzipielle Vorgehensweise bei der Abbildung der Informationen für den Gebrauch von Verben im Aktiv bzw. Passiv aus dem reellen Lexikon in das virtuelle

9.3 Weitere Beispiele und Integration

9.3.1 Weitere Beispiele

Mit der Einbeziehung von Flexionsinformationen bezieht ein Modul Lexikon regelhaft zu beschreibendes Wissen ein. Wie sich auch komplexere Regeln aus der Grammatik in das Lexikon einbeziehen lassen, wird im folgenden diskutiert.
Als Beispiel soll die Bildung von Passivformen dienen. Es wird angenommen, daß eine Grammatik Regeln enthält, die die Abbildung der Beschreibung des Gebrauchs eines Verbs in Aktivform auf den Gebrauch in Passivform beschreibt. Unter Einbeziehung des Zugriffs auf das Lexikon ist die Bestimmung der Passivform ein zweistufiger Vorgang: Überführung der Informationen zur Aktivform aus dem Lexikon und Bildung der Passivform.
Dem soll eine einstufige Verarbeitung gegenübergestellt werden. Dabei werden die Informationen zum Verb im reellen Lexikon so auf das virtuelle Lexikon abgebildet, daß sowohl die Aktiv- als auch die Passivform im Lexikon abgelegt scheinen. Dies wird in Abbildung 9.2 vereinfacht für ein transitives Verb umrissen.
Bei dieser Art der Verarbeitung wird die Berechnung zur Bestimmung der Passivform ausgenutzt, um die Informationen im Lexikon kompakter ablegen zu können oder, aus entgegengesetzter Sicht, mit der Expansion der Daten im Lexikonmodul wird gleichzeitig ein Berechnungsschritt der Grammatik ausgeführt. Wie auch immer, es wird der Berechungsaufwand im Gesamtsystem verringert.
Weitere Beispiele ließen sich anschließen. So können Formen der Derivation über die morphologischen Erscheinungen hinausgehend im Lexikonmodul effizient behandelt werden. Auch bieten sich bestimmte Variationen im Kasusrahmen von Verben für eine solche Behandlung an.
Was sich auf diesem Weg natürlich nicht ergibt, sind Lösungen für Probleme, die sich auch innerhalb der Grammatik nicht behandeln lassen.

9.3.2 Integration

Die Nutzung eines solchen virtuellen Lexikons für ein größeres System wirft einige Probleme auf. In der Diskussion in Abschnitt 9.1.2 wird das Hauptproblem bereits angesprochen: die Integration unterschiedlicher Module. Als ein solches Modul muß das Lexikon angesehen werden. Dafür gibt es jedoch technische Lösungen.
Weitaus schwieriger ist die Verbindung zur Grammatik herzustellen. Die Aufgaben, die dabei gelöst werden müssen, beginnen mit der Selektion und Extraktion geeigneter Regeln aus der Grammatik und führen zur Compilation der entsprechenden Regeln als Regeln im Lexikon. Da eine Grammatik und das Lexikon Änderungen unterliegen, ist dieser Prozeß unbedingt zu automatisieren.
Hier liegt jedoch ein Hauptproblem bei der Umsetzung des diskutierten Konzepts. Auf der einen Seite führten die Entwicklungen der letzten Jahre auf linguistischer Ebene zu immer integrierteren und umfassenderen Beschreibungen, und maßgebliches Ziel der Entwicklung von Verarbeitungsformalismen war es, als Vehikel für diese Beschreibungen zu dienen. Was hier nun auf der anderen Seite gefordert wird, ist die Zerlegung der monolithischen Verarbeitungsformalismen in effizient interagierende Module. Dafür fehlt – zuweilen neben der Einsicht – die notwendige Technik.

9.4 Ausblick

Die Einbeziehung von Regeln in das Lexikon oder in die Grammatik, die Bestimmung der Regeln und die Frage des richtigen Zeitpunkts der Auswertung der Regeln vor oder während der Systemlaufzeit muß für jedes sprachverarbeitende System diskutiert werden. Was hier in Erinnerung gerufen werden sollte, ist, daß ein solches System im betrachteten Bereich Morphologie/Syntax/Semantik nicht nur aus einem Datenspeicher Lexikon und einer Datenverarbeitungsmaschine in Form eines einheitlichen Formalismus besteht[9], sondern daß es aus Gründen der Effizienz sinnvoll ist, einzelne Module mit jeweils spezieller Funktionalität einzusetzen.
Die Komplexität und der damit verbundene Aufwand bei unifikationsbasierten Systemen haben ein Maß angenommen, daß die praktische technische Nutzbarkeit eines solchen Systems als alleiniges universelles System in Frage stellt. Ungeachtet dessen haben die Grammatiken auf diesen Systemen einen linguistisch höchst anspruchsvollen Grad erreicht. Es sollte daher verstärkt untersucht werden, wie solche Systeme *effizient* nutzbar zu machen sind.
Ob der Ansatz zur Realisierung eines virtuellen Lexikons als eigenständiges Modul in einem System unmittelbar umzusetzen ist oder ob bestimmte Ideen bei der Compilation einer Grammatik und des damit verbundenen Lexikons durch einen Formalismus ausgenutzt werden, bleibt zu untersuchen; kontraproduktiv wäre es aber, wenn dadurch Beschreibungen in Form der Grammatiken und des Lexikons verändert werden müssen.

[9] Eine Art Feigenblatt ist oft ein auf der two–level–morphology basierendes Teilmodul [4].

Literaturverzeichnis

[1] Bolc, L.:*Augmented Transition Networks.* Springer Verlag. Berlin. 1983.

[2] Bonitz, J.:*Der 16–Bit–Mikroprozessor des ESER–PC.* VEB Verlag Technik Berlin. 1989.

[3] Kieser, H. und Meder, M.: *Mikroprozessortechnik*. VEB Verlag Technik Berlin. 1982.

[4] Koskenniemi, K.: *Two–Level Morphology: A General Computational Model for Word–Form Recognition and Production.* University of Helsinki. 1983.

[5] Shieber, S. M.: *An Introduction to Unification–Based Approaches to Grammar.* CSLI Lecture Notes No. 4. Stanford. 1986.

[6] Seiffert, R.: *How could a good system for practical NLP look like?*. in: Trost, H. and Backofen, R.: *Coping with Linguistic Ambiguity in Typed Feature Formalisms.* Proceedings of a Workshop held at ECAI 92. Vienna. 1992. also: IWBS Report 228. IBM Wissenschaftliches Zentrum. Institut für Wissensbasierte Systeme.

Kapitel 10

Preferential Disambiguation and Object-Oriented Representations

J. Joachim Quantz
Technische Universität Berlin, Projekt KIT-VM11, FR 5-12,
Franklinstr. 27/28, D-10587 Berlin
jjq@cs.tu-berlin.de

10.1 Introduction

Ambiguity is an essential feature of natural language utterances: though each utterance is normally quite unambiguous in its utterance situation, abstraction from this situation makes the utterance and its constituents highly ambiguous. In Natural Language Processing (NLP) only a limited amount of the utterance situation is explicitly available, and on the basis of this limited information adequate disambiguation is beyond the scope of most existing systems. Instead we find two different "strategies" towards the phenomenon of ambiguity:

1. some systems are restricted to more or less unambiguous input;
2. some systems or components construct multiple structures or structures that are themselves ambiguous.

Obviously, both strategies are inadequate in the context of application-oriented NLP. The strategy of *preferential disambiguation*, on the other hand, explicitly addresses the problem of ambiguity. It has been applied, for example, for anaphora resolution in the FAST project [24], and has been formalized and generalized in [25, 29]. This formalization has been performed in the context of Description Logics (DL) and is based on weighted defaults and exception minimization. In this paper I briefly describe both the general characteristics of preferential disambiguation (Section 10.2) and the technical realization in terms of weighted defaults and exception minimization (Section 10.3).

[1]This work was funded by the German Federal Ministry for Research and Technology (BMFT) in the framework of the Verbmobil Project under Grant 01 IV 101Q8. The responsability for the contents of this study lies with the author.

Whereas the formalization of preferential disambiguation in [25] is based on a homogeneous representation in terms of DL, preferential disambiguation can also be achieved in the context of heterogeneous architectures. This requires, however, that the heterogeneous representations are treated as abstract data types. Given this perspective, the DL representations used in [26, 27] can be seen as an object-oriented realization of abstract data types in NLP. In Section 10.4 I discuss the advantages of abstract data types and sketch the difference between object-oriented DL representations and representations based on typed feature structures.

10.2 Preferential Disambiguation

The basic idea of *preferential disambiguation* is that different ambiguity phenomena share certain characteristics and therefore can all be handled with the same disambiguation strategy. The main hypotheses underlying this disambiguation strategy are the following:

1. The contextual information relevant for disambiguation is of quite *heterogeneous* nature and comprises syntactic, semantic, conceptual, encyclopedic, and pragmatic information.
2. The contextual information to be used for disambiguation does not impose strict constraints which would rule out alternative interpretations completely, but rather induces *preferences* for particular interpretations.
3. The preferences induced by different pieces of contextual information vary wrt their respective *relevance*.
4. When disambiguating an expression one has to combine all the preferences stemming from the context in order to determine an overall global preference. To do so a *homogeneous representation* of contextual information and the corresponding preferences is advantageous.

In the next section I will present a formalization of preferential disambiguation in terms of weighted defaults. It should be noted, however, that the idea of preferential disambiguation is independent of this particular formalization. Preferential disambiguation in the broader sense has been proposed, for example, to deal with

- anaphora [18, 24],
- definite descriptions and modifier attachment [23],
- noun phrase countability and number [3],
- modal verbs [14],
- PP-attachment [15],
- parsing of spoken language [20],
- determination of speech-event types [34, 35].

In all these cases heterogeneous information with different degree of relevance for the resolution is taken into account. Note that several formal frameworks have been proposed which aim at integrating heterogeneous resolution strategies, e.g. *contextual constraints* [10], *interpretation as abduction* [13], or *interpretation as exception minimization* [25]. Whereas the first approach is based on strict constraints, the later two are based on weighted constraints and thereby allow to represent the respective relevance of the constraints.

10.3 Exception Minimization

The strategy of exception minimization has been developed in the context of Description Logics (DL), which have become one of the major paradigms in Knowledge Representation. Combining ideas from Semantic Networks [32] and Frames [22] with the formal rigor of First Order Logic, research in DL has focussed on theoretical foundations [8] as well as on system development [5] and application in real-world scenarios [29].

In this section I sketch the basic ideas of weighted defaults and exception minimization. A detailed description of a modeltheoretic semantics and a proof theory of weighted defaults in the context of DL is given in [31]. Basically, the alphabet of a Description Logic contains *concepts* (unary predicates), *Roles* (binary predicates), and *objects* (individual constants). DL dialects vary wrt the term-forming operators they support, e.g. conjunction, disjunction, and negation for concepts and roles; value restrictions and number restrictions for concepts; composition and inversions of roles.

For the purpose of this paper it is sufficient to consider a formal logic $\mathcal{L}$ containing unary predicates (concepts) $P_1, ..., P_m$ and constants (objects) $o_1, ..., o_n$. The formulae expressible in this language then have the form $P_i(o_j)$. A standard modeltheoretic semantics maps predicates into subsets of a domain D and constants into elements of D. A model M is a model of a formula $P_i(o_j)$ iff $[\![o_j]\!] \in [\![P_i]\!]$. Strict entailment of a formula γ from a set of formula Γ is defined as usual, i.e. $\Gamma \models \gamma$ iff all models of Γ are also models of γ.

