gefördert vom
Bundesministerium
für Forschung
und Technologie

Verbmobil
Verbundvorhaben

Integration alternativer Komponenten für die Sprachverarbeitung im VERBMOBIL-Demonstrator

Andreas Turk

DFKI GmbH, Kaiserslautern

Stefan Geißler

IBM Informationssysteme GmbH

Vm **VM Report 67**
April 1995

April 1995

Andreas Turk

DFKI GmbH, Kaiserslautern
Erwin-Schrödinger-Straße (Gebäude 57)
Postfach 2080
67608 Kaiserslautern

Tel.: +49/0 631 205 - 3847
Fax: +49/0 631 205 - 3210
e-mail: turk@dfki.uni-kl.de

Stefan Geißler

IBM Informationssysteme GmbH
Institut für Logik und Linguistik
Vangerowstaße 18
69115 Heidelberg

Tel.: +49/0 6221 59 - 4482
Fax: +49/0 6221 59 - 3200
e-mail: geissler@heidelbg.ibm.com

Gehört zu den Antragsabschnitten: TP 16, TP 6

Die vorliegende Arbeit wurde im Rahmen des Verbundvorhabens VERBMOBIL vom Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie (BMBF) unter dem Förderkennzeichen 413-4001-01 IV 301 4 gefördert. Die Verantwortung für den Inhalt dieser Arbeit liegt bei den Autoren.

Inhaltsverzeichnis

1 Übersicht

VERBMOBIL ist ein Verbundvorhaben, das die Entwicklung eines sprecherunabhängigen Übersetzungssystems für kontinuierlich gesprochene Sprache zum Ziel hat. Die Ausgangssprache ist Deutsch, die Zielsprache Englisch, es besteht eine Beschränkung der Domäne auf Terminabsprachen. Die unterschiedlichen Anforderungen, die ein Übersetzungssystem erfüllen muß, reichen von der Sprachsignalaufbereitung über Spracherkennung, Sprachverarbeitung und Übersetzung bis zur Generierung und Synthetisierung von Sprachausgabe. Als Voraussetzung für die effektive Verarbeitung müssen Daten wie Sprach- und Domänenmodelle, Lexikon und Grammatik existieren. Die Vielzahl der beteiligten Projektpartner stellt z. T. konkurrierende Lösungen für die komplexen Teilaufgaben in VERBMOBIL bereit. Sofern diese Lösungen in Form von *Softwaremodulen* vorliegen, ist es die Aufgabe der Systemintegration (SI) im Teilprojekt 16, diese zu einem Gesamtsystem zu komponieren, das sich dem Benutzer über eine einfache Schnittstelle adäquat präsentiert. Die den Modulen zugrundeliegenden Verarbeitungsstrategien sollen auch in diesem resultierenden Gesamtsystem nachvollziehbar sein.

Die Ziele der SI sind, neben der Gewährleistung einer allgemeinen Funktionsfähigkeit des Gesamtsystems, besonders *intuitive* und *explizite* Hervorhebung der Systemfunktionalität. Erstere ist nur durch die geeignete Festlegung der Benutzerschnittstelle zu erreichen, letztere durch die Bereitstellung von speziellen Medien zur Visualisierung der während der Verarbeitung anfallenden Daten. Erst auf den zweiten Blick offenbaren sich weitere Optionen, die das System bereitstellen soll: Es wird eine Experimentierumgebung notwendig, die die Evaluierung einzelner Softwaremodule erlaubt. Hierzu gehört die Bewertung konkurrierender Ansätze der Teilverarbeitung ebenso, wie die automatische Datenerhebung zur Rückmeldung von Fehlern, Performanz- und Abdeckungsproblemen an die Modulentwickler. Eine notwendige Voraussetzung ist die Austauschbarkeit einzelner Module unter Beibehaltung des Gesamtsystemverhaltens, wie auch die Ausübung von Kontrolle[1], um Datenfluß und Konfigurationen unkompliziert beeinflussen zu können.

Die Verwendung der unterschiedlichsten Programmiersprachen zur Implementierung der Module stellt eine besondere Herausforderung dar.

Die SI verfolgt die o. g. Ziele mit Hilfe einer Strategie, die auf der Anwendung objektorientierter Techniken des *Softwareengineerings* basiert. Eine von autono-

[1] Kesseler [1994] beschreibt einen Ansatz, der zugunsten eines Modells kooperierender Agenten auf zentrale Kontrolle verzichtet. Die Einführung zentraler Kontrolle ist in diesem Sinne künstlich, und außerhalb der Entwicklung und des Testens sogar unerwünscht.

men, kommunizierenden Modulen geprägte Architektur ist also obligatorisch. Das Kommunikationsmedium kann wiederum einer Manipulation zugänglich gemacht werden.

Ein konkretes Beispiel für die Erfüllbarkeit dieser Anforderungen hat die Systemgruppe den Projektpartnern und Gutachtern am 16. Februar 1994 