Given such a strict logic $\mathcal{L}$, a nonmonotonic extension in terms of weighted defaults can be straightforwardly defined by using the framework of preferential model theory [16, 19]. In this framework, a preferential nonmonotonic entailment relation $\mid\!\approx$ is defined by taking into account only maximal models. Thus given a preferential ordering $\sqsubset$ on the set of models, $\Gamma \mid\!\approx \gamma$ iff all models of Γ which are maximal wrt $\sqsubset$ are models of γ. Given this framework one only has to define a suitable ordering $\sqsubset$ to obtain preferential entailment.

Defaults can best be characterized as rules which allow for exceptions. The most natural way of defining an ordering on models is therefore to prefer models which minimize exceptions. In the case of weighted defaults, which have the form $P_i \leadsto_w P_j$, this minimization should take into account the weight w of the default (the higher the weight w, the more relevant the default). Obviously, an object o is an exception to a default $P_i \leadsto_w P_j$ in a model iff $[\![o]\!] \in [\![P_i]\!] \wedge [\![o]\!] \notin [\![P_j]\!]$.

We thus obtain for each model a set of exceptions, where each exception is a pair of an object and a default. Given the weights of the defaults we can straightforwardly compute a negative score for each model by adding up the weights of the defaults occurring in the exception set. Note that the weight of a default is added for each object which is an exception to it. The preference ordering on models is then directly obtained from the negative scores of the models, i.e. $M_1 \sqsubset M_2$ iff $\text{score}^-(M_1) >$

$\text{score}^-(M_2)$.
Note that the numerical priorization achieved by weighted defaults differs considerably from the standard approach to priorization in Nonmononotic Logics. Given a lexicographic priorization, as in [2, 6, 17, 28, 33], a stronger default can cancel an arbitrary number of weaker defaults. Weighted defaults, however, allow the accumulation of weak defaults to overwrite a stronger default. Thus instead of having only priorities between individual defaults, weighted defaults express priorities between *multisets* of defaults.

10.4 Object-Oriented Representations

In the characterization of preferential disambiguation in Section 10.2 I have stressed that a homogeneous representation of contextual information and the corresponding preferences is advantageous. This can be achieved, for example, by implementing a syntactic parser based on DL [26]. The main idea of such a parser is to represent the constituents of an expression, i.e. phrases and words, as DL objects.
Though a homogeneous representation is to be preferred from a theoretical point of view, a heterogeneous architecture in which special-purpose formalisms are used to represent different information levels does in general yield better results wrt efficient performance. In the following I will argue that preferential disambiguation can also be applied to heterogeneous representations, provided these representations can be treated as *abstract data types*. From this perspective, the DL representation of syntactic structure, conceptual content, and reference proposed in [26, 27] should be seen as object-oriented realizations of such abstract data types.
In order to explain the notion of abstract data types it is useful to first consider the standard design of NLP systems. Many NLP systems contain different components as syntactic analysis, semantic interpretation, etc. which are combined in a sequential architecture, e.g. [1]. A popular way of realizing the interface between the different components is to pass a data structure, e.g. a phrase structure tree containing the syntactic analysis of a sentence. The subsequent component *parses* this data structure and in doing so constructs its own representation. In the following I argue that this approach reflects some basic assumptions underlying linguistic theories, but is inadequate from a software-engineering point of view.
Parsers based on syntactic theories in Computational Linguistics [36], such as GB, LFG or HPSG, take a sequence of words as input an construct one or more phrase structure trees, which represent the alternative syntactic structures of the sentence. Similarly, semantic interpretation takes a sequence of words or a syntactic structure as input and produces one or more semantic representations (usually formulae in some logical language representing the truth conditions of the sentence). More often than not, syntactic and semantic interpretation is closely related, either due to integrated theories (GB, LFG, HPSG), or due to a correspondence between syntactic and semantic construction rules (Montague Grammar [9]).
Given such a sequential architecture, syntax and semantics tend to focus on an internal perspective, i.e. they are mainly concerned with problems arising in the construction of their representations (e.g. ambiguity etc.). I regard this as a rather necessary endeavor, far from being irrelevant. I propose, however, to augment this internal perspective by an *external* one. Syntax and semantics have to provide other components with the relevant information required. It should be obvious that these components will have an external perspective on syntax and semantics, i.e. they are not necessarily interested

in the particular internal structures constructed in syntax and semantics.
Note that the focus on an internal perspective is rather natural if an implementation is intended to *test* a particular linguistic theory. As soon as a system is intended to be used in a particular real-world application, however, the external perspective becomes much more relevant. In particular, the examples for preferential disambiguation cited in Section 10.2 illustrate two important points:

1. when building NLP systems the respective application poses requirements which do not necessarily correspond to research issues in theoretical Computational Linguistics;

2. in order to meet these requirements, information from various levels has to be taken into account, thereby treating components like syntax or semantics as service modules.

For illustration, consider some of the information needed for determining speech-event types [34, 35]. To do so one has to know, for example,

- whether the sentence is declarative or interrogative,
- whether it contains certain keywords,
- whether it contains an explicit temporal description,
- whether it contains an anaphoric reference to a temporal discourse referent.

How this is actually represented in the syntactic structure or the semantic representation is completely irrelevant. In other words, from an external perspective we want to *abstract* over the particular syntactic and semantic representations. This is usually achieved in software engineering by using *abstract data types*, instead of parsing the internal data structures used in other modules.
An abstract data type is independent of its representational implementation and defined by the access predicates (updates and queries) it provides, e.g. [11, Chap. 2]. In the context of NLP, one can distinguish two different types of access predicates, namely predicates providing atomic information about linguistic entities (e.g. case, tense, etc.) and predicates providing information about relations between complex entities e.g. syntactic relations between linguistic entities, such as subject, head daughter, etc.). Access predicates thus look like

case(+Sign,-Case)
subject(+Sign,-Subject)

Furthermore, not all access predicates make sense for all syntactic objects, e.g. 'case' is not meaningful for verbs, 'subject' is not meaningful for prepositions, etc. These restrictions can be captured by introducing a type hierarchy. Access predicates can then be specified for the most general type they apply to and are inherited to subtypes.
Note that an important advantage of using abstract data types for syntactic representations is that one can thereby abstract over the particular realizations in different syntactic theories. This can be illustrated by considering an access predicate like 'subject(Sign,Subject)'. Even though syntactic theories like GB, HPSG, or LFG differ wrt the treatment of subject, the disctinctions can be abstracted over, so that syntactic modules realizing different syntactic theories behave identical in all, or at least in most cases.

Such an interface between syntax and semantics relying on abstract interface predicates has been realized in the semantic interpretation system SCOLD [21]. Defining access predicates is thus a step towards a *standardization* of syntactic theories. Note that this standardization only concerns the external perspective, i.e. it does not necessarily have an impact on the internal perspective of syntactic modules.

Note also that the DL representation of linguistic structure, conceptual content, and reference proposed in [27] can be seen as an object-oriented realization of abstract data types.[1] The main distinction between DL and the typed feature structures used in Computational Linguistics [7] is in fact that DL explicitly distinguish between types and objects, whereas typed feature structures do not support an explicit object level but only provide types.

Due to the missing object level in typed feature structures, exception minimization cannot be defined straightforwardly. (Note that the exceptions are defined wrt a given set of objects). Furthermore, access predicates can easily be defined in DL by using role-forming operators as inversion, composition, domain, or range, which are not available in typed feature structures.

Finally, consider the relationship between object-oriented representations and non-sequential architectures. In [12, 2.4.3] four different architectures are distinguished, namely

- a *linear or sequential* architecture, in which each component provides information to the subsequent component;
- a *cascaded* architecture, which is still linear but allows communication in two directions, i.e. a component provides information also to the componente preceeding it;
- a *blackboard* architecture, in which components share a common data structure;
- a *heterarchical* architecture, in which each component can communicate with all other components.

It should be obvious that object-oriented representations are compatible with all four architectures. It should also be obvious that passing entire data structures without providing any access predicates, i.e. without any information hiding, maybe feasible for sequential and cascaded architectures, but is definitely not feasible in blackboard or heterarchical architectures.

The main advantage of blackboard and heterarchical architectures is that they allow incremental and parallel processing, which is especially relevant in application-oriented NLP systems [20]. Consider again determination of speech-event types in automatic dialogue interpreting as an example. In many cases this determination and the translation based on it does not require deep syntactic or semantic analysis. It is thus possible to realize the concept of a *variable depth of analysis* in which information is only computed if it is required by other components (or needed to compute information required by other components).

The specification of access predicates is thus not only a step towards standardization. On the basis of the actual use of access predicates in a particular application it is possible to optimize a module wrt the specific requirements of this application.

[1] According to Booch, object-oriented programming combines abstract data types and inheritance [4, p. 38].

10.5 Conclusion

In this paper I have briefly described the general characteristics of preferential disambiguation and the technical realization in terms of weighted defaults and exception minimization. Though a homogeneous representation of different information levels, e.g. in terms of Description Logics (DL), is advantageous from a theoretical point of view, preferential disambiguation can also be achieved for heterogeneous architectures. I have argued that this requires that the heterogeneous representation structures, e.g. syntactic and semantic structures, are treated as abstract data types. The most important advantages of this approach are:

- a separation between an internal and an external perspective on the representation structures;
- a step towards standardization of syntactic and semantic representations;
- a step towards more flexibility wrt the general architecture, e.g. incremental and parallel processing.

Whereas the object-oriented representations in terms of DL can be immediately used as abstract data types, this is not possible for representations based on typed features structures.

Acknowledgements

The idea of treating syntactic and semantic representations as abstract data types owes much to discussions with Scott McGlashan, Sebastian Millies, and C.J. Rupp. The interfaces implemented in the VERBMOBIL demonstrator are a first step towards realizing this idea.

Bibliography

[1] J. Allen, *Natural Language Understanding*, Menlo Park: Benjamin/Cummings, 1987

[2] F. Baader, B. Hollunder, "How to Prefer More Specific Defaults in Terminological Default Logic", *IJCAI-93*, 669–674

[3] F. Bond, K. Ogura, S. Ikehara, "Countability and Number in Japanese to English Machine Translation", *Coling-94*, 32–38, 1994

[4] G. Booch, *Object-Oriented Analysis and Design*, Redwood City: Bejamin Cummins, 1994[2]

[5] R. Brachman, D.L. McGuiness, P.F. Patel-Schneider, L. Alperin Resnick, A. Borgida, "Living with CLASSIC: When and How to Use a KL-ONE -like Language", in J. Sowa (Ed.), *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, San Mateo: Morgan Kaufmann, 1991, 401–456

[6] G. Brewka, *Nonmonotonic Reasoning: Logical Foundations of Commonsense*, Cambridge: Cambridge University Press, 1991

[7] B. Carpenter, *The Logic of Typed Feature Structures*, Cambridge: Cambridge University Press, 1992

[8] F.M. Donini, M. Lenzerini, D. Nardi, W. Nutt, "Tractable Concept Languages" *IJCAI-91*, 458–463

[9] D. R. Dowty, R. E. Wall, S. Peters, *Introduction to Montague Semantics*, Dordrecht: D. Reidel, 1981

[10] K. Eberle, W. Kasper, C. Rohrer, "Contextual Constraints for MT", *Proceedings of the TMI'92*, 213–224, 1992

[11] H. Ehrig, B. Mahr, *Fundamentals of Algebraic Specification 1*, Berlin: Springer, 1985

[12] G. Görz, *Strukturanalyse Natürlicher Sprache*, Addison Wesley, 1988

[13] J.R. Hobbs, M. Stickel, D. Appelt, P. Martin, "Interpretation as Abduction", *Artificial Intelligence* **63**, 69–142, 1993

[14] B. Kipper, "Ambiguitätsprobleme bei der Modalverbanalyse", in [30], 81–103

[15] , L. Konieczny, B. Hemforth, N. Voelker, "The Impact of Context and Semantic Bias on Constituent Attachment in Reading", in [30], 105–126

[16] S. Kraus, D. Lehman, M. Magidor, "Nonmonotonic Reasoning, Preferential Models and Cumulative Logics", *Artificial Intelligence* **44**, 167–207, 1990

[17] V. Lifschitz, "Computing Circumscription", *IJCAI-85*, 121–127

[18] S. LuperFoy, E. Rich, *A Computational Model for the Resolution of Context Dependent References*, Austin: MCC Technical Report, 1990

[19] D. Makinson, "General Patterns in Nonmonotonic Reasoning", in D. Gabbay, C. Hogger, J. Robinson (eds), *Handbook of Logic in Artificial Intelligence*, Oxford: Oxford University Press, 1994

[20] W. Menzel, "Parsing of Spoken Language under Time Constraints", *ECAI-94*, 560–564, 1994

[21] S. Millies, M. Pinkal, "Linking One Semantic Interpretation System to Different Syntactic Formalisms", in M. Pinkal, R. Scha, L. Schubert (eds), *Semantic Formalisms in Natural Language Processing*, Dagstuhl Seminarreport 57, 1993, 35–39

[22] M. Minsky, "A Framework for Representing Knowledge", in P.H. Winston (Ed.), *The Psychology of Computer Vision*, New York: McGraw-Hill, 1975, 211–277

[23] F.C.N. Pereira, M.E. Pollack, "Incremental Interpretation", *Artificial Intelligence* **50**, 37–82, 1991

[24] S. Preuss, B. Schmitz, C. Hauenschild, C. Umbach, "Anaphora Resolution in Machine Translation", in W. Ramm (Ed.), *Text and Context in Machine Translation: Aspects of Discourse Representation and Discourse Processing*, Brussels: European Commission, 1994, 29–52

[25] J.J. Quantz, "Interpretation as Exception Minimization", *IJCAI-93*, 1310–1315

[26] J.J. Quantz, "An HPSG Parser Based On Description Logics", *COLING'94*, 412–416

[27] J.J. Quantz, *Preferential Disambiguation in Natural Language Processing*, PhD Thesis (submitted), Technische Universität Berlin, 1995

[28] J.J. Quantz, V. Royer, "A Preference Semantics for Defaults in Terminological Logics", *KR-92*, 294–305

[29] J.J. Quantz, B. Schmitz, "Knowledge-Based Disambiguation for Machine Translation", *Minds and Machines* **4**, 39–57, 1994

[30] J.J. Quantz, B. Schmitz (eds), *Ambiguity and Strategies of Disambiguation*, KIT Report 120, Technische Universität Berlin, 1994

[31] J.J. Quantz, S. Suska, "Weighted Defaults in Description Logics—Formal Properties and Proof Theory", in B. Nebel, L. Dreschler-Fischer (eds), *KI-94: Advances in Artificial Intelligence*, Berlin: Springer, 1994, 178–189

[32] M.R, Quillian, "Semantic Memory", in M. Minsky (Ed.), *Semantic Information Processing*, Cambridge (Mass): MIT Press, 1968, 216–270

[33] , M. Ryan, "Representing Defaults as Sentences with Reduced Priority", *KR-92*, 649–660

[34] B. Schmitz, S. Jekat-Rommel, *Eine zyklische Approximation an Sprechhandlungs-typen–zur Annotierung von Äußerungen in Dialogen*, Verbmobil Report 28, 1994

[35] B. Schmitz, J.J. Quantz, "Speech-Event Types in Automatic Dialogue Interpreting", submitted for publication

[36] P. Sells, *Lectures on Contemporary Syntactic Theories*, Stanford: CSLI Lecture Notes 3, 1985

Kapitel 11

Contraint Logic Interaction (CLI) — Ein neues Interaktionskonzept (nicht nur) für sprachverarbeitende Systeme

Thomas Stürmer
IBM Deutschland Informationssysteme GmbH,
Institut für Logik und Linguistik,
thst@heidelbg.ibm.com

Zusammenfassung

Wir werden ein Konzept zur Interaktion von Modulen eines integrierten Systems vorstellen, das auf der Verarbeitung logischer intermodularer Constraints, d.h. auf Bedingungen zwischen modulinternen Zuständen jeweils verschiedener Module, beruht. Wir wollen dieses Konzept *Constraint Logic Interaction* (CLI)[1] nennen.
Eine aus diesem Konzept abgeleitete Systemarchitektur für sprachverarbeitende Systeme (SVSe) stellt zusammen mit spezifischer linguistischer Information über Interaktion eine Schritt in Richtung optimaler Kommunikation desambiguierender Information in einem modulare SVS dar.

11.1 Das Problem

Integrierte SVSe, wie sie in jüngster Zeit gefordert werden [Pereira & Grosz, 1993, Seite 1] und Vision so ehrgeiziger Projekte wie Verbmobil sind, sind auf absehbare Zeit nicht in Sicht. Trotz der enormen Anstrengungen, die z.B. in vielen Verbundprojekten (LILOG, ASL, Verbmobil) unternommen wurden und werden, sind Insellösungen für linguistische Teilprobleme oder Laborprototypen von inakzeptabler Performanz Stand der Kunst [Seiffert, 1992].

[1]Die Ähnlichkeit zu CLP, (...*Programming*) ist nicht ganz unbeabsichtigt.

11.1.1 Beobachtungen

Nahezu jede natürlichsprachliche Äußerung ist in vielerlei Hinsicht mehrdeutig oder vage. Das Verstehen einer Äußerung wie *"Besser mit dem kleinen!"* hängt von zahlreichen Faktoren ab, die im Kontext, Kotext oder sonstigem Wissen bestimmt sind und die bei einer Interpretation berücksichtigt werden müssen. Das Fehlen expliziter Information über solche Faktoren ist in zwischenmenschlichen Dialogen durchaus erwünscht: Bereits gesagtes, im sozio-kulturellen Umfeld definiertes, in nichtverbalem Handeln ausgedrücktes bedarf keiner expliziten sprachlichen Realisierung und über verbleibende Unbestimmtheiten wird in der Regel treffsicher spekuliert. Andererseits sind genau jene Unbestimmtheit und (unentdeckte) Divergenzen in o.g. zugrundegelegten Annahmen Grund für Mehrdeutigkeiten und damit für (gewollte und ungewollte) Mißverständnisse.
Diese Unbestimmtheit ist natürlicher Sprache inhärent und ihre Behandlung in einem SVS ein Riesenproblem. Die explizite Repräsentation dieser Faktoren hat (nicht nur) praktische Grenzen: es kann unmöglich alles modelliert werden, was einmal für irgendeine Spracheingabe wichtig werden könnte[2].
SVSe sind traditionell aus Modulen aufgebaut, die üblicherweise mit den wohlbekannten linguistischen Abstraktionsebenen korrespondieren. Einerseits ist Modularisierung zur Reduktion der Komplexität des Gesamtproblems auf handhabbare Teilprobleme unumgänglich, andererseits ist sie Ursache für modulinterne Ambiguität: Information, die konzeptionell Gegenstand der Problemlösung anderer Module ist, für einen aus der Sicht des Gesamtproblems konsistenten Fortgang der Teilproblemlösung aber dringend benötigt wird, ist nun nicht mehr ohne weiteres verfügbar (eine Ausnahme bilden constraintbasierte Grammatikformalismen, die in einem einheitlichen Formalismus abgearbeitet werden, vgl. Abschnitt 11.2). Dadurch wird die o.g. Unbestimmtheit systematisch vergrößert und das Problem „künstlich" verschärft, da Mehrdeutigkeiten entstehen, die in natürlicher Sprache eigentlich nicht enthalten sind.
Darüberhinaus entstehen bei der Behandlung einiger natürlichsprachlicher Phänomene, wie der Auflösung anaphorischer Referenzen, zusätzliche Schwierigkeiten, da sie sich nicht lokal in einem Modul bearbeiten lassen, sondern mglw. mehrere andere Module affektieren.
Soll nun ein komplexes NLS aus solchen traditionellen Modulen aufgebaut werden, bedarf es elaborierter Hilfsmittel für ihre Kombination zu einem Gesamtsystem.

11.1.2 Design komplexer Systeme (ein kurzer Ausflug)

Um das Folgende exakt einordnen zu können, lohnt es sich, einen kleinen Abstecher in die altbekannte Begriffswelt des Programmentwurfs zu unternehmen[3].
Die Entwicklung eines komplexen Programmsystems beginnt mit der *Problemanalyse* (vgl. Abb. 11.1). Sie ist, wie im Fall der Linguistik für SVSe, mglw. Gegenstand eines eigenen Forschungsbereichs. Typischerweise werden schwierige Probleme in einfachere *Teilprobleme* zerlegt, deren Lösung entweder bekannt ist, oder die ihrerseits schrittweise vereinfacht werden, usw. So entstehen sukzessive Ebenen fallenden Abstraktionsgrades bzw. wachsender Detailiertheit einer Problembeschreibung.
Die jeweiligen Teilprobleme sollten möglichst voneinander unabhängig sein; dies hat

[2] Für vieles andere, z.B. Gestik oder Mimik, fehlen schlicht die perzeptiven Möglichkeiten.

[3] Hier soll keinesfalls eine Diskussion über Terminologie entfacht werden. Es geht hier „nur" um das Schaffen einer gemeinsamen Basis zur Identifikation der Ursachen von Fehlentwicklungen.

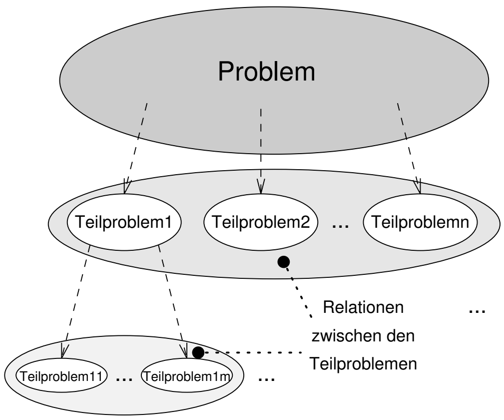


Abbildung 11.1: Problemanalyse durch Dekomposition

den angehmen Effekt, daß deren weitere Untersuchung jeweils weitgehend isoliert stattfinden kann.
Extrem wichtig für die Beschreibung des Gesamtproblems sind jedoch auch die *Relationen*, in denen identifizierte Teilprobleme einer Abstraktionsebene zueinander stehen. Analog zu den Teilproblemen müssen auch sie eine schrittweise Verfeinerung erfahren, da sonst die Komposition der Beschreibung der Teilproblemlösungen zur globalen Problemlösung nicht gelingen kann.
Auf der Basis dieser Problemanalyse wird ein Modell für das zu lösende Problem erstellt: Teilproblemlöser werden als Module des PS, die gemäß der identifizierten Relationen miteinander interagieren, verbunden. Dies führt zu einer problemnahen formalen Spezifikation des Problems, der eine ebenso problemnahe Vorstellung eines Verarbeitungsmodells zugrunde liegt; beides zusammen ergibt den gesuchten PS. Im Bereich der Computerlinguistik wäre bspw. das Resultat einer solchen Untersuchung der Syntax der deutschen Sprache „ihre“ Notation mit den Mitteln einer bestimmten Grammatiktheorie, also in einem konreten Grammatikformalismus und das Verarbeitungsmodel eine bestimmte Form des Parsings.
Zwischen dem problemnahen Modell des PS und einer programmiersprachlichen Beschreibung besteht in der Regel eine (mehr oder weniger große) konzeptionelle Distanz, die durch den evtl. mehrstufigen Prozeß der *Implementierung* überwunden werden muß (vgl. Abb. 11.2). Die Schritte einer Implementierung prägen die durch die Modellbildung gewonnene Strukturierung – also des PSs – in Richtung des Programmierparadigmas aus. Das bedeutet insbesondere, daß letzten Endes jedes Modul und jegliche Interaktion eine programmiersprachliche Realisierung zu erfahren hat.
Bei der Nachbildung von Aspekten menschlicher Kognition, z.B. der Problemlösefähigkeit oder der Verarbeitung natürlicher Sprache, ist bereits die Modellbildung höchst kontrovers diskutierter Gegenstand einer ganzen wissenschaftlichen Disziplin und ein abschließendes Ergebnis – soweit überhaupt möglich – nicht in Sicht. Oft ist eine Implementierung die einzige Möglichkeit der Validierung eines solchen Modells und wird somit selbst zu einem Teil einer iterierten Theoriebildung innerhalb der Disziplin: Ein Zyklus aus Problemanalyse, Modellbildung und Implementierung wiederholt sich beliebig oft.

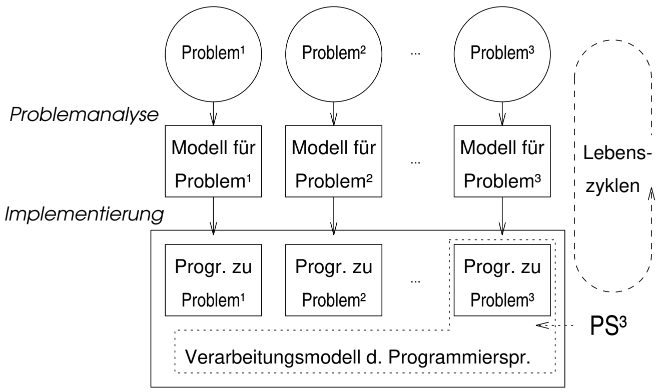


Abbildung 11.2: Problemanalyse, Modellbildung, Implementierung

11.1.3 Und SVSe?

Der Bereich der Linguistik hat (bereits lange vor der Konstitution der sog. Computerlinguistik) eine Aufspaltung in seine Teildisziplinen erfahren, die dann als solche (bis auf wenige Ausnahmen) gänzlich isoliert voneinander entwickelt wurden. Die Beziehungen zwischen den resultierenden linguistischen Teiltheorien wurden nie in dem für eine Integration erforderlichen Maße erforscht; so ist der Bereich der Interaktion bereits auf linguistischer Ebene vergleichsweise unterentwickelt und eine Anforderung entsprechender Verarbeitungsmodelle wurde an die Systementwickler nie gestellt[4].
Eine Ausnahme und einen kleinen Fortschritt stellt die Entwicklung von Formalismen in Teildisziplinen dar, deren Inventar sich auch zur Beschreibung der Problemlösung anderer Teildisziplinen eignet und Verwendung findet (z.B. merkmalsbasierte Grammatiktheorien wie LFG oder HPSG, s. dazu Abschnitt 11.2). Aber auch sie stellen erst einen möglichen Rahmen für eine detailierte Untersuchung dar, und lösen auf keinen Fall das Problem der Interaktion zwischen heterogenen Ansätzen, die eine solche gemeinsame Basis nicht finden.
Größere Hoffnung setzt der Autor in die jüngere Entwicklung sog. *Linking Theorien*, die bisher noch einen sehr engen Anwendungsbereich haben, und – hier von größerem Interesse – bisher zu keiner Entwicklung entsprechender Verarbeitungsmodelle führte.

11.1.4 Kleiner Wunschzettel

Entscheidend für die Konstruktion eines SVSs aus Modulen ist neben deren Qualität, der bisher das Hauptaugenmerk galt, die Qualität der Modellierung der Relationen zwischen ihnen, und zwar auf allen Ebenen der Problemanalyse und Implementierung. Aus den oben dargestellen Schwierigkeiten mit der Komplexität der Verarbeitung natürlicher Sprache lassen sich folgende Anforderungen an ein SVS ableiten:

1. Ein modular aufgebautes SVS kann nur durch Kooperation der Module zufriedenstellend arbeiten. Notwendig ist ein Konzept zur effizienten Koordination jeweils modullokalen Problemlöseverhaltens.

[4] Die Vertreter anerkannter linguistischer Teildisziplinen waren und sind im Grunde froh, in der Relation ihrer lokalen Problemlösung zu der anderer Teildisziplinen vage und schematisch bleiben zu können.

2. Ein kooperierendes System muß notwendigerweise aus zeitsynchron arbeitenden Modulen bestehen, da nur dann kooperatives Verhalten möglich ist.

3. Damit ist ein solches System auch zu einer inkrementellen Arbeitsweise gezwungen, wobei die Größe der Inkremente durchaus modulspezifisch variieren kann.

Als Beitrag der Informatik ist also eine Systemarchitektur erforderlich, die, vorbehaltlich der Existenz solcher problemspezifischer Daten, direkt die Vorgabe der *interaktiven inkrementellen Verarbeitung* unterstützt.
Intermodulare Kommunikation desambiguierender Information allein wird jedoch i.allg. nicht zu einer eindeutigen Lösung eines Sprachverarbeitungsproblems führen. Daher sollte ein SVS in der Lage sein, anstelle aller möglichen nur das *plausibelste* Ergebnis zu berechnen. Die gewünschte Systemarchitektur soll einer entsprechenden Erweiterung nicht im Wege stehen.[5]

11.2 Interaktion, aber wie?

11.2.1 Interaktive Architekturen

Es liegt nahe, ein SVS modular aus auf ihre Aufgabe hin optimierte Problemlöser aufzubauen, und die notwendige Kooperation auf den gezielten Austausch von Nachrichten in einem Kommunikationssystem abzubilden. Solche Ansätze werden oft auch etwas respektlos *„AI approaches“* genannt. Wir wollen sie i.folg. *interaktive Architekturen* nennen.
Architekturansätze für die Behandlung von Interaktion zwischen Modulen sind z.B. Blackboardsysteme, Hierarchische Systeme, Multiagentensysteme, etc. Einige Ansätze wurden speziell für die psycholinguistisch motivierte Sprachverarbeitung entworfen, z.B. die von [Görz, 1992] vorgeschlagene Interaktive Inkrementelle Architektur (IIA) in ASL.

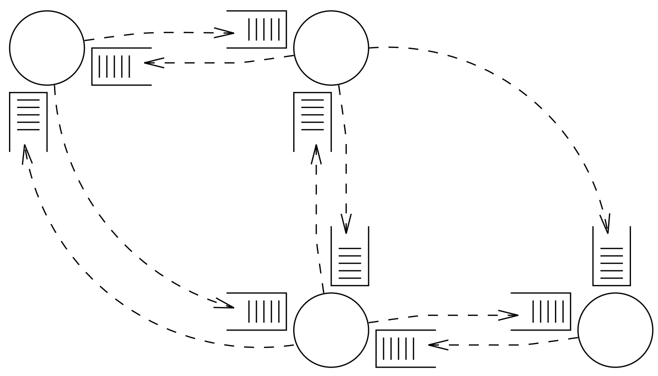


Abbildung 11.3: Direkte Kommunikation zwischen Modulen in einer Multiagentenumgebung

[5] Dieser Aspekt soll an dieser Stelle jedoch nicht vertieft werden. Ein Vorschlag zu Behandlung und eine Vielzahl einschlägiger Referenzen findet sich in [Stürmer, 1993].

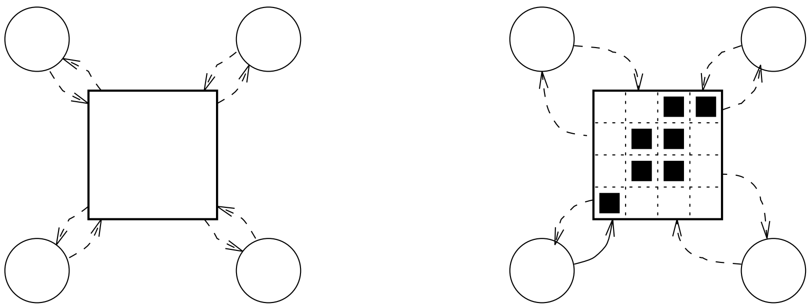


Abbildung 11.4: Kommunikation von Modulen über eine unstrukturierte (links) und eine unstrukturierte Blackboard (rechts, schwarz gekennzeichnete Felder erlauben keine Einträge).

Diese Ansätzen charakterisiert ist ein kaum handhabbares *Kontrollproblem*. Dabei ist es prinzipiell unerheblich, ob die Kommunikation über spezielle Kanäle oder strukturierte wie unstrukturierte Blackboards, etc., läuft. Unter Berücksichtigung des Vorangegangenen ist dies auch keineswegs überraschend: In sämtlichen uns bekannten Systemen, die auf einer interaktiven Architektur aufbauen, bestimmen linguistische und psycholinguistische Erkenntnisse allenfalls die Art der Verbindungen, nicht aber deren situativen Inhalt. Das Architekturmodell soll mit Mitteln der Informatik des Fehlen linguistischer Information beheben; das kann nur scheitern.
Andererseits verfügen die genannten Ansätze über keine Interaktionskonzepte, die es erlauben würde, mit dynamischer Information dieser Art umzugehen. Dies ist jedoch kein prinzipieller Mangel: Entsprechende Erweiterungen sind möglich (vgl. Abschnitt 11.3).

11.2.2 Interaktive Formalismen

Interaktion kann auf einem sehr hohen Abstraktionsniveau bei der Verwendung merkmalsbasierter Grammtikformalismen, z.B. HPSG [Pollard & Sag, 1987] beschrieben werden. Dabei kann in einer Datenstruktur auf dieselbe Weise Wissen aus verschiedenen linguistischen Abstraktionsebenen repräsentiert und aufeinander bezogen werden. Die Sicht auf Merkmale als Constraints führt zur *„constraint pool metaphor“* [Fenstad et al., 1987] als Verarbeitungsmodell: Constraints aus den unterschiedlichsten Wissensquellen werden in einem sog. Constraint Pool zusammengeworfen und von einem uniformen Verfahren gelöst. Zusammen mit Entwicklungen im Bereich *Constraint Logic Programming* wird eine fast unmittelbare Implementierbarkeit gewährleistet. Solche Ansätze bestechen vor allem durch ihre theoretischen Eigenschaften.
Über die Möglichkeit der Koindizierung von Teilstrukturen stehen elegante Mittel zur Beschreibung von Interaktion zur Verfügung (was sicher ein Grund für die Popularität von HPSG ist). Da Koreferenzen zwischen Merkmalen auch nur einfache Constraints sind, bedarf es keiner gesonderten Behandlung, sie werden einfach mitgelöst. Fehlende Information kann (innerhalb gewisser Grenzen) durch gezielte Unterspezifikation behandelt werden.
Die Integration der verschiedenen linguistischen Ebenen findet im wesentlichen bereits in den Datenstrukturen des Lexikon statt. Dadurch entstehen riesige Klausel-

mengen, die Information im ganzen Spektrum zwischen Phonologie und Semantik und so aufwendige Sprachmittel wie Listen, Disjunktionen, Negationen oder Mengen enthalten können. Die in der Datenstruktur vorhandene Modularität, die beizubehalten für den Berechnungsaufwand günstig wäre, geht verloren, und Kontrollmöglichkeiten, mit deren Hilfe die Inferenzschritte gesteuert werden können, fehlen weitgehend: Die Berechnungskomplexität ist katastrophal [Seiffert, 1992].
Daneben scheint es schwierig, signifikante Teile aus den Bereichen Semantik und Pragmatik zu behandeln ohne die Theorie zu erweitern. Merkmalsstrukturen scheinen zwar ganz allgemein zur Repräsentation linguistischer Information gut geeignet zu sein, aber Merkmalsunifikation (oder Varianten) als uniformer Verarbeitungsmechanismus scheint zu schwach für alle Aufgaben innerhalb eines SVS[6]. Um die gewünschte Abdeckung zu erreichen muß dann trotzdem auf interaktive Architekturen ausgewichen werden.

11.2.3 Die Lehren daraus

Koindizierung in merkmalsbasierten Formalismen haben zum einen den Charakter von *Abfragen*, mit deren Hilfe die jeweils Ebenen-lokale Problemlösung koordiniert wird: Disjunktionen werden desambiguiert, Fälle unterschieden. Zum anderen dienen sie dem *Datenaustausch* zwischen verschiedenen Ebenen.
In einer detailierten Untersuchung der Interaktion zwischen Syntax (c-Struktur und f-Struktur) und Semantik in der LFG entwickelt [Halvorsen, 1988] ein logisches Modell der Syntax-Semantik-Schnittstelle, das über die Verwendung von interaktiven Formalismen abstrahierbar ist (s. Abschnitt 11.3). Er beschreibt Interaktion zwischen diesen Ebenen mit Hilfe von

- Projektionen, die jeweils bestimmte Strukturen der Semantik, der funktionalen oder der Konstituentenstruktur bezeichnen, und
- Constraint-Gleichungen zwischen diesen Strukturen.

Interaktive Architekturen und interaktive Formalismen in SVS sind streng genommen nicht miteinander vergleichbar; sie befinden sich auf unterschiedlichem Abstraktionsniveau: Die Formalismen stehen dem Linguisten unmittelbar zur Notation seiner Theorie zur Verfügung, während Architekturen sehr implementationsnahe Konzepte sind. Es ist sogar denkbar (und scheint uns ausgesprochen vielversprechend), interaktive Formalismen durch Systeme mit interaktiver Architektur effizient zu verarbeiten.
Dennoch stellen beide unter dem Aspekt der Behandlung von Interaktion konkrete Verarbeitungsmodelle dar, die durch Anleihen voneinander profitieren können.

11.3 Constraint Logic Interaction (CLI)

Wir wollen die Modularisierung des PSs und damit eine Partitionierung der Informationsmenge beibehalten. Es soll möglich sein, gute lokale (hochspezialisierte) Teilproblemlöser zu verwenden, die keine gemeinsamen Datenstrukturen und Inferenzprozesse verwenden.

[6] Da merkmalsbasierte Formalismen i.allg. Turing-mächtig sind, ist das natürlich kein prinzipielles Problem. Es geht hier vielmehr um die Frage, ob die Verarbeitung für das Problem adäquat ist.

Ein dynamisches Interaktionskonzept, das als ein Modell für eine Implementierung von interaktiven logischen Constraints (*Constraint Logic Interaction* (CLI)) gesehen werden kann, soll Kooperation der Module ohne zusätzlich entstehendes Kontrollproblem unterstützen.
Bestandteil von CLI sind

Projektionen: Entstehen im Lauf der Bearbeitung eines Problems Korrespondenzen zwischen internen Zuständen unterschiedlicher Module, werden diese mit Hilfe modulspezifischer Projektionen außerhalb „sichtbar“ gemacht.

Perspektiven: Interne Daten eines Moduls haben i.allg. ein spezielles internes Datenformat, das die Lösung der spezifischen Aufgaben unterstützt. Daher schließt eine Projektion von Zuständen deren Übersetzung in ein mglw. neutrales oder für ein anderes Modul passendes Format mit ein.

Constraints: Über den projizierten Zuständen können Constraints formuliert werden, die Zusammenhänge zwischen Zuständen über Modulgrenzen hinweg repräsentieren. CLI unterstützt Gleichheit und aussagenlogische Constraints über booleschen Zuständen.

11.3.1 Projektionen

Idealerweise sind Module “black boxes”, d.h. Details ihrer Implementierung sind nach außen weder sichtbar noch zugreifbar oder gar modifizierbar. Kommunikation mit der „Außenwelt“ läuft über vordefinierte Schnittstellen.
Sollen nun im Lauf der Bearbeitung eines Problems modulinterne Zustände außerhalb sichtbar werden, kann dies mithilfe einer speziellen Projektion geschehen (vgl. Abb. 11.5). Interne Daten eines Moduls haben i.allg. ein spezielles internes Datenformat, das

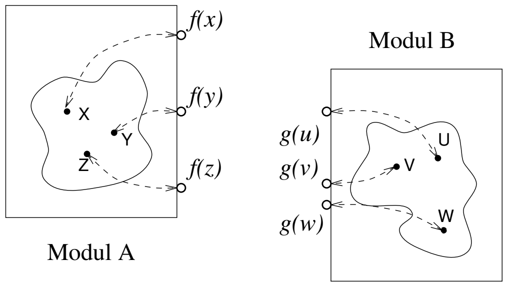


Abbildung 11.5: Projektionen und Perspektiven f und g als Interface zu internen Zuständen der Module A und B.

die Lösung der spezifischen Aufgaben unterstützt. Daher muß eine sinnvolle Projektion von Zuständen deren *Transformation* in ein mglw. neutrales Format mit einschließen.

11.3.2 Constraints

Über den projizierten Zuständen können Constraints formuliert werden, die Zusammenhänge zwischen Zuständen über Modulgrenzen hinweg repräsentieren, sog. *intermodulare Constraints* (vgl. Abb. 11.6). Da jeder modifizierende Zugriff eines Moduls

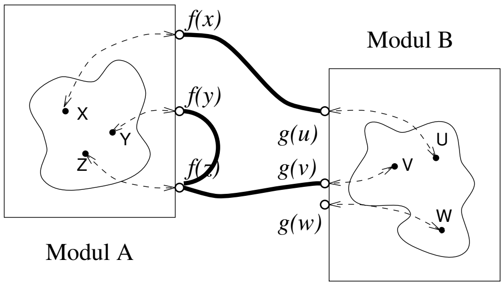


Abbildung 11.6: Constraints über projizierten Zuständen

auf einen projizierten Zustand auch global unmittelbar beobachtbar ist, laßt sich die Konsistenz eines Systems solcher Constraints auch global kontrollieren. Dies kann in einem eigens dafür konstruierten Constraintlösemodul geschehen (s. Abb. 11.7).
Ein Zustand, der mit einem anderen Zustand korrespondiert, kann nicht mehr jeden beliebigen Wert annehmen, ohne die globale Konsistenz des Systems zu verletzen.

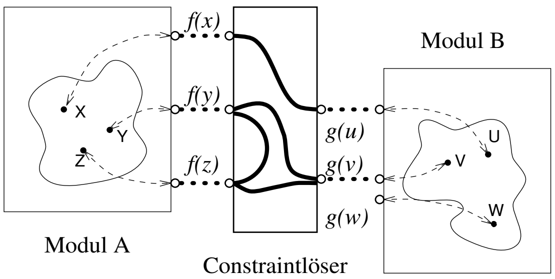


Abbildung 11.7: Die globalen Constraints werden in einem speziellen Modul, dem Constraintlöser, verwaltet. Im Falle einer Verletzung von Constraints stellt dieses Modul Dienste zur Verfügung, die beim Bereinigen des Konflikts helfen.

Sollte ein Zustand intern modifiziert werden, der schon durch externe Constraints auf einen anderen Wert festgelegt ist, muß eine geeignete Konfliktbehandlung in Gang gesetzt werden. Die Konfliktbehandlung kann die Arbeit der Module unterbrechen und den Clash aufgrund der Datenlage im Constraintlöser selbst, z.B. durch irgendeine Präferenz, bereinigen, oder den Modulen eine Einigung überlassen. Ggf. muß der Constraintlöser Hilfen zur Rücknahme von bereits getroffenen Annahmen zur Verfügung stellen.
Betrifft die geplante Modifikation einen unrestringierten Zustand, wird sie vorgenommen und durch die Projektion automatisch dem Constraintlöser mitgeteilt; dieser propagiert einen entspechenden Wert für jedes affektierte Constraint an seinen Empfänger, der die interne Projektion nun in seiner lokalen Annahmenmenge vorfindet.
Besonders attraktiv ist dieses Vorgehen, wenn betroffene Module eine Art *intelligentes Backtracking* ermöglichen (z.B. durch Earley-Deduktion, wenigstens aber einer Verwaltung bereits errechneter wichtiger Zwischenergebnisse).

11.3.3 Zu Ausdruckskraft und Komplexität von CLI

Wie bereits oben festgestellt (Abschnitt 11.2.3), erschöpft sich das interaktive Potential von merkmalsbasierten Formalismen mit den Möglichkeiten der Koindizierung. Dies entspricht in unserem Ansatz dem Constraint der *Gleichheit* (modulo Projektion).
Die Verwendung von Koindizierung als Abfrage (eigentlich die Kommunikation eines Datums mit der Feststellung der Konsistenz) erscheint uns als unangemessen aufwendig. Effizienter und transparenter (weil klar ist um was es dabei eigentlich geht), ist die Verwendung einfacher *boolescher Constraints*. In diesem Fall ist auch die Frage der Perspektiven bei Projektionen trivial: Wahrheitswerte sind, falls Sie in ihrer modulspezifischen Repräsentation nicht von vornherein übereinstimmen, leicht realisierbar. Es steht jedoch zu erwarten (und zu hoffen), daß in Zukunft elaborierte Linking Theorien stärkere Ausdruckmittel, z.B. *volle Aussagenlogik*, erfordern.
Diese Constraints werden ohne Kontrollproblem in einem Constraintlöser verarbeitet. Kern ist ein sog. generisches Reason-Maintenance-System (RMS) [Beckstein, 1994]. Gleichheit und von einem Kontrollmodul (z.B. probabilistisch) gesteuerte Auswahl von Alternativen wird effizient auf je einer zusätzlichen Ebene behandelt, lediglich die *Erfülltheit* der Constraints wird von dem RMS verwaltet (vgl. Abb. 11.8).

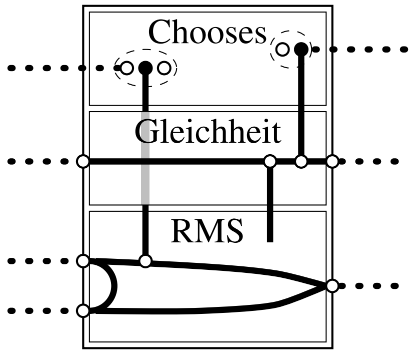


Abbildung 11.8: Der Constraintlöser im Detail

Die explizite Repräsentation modullokaler Annahmen bei der Alternativenwahl stellt die Voraussetzungen für ein plausibles globales Suchverfahren dar, das aber nicht das Thema dieses Beitrags ist.
Der Aufwand für die Gewährleistung der Gleichheitsconstraints ist linear zur Anzahl der Constraints und abhängig von der Komplexität des RMS (da die Abhängigkeiten zu anderen Constraints dort verwaltet werden).
Die Komplexität der Interaktion insgesamt ist damit der des Update-Algorithmus des RMS äquivalent und abhängig von der Anzahl der projizierten Zustände. Für den einfachen Fall der Nachbildung der Koindizierung ist sie linear, und damit eigentlich geschenkt. Für den Fall voller Aussagenlogik kann als erste Näherung z.B. auf boolesche Constraintpropagierung (unter Verzicht auf Vollständigkeit) zurückgegriffen werden. Ansonsten ist sie natürlich exponentiell mit der Anzahl der projizierten Zustände. Aber selbst das muß relativiert werden, angesichts der NP-Vollständigkeit nahezu aller in SVS beteiligten Teilsysteme und der durch CLI erhaltenen Modularisierung (und somit der modullokalen Basis jener NP-Vollständigkeit).

Die Arbeit der Module ist jeweils zusätzlich belastet durch einen für jede Modifikation eines projizierten Zustands konstanten Aufwand, wobei an dieser Stelle keine Aussagen darüber möglich sind, wie häufig ein bestimmtes Modul einen bestimmten Zustand beschreibt (im Idealfall nur einmal).

11.4 Wertung

CLI ist ein Architekturmodell zur effizienten Verarbeitung intermodularer Constraints. Es ist dazu in der Lage, ohne ein Kontrollproblem den Erhalt der globalen Konsistenz der Problemlösung (nicht nur) in einem SVS zu sichern. Es ist um ein Modul zur plausible Steuerung der Verarbeitungsprozesses erweiterbar.
Es eignet sich sowohl zur Kopplung heterogener Module, als auch als effizientes Verarbeitungsmodell für die verteilte Implementierung homogener (bzgl. Datenstrukturen und Inferenzmechanismen) Systeme.
Das alles steht und fällt mit dem Vorhandensein regelhafter situativer linguistischer Beschreibung der Interaktion.

Literaturverzeichnis

[Beckstein, 1994] Clemens Beckstein. 1994. Architektur und logische Grundlage monotoner Systeme zur Begründungsverwaltung. Habilitationsschrift.

[Fenstad et al., 1987] Jens Erik Fenstad, Per-Kristian Halvorsen, Tore Langholm und Johan van Benthem. 1987. *Situations, language and logic*, volume 34. D. Reidel Publishing Company, Dordrecht, NL.

[Görz, 1992] Günther Görz. 1992. Kognitiv orientierte Architekturen für die Sprachverarbeitung. Technical Report ASL-TR-39-92, BMFT-Verbundprojekt ASL, Friedrich-Alexander-Universität Erlangen-Nürnberg.

[Halvorsen, 1988] Per-Kristian Halvorsen. 1988. Algorithms for semantic interpretation. In *Semantics and Computational Linguistics*, Lugano, CH.

[Pereira & Grosz, 1993] Fernando C.N. Pereira und Barbara J. Grosz. 1993. Introduction to the special volume NLP. *Artificial Intelligence*, 63(1–2):1–15, October.

[Pollard & Sag, 1987] Carl Pollard und Ivan A. Sag. 1987. *Information-based syntax and semantics I*, volume 13 of *CSLI lecture notes*. Center for the Study of Language and Information, Stanford, Cal.

[Seiffert, 1992] Roland Seiffert. 1992. How could a good system of practical NLP look like? IWBS-Report 228, IBM Deutschland GmbH, Wissenschaftliches Zentrum, Institut für Wissensbasierte Systeme (IWBS), Juli.

[Stürmer, 1993] Thomas Stürmer. 1993. Semantic-oriented chart parsing with defaults. In Klaas Sikkel und Anton Nijholt, Hrsg., *Parsing and Natural Language*, Twente Workshop on Language Technology, TWLT 6, Seiten 99–106, Enschede, NL. Universiteit Twente.

Kapitel 12

Performance strategies in cognitive parsing

Barbara Hemforth and Lars Konieczny
Centre of Cognitive Science
Institute for Computer Science and Social Research
University of Freiburg
Friedrichstr. 50, 79098 Freiburg
e-mail: barbara | lars@cognition.iig.uni-freiburg.de

Abstract

Psycholinguistic models of the human language processor usually postulate universal principles underlying processes of language comprehension and production which are derived from the cognitive architecture of the human mind. In recent years, language specific preferences that have been found for some constructions throw doubt on this assumption. In this paper, we will discuss current principle- and exposure-based accounts on human language processing in the light of crosslinguistic data on modifier attachment.

12.1 Introduction

One thing that is well known about human parsing is that in cases of structural ambiguities, not all structural alternatives are pursued until disambiguating information shows up later in the sentence. Ambiguities are resolved in favor of one or at least a small choice of structural possibilities. The question of which alternative is chosen in which cases and why it is chosen, is central to psycholinguistic studies on cognitive parsing. If the primarily preferred analysis fails, a reanalysis has to be performed or the incoming material has to be adjusted to what has been expected, i.e. it has to be repaired (Konieczny, Hemforth, & Scheepers, 1994). Processes of reanalysis or repair

[1] This research was supported by the German National Research Foundation (Deutsche Forschungsgemeinschaft, DFG, Str 301/4-3). We would like to thank Christoph Scheepers and Gerhard Strube for many helpful discussions, Kim O'Brien for her helpful comments on an earlier draft of this paper, Martin Ernst Kraus, Nicole Völker, and Thomas Mulack for running the subjects, and Matthias Nückles for helping us preparing the final manuscript.

are more or less costly, depending on the amount of restructuring necessary to arrive at a satisfactory analysis. If those processes are very costly, they show up in conscious garden-paths as in example (1). Restructuring that can be performed fast and easily, is often not noticed by the comprehender, but can be established in sophisticated psycholinguistic experiments.

(1) a. *The singer taught the lyrics became nervous.*

b. *Daš der Vertreter der Versicherungsgesellschaft wegen gelogen hat, war ein Skandal.*
That the broker the insurance company (gen or dat) because of lied, was a scandal.[2]
It was scandalous, that the broker lied because of the insurance company.

During the past 30 years, psycholinguists have tried to establish the strategies or principles that underlie parsing preferences. Many abstract models have been proposed to explain parsing preferences in a large variety of syntactic structures and in an increasing number of languages (e.g. Kimball, 1973; Fodor & Frazier, 1978; Wanner, 1980; Abney, 1989; Pritchett, 1992 and many others). Similar to abstract principles which are assumed to underlie grammar (e.g. Chomsky, 1986), those parsing models claim universality in that they are supposed to be valid for every possible language in the world. The main reason for that claim is that the cognitive parsing principles are derived from a general principle of cognitive economy. It is assumed that those principles result from the functioning of the architecture of the human language processing system which should be the same for all human beings in the world, independently of their mother tongue.

Of course, psycholinguistic models differ with respect to the hypotheses concerning the aspects of economy which are supposed to be central for disambiguation preferences. Some put most weight on purely structural economy like the principle of minimal attachment (2) and late closure (3); others propose a presuppositional minimalism with respect to encyclopedic and referential inferences combined with different readings of a given (partial) sentence structure (4).

(2) *minimal attachment*
Do not postulate any potentially unnecessary nodes. (Frazier, 1987, p. 562)

(3) *late closure*
If grammatically permissible, attach new items into the clause or phrase currently being processed (i.e. the phrase or clause postulated most recently). (Frazier, 1987, p. 562)

(4) *The principle of a priori plausibility*
If a reading is more plausible in terms of either general knowledge about the world, or of specific knowledge about the universe of discourse, then, other things being equal, it will be favored over one that is not. (Crain & Steedman, 1985, p. 330)

In our model, which we will describe very briefly in the next section, a principle of parametrized head attachment is proposed, which is supposed to guarantee that for purely structural reasons the sentence processor always first chooses the alternative

[1]German examples are first translated in German word order, then in the correct English word order.

which can be semantically evaluated in the most effective way. We will later focus on one of the principles of the Parametrized Head Attachment Model and discuss its universal validity in the light of data from crosslinguistic modifier attachment studies.

12.2 A universal account of human parsing: parametrized head attachment

In a series of experiments over the last few years we have investigated the initial attachment preferences in structurally ambiguous sentences like (1-2).

(5) a. *Susan schlug das Mädchen mit dem Buch.*
Susan hit the girl with the book.

b. *Daß Susan das Mädchen mit dem Buch schlug, ... That Susan the girl with the book hit, ...*

(6) a. *Daß der Lehrer der Tochter das Buch geliehen hat, ...*
That the teacher the daughter the book lent has, ...

b. *Daß der Lehrer der Tochter das Buch entdeckt hat, ...*
That the teacher of the daughter the book discovered has, ...

In (5), the PP "with the book" can either be attached to the VP, in which case "the book" is understood as an instrument of "hitting", or to the direct-object-NP, such that "the book" belongs to "the girl". Similarly, the NP "der Tochter" in (6) is ambiguously case marked, such that it could be attached as a genitive-case NP ("of the daughter") to the preceding NP "the teacher", or as dative-object to a verb. However, the verbs in (6a) and (6b) differ in that they either force (a) or forbid (b) the verb-argument reading.

Firstly, we ran a word-by-word self-paced reading experiment with sentences like (5). In the second series we investigated NP-attachment preferences (6) in a self-paced reading study, using both a word-by-word, and a segment-by-segment presentation method. The results of both presentation methods differed substantially. We re-ran both experiments using an eye-tracking method. In our most recent experiment, we varied the contexts with respect to referential ambiguity of the direct object. All these experiments (with the exception of the one with a segment-by-segment presentation) came up with the same pattern of results:

- In verb-second sentences, there is a slight preference to attach the ambiguous phrase to the VP, which is modulated by lexical features of the verb.
- In verb-final sentences, however, there is a preference to attach the ambiguous PP or NP to the preceding NP initially.

These results have shown to be inconsistent with one of the most discussed models of human sentence processing, the garden-path model (Frazier & Rayner, 1982; Frazier, 1987; 1990) and its successor, construal theory (Gilboy, Sopena, Frazier, & Clifton, 1995; Frazier & Clifton, forthcoming), which both incorporate minimal attachment (MA, 2) and late closure (LC, 3) as their core principles of syntactic processing.

According to construal theory, phrases such as the PP in (5) are preferentially attached as arguments, leading to so-called "primary relations". In these cases, MA predicts a preference for VP-attachment, if the NP-attachment reading is considered more complex, otherwise a preference for NP-attachment is predicted by LC . However, attachment preferences should not differ in either case, when the verb-placement is varied.

As an alternative, which is more consistent with the data, we proposed the *parametrized head attachment model*[3] (7). This model (Konieczny, Scheepers, Hemforth, & Strube, 1994) which assumes human parsing to be directed by the parametrized head attachment principle (PHA), a principle which is mainly based on the availability of lexical heads during processing, and their respective lexical properties, in order to ensure an immediate semantic evaluation of proposed syntactic structures.

(7) *parametrized-head attachment, PHA (Konieczny, Hemforth, Scheepers, & Strube, 1994): (Attempt to) apply head attachment (a) before preferred-role attachment (b) before most-recent head attachment (c).*

a. *head attachment*
Prefer to attach an item to a phrasal unit whose lexical head has already been read.

b. *preferred-role attachment*
Prefer to attach an item to a phrasal unit whose head preferentially subategorizes for it.

c. *most-recent head attachment*
Prefer to attach an item to the head that was read most recently.

PHA predicts differing preferences for verb-second and verb-final structures: in verb-second structures the PP should be attached according to the lexical features of the verb. In verb-final clauses, however there should be a preference to attach the ambiguous phrase to the preceding NP, because its lexical head has already been read whereas the head of the VP is not yet available. Most recent head attachment has been proposed as a structure independent equivalent to late closure (3), to provide an explanation for the garden-path effect which has been found in sentences like (8), among many others.

(8) *Tom said Bill will die yesterday.*

In the following sections, the universal validity of most recent head attachment or late closure, respectively, will be discussed in more detail.

12.3 A crosslinguistic comparison of modifier attachment

12.3.1 Relative clauses

Crosslinguistic studies on modifier attachment have become a central topic in psycholinguistic work over the past few years. The main reason for this fact is that the

[2]PHA is the core principle of the SOUL-(Semantics-Oriented Unification-based Language)-Processor that has been implemented by Lars Konieczny (Konieczny & Strube, submitted; Konieczny, in prep).

universality of parsing principles can be put to test most convincingly by comparing similar structures in as many different languages as possible. This might seem trivial, but it has proven to be rather difficult to find comparable structures in different languages. Most local ambiguities, like e.g. the main verb vs. reduced relative ambiguity in (1) are completely specific for a small class of languages. However, there is a local ambiguity that can be found in many languages as different as German, English, French, Spanish, Italian, Japanese, Finnish, Hungarian, among many others. This ambiguity can be seen in (9): in this kind of structure, the relative clause (*who was on the balcony*) can either be attached "high" to the head of the preceding complex NP (*the servant who was.*) or "low" to the noun within the modifying PP (*the actress who was .*).

(9) *Someone shot the servant of the actress who was on the balcony.*

According to most recent head attachment, attachment to the "lower" head *actress* should be preferred initially, because it is the most recent one. Actually, that is what has been found in some English questionnaire studies. The problem with these structures is that this preference does not hold for all the languages in the world. In one of the first papers that was published on this topic, Cuetos and Mitchell (1988) showed that subjects prefer high attachment of the relative clause (la criada que estaba.) in the Spanish equivalent of (9).

(10) *Alguien disparó la criada del actor que estaba en el balcón.*

If parsing principles like those mentioned above really are process generated and, by consequence, universal, language specific differences like those which have been established for relative clauses cannot be explained without problems.

12.3.2 The modifier straddling hypothesis

One explanation that has been offered by Cuetos and Mitchell (1988) is the *modifier straddling hypothesis* (MSH). They assume that in some cases frequency based preferences may override universal principles. In Spanish, it might be the case that NP-NP-RC sequences fall into one category with other NP-modifier-RC sequences. In this category, the most frequent structure would be NP-adjective-RC, for which only relative clause attachment to the head noun is allowed. Since "high" attachment is necessarily the most frequent attachment in the category, a perhaps still universal preference for low attachment may be overridden in this particular case, though still applicable to other structures.
The predictions which can be derived from the MSH are clear-cut: high attachment preferences should be found for languages with postnominal adjectives, low attachment preferences for those with prenominal adjectives. In a number of studies, it was possible to show that this prediction is wrong. In German (11; Hemforth, Konieczny, & Scheepers, 1994) as well as Dutch (12; Brysbaert & Mitchell, submitted) (both prenominal adjective languages) a clear preference for high attachment of relative clauses could be established.

(11) *Jemand erschoß die Dienerin der Schauspielerin, die auf dem Balkon war.*

(12) *Iemand shot op de knecht van de actrice die op het balkon sat.*

12.3.3 The refined garden-path theory

The earliest studies on relative clause attachment used only questionnaire data to investigate the preferences in different languages. So it could easily have been the case that the preferences that were found in these studies do not reflect the first analysis that is pursued, but might be the result of more elaborate pragmatic processing. This idea was propagated by Frazier (1990) and de Vincenci and Job (in press). According to what has been called the refined garden-path theory (Mitchell, 1994), relative clauses are first attached low following the universal principles. Later on, a pragmatic reevaluation based on a principle of relativized relevance (13) may result in a final preference for high attachment.

(13) *Principle of relativized relevance*
Other things being equal (e.g., all interpretations are grammatical, informative, and appropriate to discourse), preferentially construe a phrase as being relevant to the main assertion of the current sentence. (Frazier, 1990, p.321).

A number of self-paced reading studies in Spanish did not reveal any early low attachment effects.[3] So, a different explanation had to be provided from the universalists camp.

12.3.4 Construal theory

According to Construal Theory (Gilboy et al., 1995), language specific differences are due to the fact that process generated principles like late closure only apply to primary relations (roughly argument or thematically licensed relations). Phrases which have to be modifiers like relative clauses are not directly attached to the phrase marker of the sentence in the first go but only construed as part of a thematic domain. Attachment preferences then follow from a complex set of pragmatic (13) and thematic constraints. Crosslinguistic differences are explained by the interaction of pragmatic principles. The Gricean principle of clarity is assumed to lead to a low attachment preference for modifiers in some languages: in English, there is an umambiguous alternative to the constructions which have been discussed so far (14).

(14) *Someone shot the actress's servant who was on the balcony.*

In sentences like (14) the relative clause can only be attached to the head noun *servant.* Since this alternative exists, readers should presuppose that the writer would have chosen the unambiguous form if head-noun attachment had been intended. Thus, a preference for low attachment results for structures like (9) that overrides the principle of relativized relevance. In Spanish, no alternative form could have been chosen, so, no preference follows from the principle of clarity. Only the principle of relativized relevance applies, leading to a preference for high attachment.[4]

[3] A study on Italian relative clauses which seemed to provide evidence for early low attachment (de Vincenzi & Job, in press) may have been methodologically flawed because of segmentation artifacts in the presentation of the materials (Brysbaert & Mitchell, submitted).

[4] It seems to be difficult to think of a cognitive parser applying the principle of clarity, because this would mean that alternative constructions to the one that is currently processed would have to be generated on-line.

12.4 Is there an early preference for low attachment?

The main difference between the refined garden path theory and construal theory is the question of whether there is an initial preference for attaching the modifier low. Most of the studies that have been done up to now cannot really answer this question for methodological reasons. Firstly, self-paced reading studies may not be sensitive enough for subtle first analysis effects. Secondly, in Spanish as well as in English it is not possible to tap into the disambiguation process really early, because, unlike in German, relative pronouns cannot provide unambiguous information as to which of the possible hosts is intended to be modified. If disambiguating information is provided later in the sentence as in (15), an early low attachment effect may long have disappeared.

(15) *Alguien disparó el criado del actor que estaba en el balcón con su marido.*
Someone shot the (male) servant of the actress who was on the balcony with her husband.

In German, we have the opportunity to tap into early attachment processes as well as late ones. In an eye movement experiment, we investigated sentences like (16a-d), in which we introduced either an early disambiguation of the structurally ambiguous relative clauses by the gender of the relative pronoun (16c,d) or a late disambiguation by number agreement of the potential host and the finite verb of the relative clause (see table 1).

(16) a. *Klaus traf die Lehrerin der Töchter, die in Deutschland lebten, und freute sich sie wiederzusehen.*
Klaus met the teacher of the daughters who lived (plural) in Gemany, and was glad to see her again.

b. *Klaus traf die Lehrerin der Töchter, die in Deutschland lebte, und freute sich sie wiederzusehen.*
Klaus met the teacher of the daughters who lived (sing) in Gemany, and was glad to see her again.

c. *Klaus traf den Lehrer der Tochter, die in Deutschland lebte, und freute sich ihn wiederzusehen.*
Klaus met the teacher (masc) of the daughter (fem) who (fem) lived in Gemany, and was glad to see her again.

d. *Klaus traf den Lehrer der Tochter, der in Deutschland lebte, und freute sich ihn wiederzusehen.*
Klaus met the teacher (masc) of the daughter (fem) who (masc) lived in Gemany, and was glad to see her again.

Methods. Eye-movements were recorded in order to provide an on-line measure of processing complexity due to the predicted preferences. For the statistical analyses, the data were summarized for two regions, firstly, the word preceding the relative pronoun and the relative pronoun proper[5] and secondly, the rest of the relative clause,

[5] When the word preceding the relative pronoun is read, the pronoun is wihin the perceptual span in many cases; i.e. it can already be perceived while the preceding word is fixated (e.g. Rayner & Pollatsek, 1989).

	Number (late)	Gender (early)
high	5b	5d
low	5a	5c

Table 12.1: Design

yielding two dependent variables, namely first pass reading times, and regression path durations, RPDs, (Konieczny, Hemforth & Völker, 1994; Koneczny in prep., see also Liversedge, 1994; Brysbaert & Mitchell, submitted).
First pass reading times (FPRTs) were computed as the sum of all fixations on a region, that a reader looks at for the first time, before moving on to another word/region. FPRTs reflect the processing load at a particular region, if the eyes do not move to another region until all processes that were initialized at this region have come to an end here. If, however, an item within a region initiates a reanalysis, that results in a regressive saccade to a previous region, the duration of the reanalysis is not mirrored in the first pass reading time. We therefore calculated regression path durations, RPDs, as an alternative measure of the total effort of a reanalysis. RPDs were calculated as the sum of the first pass reading time of the region and the duration of all fixations following an immediate regression from that region, until the region is reached again or skipped.

Hypotheses. If relative clauses are initially attached low, reading times should increase when gender information forces high attachment. Because of the questionnaire data mentioned above, we expect a late preference for high attachment, i.e. increased reading times, when low attachment is forced by number agreement.

Results. For the sake of readability, no detailed statistics will be given (but see Hemforth, Konieczny, & Scheepers, in prep). All effects that will be discussed are statistically reliable unless indicated otherwise.

FPRTs for the first region (including the relative pronoun) are reliably longer when high attachment is forced by the relative pronoun (see figure 1). No reliable differences could be established for the second region. If regression times are included as for regression path durations, however, the picture changes. No preference can be found for the first region, but in the second region, regression path durations are reliably longer, when the relative clause has to be attached low (see figure 2).
If attachment of the relative clause is disambiguated late by number agreement, no reliable preferences show up in the FPRTs (see figure 3). Regression path durations, however, are reliably longer in region 2, if low attachment is forced (see figure 4). No reliable differences could be obtained in region 1.

Discussion. The picture that arises from the eye movement data is still not as clear as it could be. When looking at the FPRTs, there appears to be an initial low attachment preference as predicted by the refined garden path theory. But what really happens here is that in those cases when low attachment is forced by the relative pronoun, subjects refixated earlier passages of the sentence more often and/or longer. The total amount of time used to integrate the region including the relative pronoun does not

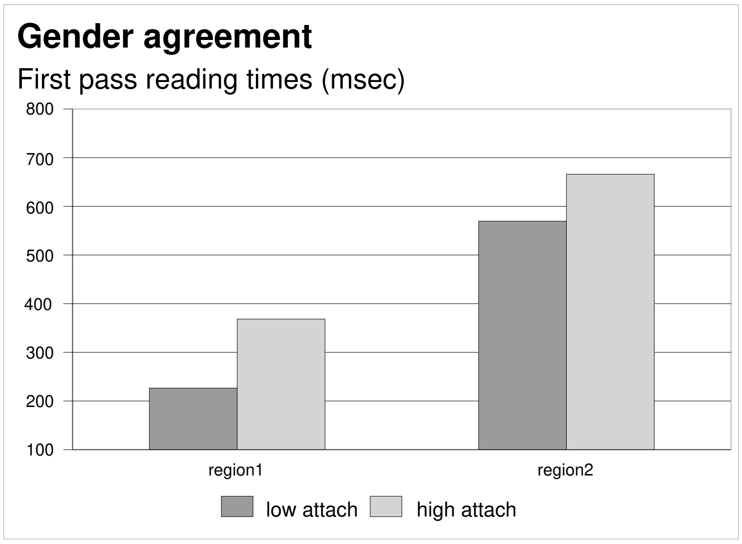


Figure 12.1: Gender-agreement (FPRT)

differ between the two conditions, as reflected in the regression path durations. This is what is predicted by construal theory. It will have to be investigated in more detail, however, which underlying processes result in longer fixations on the critical region when high attachment is forced, on the one hand, and more/longer regressions when low attachment is forced, on the other.

At the end of the relative clause, a clear preference for high attachment (i.e. increased regression path durations when low attachment is forced) could be established, which is compatible with the questionnaire data.

The on-line data that have been found for German relative clause attachment can be regarded as validated by a recent study in Dutch. Brysbaert and Mitchell (submitted) ran a very similar experiment on Dutch relative clause attachment with mostly comparable results. They did even find a slight nonsignificant preference for low attachment if the attachment site was disambiguated early by gender information.

12.5 Are modifiers all alike?

The data we have presented up to now are mostly compatible with construal theory. But, of course, construal theory does not only make predictions for relative clauses but for all sorts of attachment ambiguities. In particular, all sorts of nonprimary modifying relations should be handled by the same set of pragmatic principles. To find out whether this prediction holds, we conducted a further questionnaire study to compare relative clauses and prepositional phrases in sentences like (17a,b).

(17) a. *Die Tochter der Lehrerin, die aus Deutschland kam, traf John.*
The daughter of the teacher who came from Germany met John.

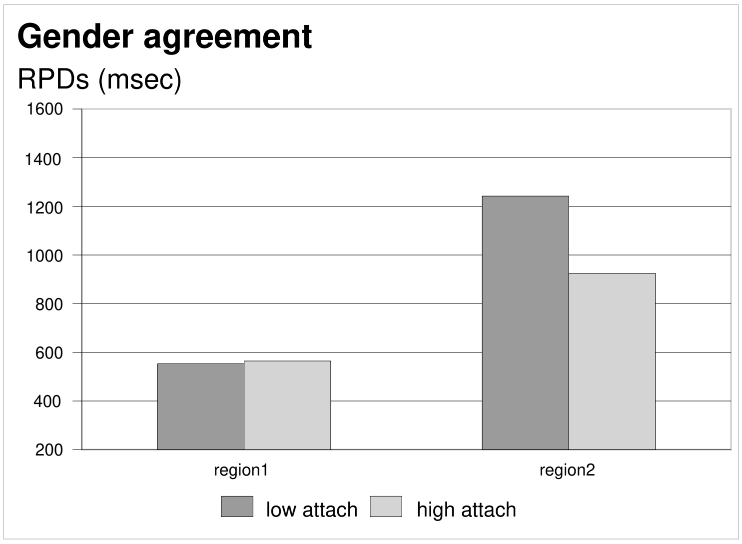


Figure 12.2: Gender-agreement (RPD)

b. *Die Tochter der Lehrerin aus Deutschland traf John.*

The daughter of the teacher from Germany met John.

Since German is a verb-second language the PP in (17b) cannot be analyzed as a verb argument. It has to be a modifier of one of the preceding NPs. So, according to Construal Theory, it should behave similarly to relative clauses in comparable constructions (17a). In our experiments, we found a strong interaction between modifier type (PP vs RC) and attachment preference. Whereas high attachment is preferred for RCs, PPs are preferentially attached low. This finding is supported by the fact that in sentences where high attachment is induced for pragmatic reasons (18a,b; proper nouns tend to refuse modifiers in those constructions) PP-modifiers render the sentences reliably less acceptable than RC-modifiers.

(18) a. *Die Tochter von Mary, die aus Deutschland kam, traf John.*
The daughter of Mary who came from Germany met John.

b. *Die Tochter von Mary aus Deutschland traf John.*
The daughter of Mary from Germany met John.

To account for the possibility that the differences we found for RCs and PPs are due to the fact that in German RCs but not PPs have to be separated by commas, we are currently running an acoustic experiment. In a questionnaire study with Irish students, using a translation of our German sentences without commas, our subjects showed highly comparable preferences to those we found in German.

To conclude, a psycholinguistic theory that does not differentiate between different kind of modifiers as construal theory obviously cannot account for the data. It may be

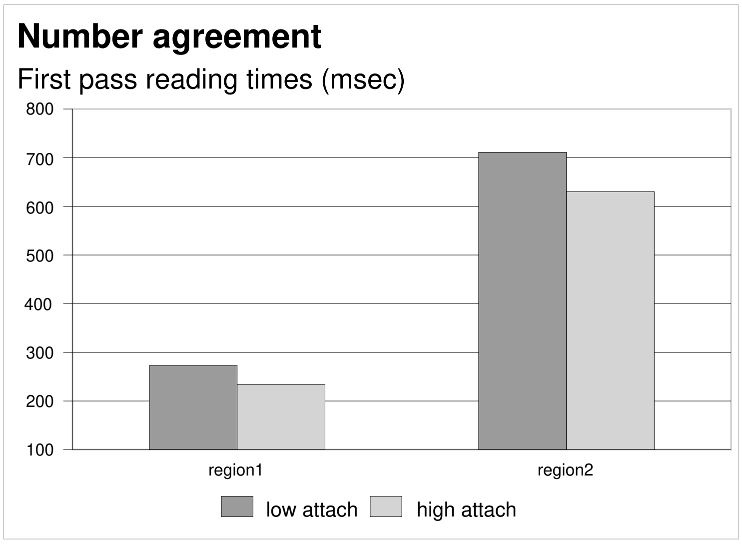


Figure 12.3: Number-agreement (FPRT)

the case that relative clause attachment does not follow the principle of most recent head attachment (or late closure), but prepositional phrase attachment in comparable constructions does.

12.6 Only a question of statistics?

There is an alternative conception to universal approaches that has been proposed in the past few years: the tuning hypothesis (19; Mitchell & Cuetos, 1991). According to Mitchell and Cuetos, the only universal principle is that parsing preferences are exposure-based.

(19) *linguistic tuning hypothesis*
Parsing strategies result from the frequency of success of structural analyses during the history of exposure to linguistic structures.

To predict parsing preferences from the tuning hypothesis, careful corpus analyses have to be carried out. For those rare cases where corpus statistics are available, structures which can be shown to be preferred in on-line parsing usually are more frequent, which at first sight appears to support an exposure-based account on parsing. It could, however, very well be the case that less preferred structures are less frequent in corpus analyses, because they are more difficult to parse.

Without doubt, frequency does play an important role in cognitive parsing, in particular, when lexical access is concerned. It may even play a modulating role in parsing. But even for relative clause attachment, which seems to be very special maybe because of the referential processes involved, parsing preferences are impressingly similar across languages (modulo a somewhat shifting baseline preference; see Gilboy et a. 1995 for

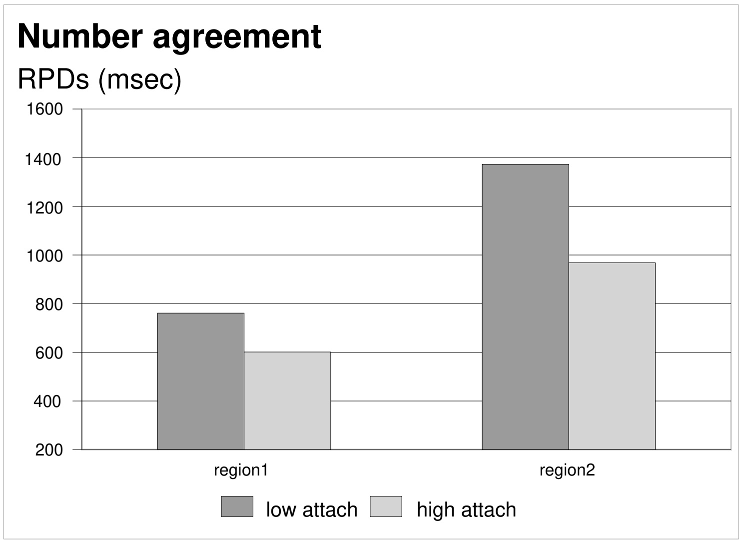


Figure 12.4: Number-agreement (RPD)

a comparison of a variety of structural combinations in English and Spanish). For other structures, similarities are even more impressing. Admittedly, we do not know yet what makes language specific differences in cognitive parsing. But this may only be due to our poverty of imagination. To give up universal principles because of some language specific differences in relative clause attachment seems like throwing out the baby with the bathwater.

References

Abney, S. (1989). A computational model of human parsing. *Journal of Psycholinguistic Research, 18.1,* 129-144.

Brysbaert, M. & Mitchell, D. (submitted). *Modifier attachment in sentence parsing: Evidence from Dutch.*

Chomsky, N. (1986). *Knowledge of language: Its nature, origin, and use.* New York: Praeger.

Crain, S., & Steedman, M. (1985). On not being led up the garden path: The use of context by the psychological parser. In D. R. Dowty, L. Karttunen, & A. Zwicky (Eds.), *Natural Language Parsing: Psychological, Computational, and Theoretical Perspectives* (pp. 320-358). Cambridge: Cambridge University Press.

Cuetos, F., & Mitchell, D. (1988). Cross linguistic differences in parsing: Restrictions on the issue of the Late Closure strategy in Spanish. *Cognition, 30,* 73-105.

De Vincenzi, M. & Job, R. (in press). An investigation of late closure: the role of syntax, thematic structure and pragmatics in initial and final interpretation. *Journal of Experimental Psychology: Learning, Memory, & Cognition.*

Fodor, J.D. & Frazier, L. (1978). The sausage machine: a two stage parsing model. *Cognition, 6,* 291-325.

Frazier, L. (1987). Sentence processing: A tutorial review. In M. Coltheart (Ed.), *The psychology of reading* (pp. 559-586). Hove/London/Hillsdale: Lawrence Erlbaum.

Frazier, L. (1990). Parsing modifiers: Special purpose routines in the human sentence processing mechanism. In Balota, D. A., D'Arcais, G. B. F. & Rayner, K. (Eds.), *Comprehension process in reading* (pp. 303-331). Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Frazier, L., & Rayner, K. (1982). Making and correcting errors during sentence comprehension: eye movements in the analysis of structurally ambiguous sentences. *Cognitive Psychology, 14,* 178-210.

Gilboy, E., Sopena, J., Frazier, L., & Clifton, C. (1995). Argument structure and association preferences in Spanish and English complex NPs. Cognition, 54, 131-167.

Hemforth, B., Konieczny, L., & Scheepers, C. (1994). On reanalysis: Head position and syntactic complexity. In B. Hemforth, L. Konieczny, C. Scheepers & G. Strube (Eds.), *First analysis, reanalysis, and repair* (Vol. IIG-Berichte 8/94, pp. 23-50). Freiburg: Institut für Informatik und Gesellschaft.

Hemforth, B., Konieczny, L. & Scheepers, L. (1994). Probabilistic or universal approaches to sentence processing: How universal is the human language processor? In H. Trost (Ed.), *KONVENS94.* Berlin: Springer.

Kimball, J. (1973). Seven principles of surface structure parsing in natural language. *Cognition, 2,* 15-47.

Konieczny, L. (in prep.). Sentence processing. Doctoral dissertation at the University of Freiburg, Germany.

Konieczny, L., Hemforth, B. & Scheepers, C. (1994). Reanalysis vs. internal repairs: Nonmonotonic processes in sentence perception. In B. Hemforth, L. Konieczny, C. Scheepers & G. Strube (Eds.), *First analysis, reanalysis, and repair* (Vol. IIG-Berichte 8/94, pp. 1-23). Freiburg: Institut für Informatik und Gesellschaft.

Konieczny, L., Hemforth, B. & Völker, N. (1994). The impact of context and semantic bias on constituent attachment in reading. In J. Quantz & B. Schmitz (Eds.), *Desambiguierungsstrategien.* KIT-Reports.

Konieczny, L., Scheepers, C., Hemforth, B. & Strube, G. (1994). Semantikorientierte Syntaxverarbeitung. In S. Felix, C. Habel, & G. Rickheit (Eds.), *Kognitive Linguistik: Repräsentationen und Prozesse.* Opladen: Westdeutscher Verlag.

Liversedge, S. (1994). *Referential contexts, relative clauses, and syntactic parsing.* University of Dundee: Unpublished doctoral thesis.

Mitchell, D. (1994). Sentence Parsing. In M. A. Gernsbacher (Ed.), *Handbook of Psycholinguistics* (pp. 375-410). San Diego: Academic Press.

Mitchell, D. C. & Cuetos, F. (1991). The origins of parsing strategies. In C. Smith (Ed.), *Current Issues in Natural Language Processing.* Austin: Center for Cognitive Science, University of Texas.

Pritchett, B. (1992). *Grammatical competence and parsing performance.* Chicago: University of Chicago Press.

Wanner, E. (1980). The ATN and the sausage machine: Which one is baloney? *Cognition, 8,* 209-225.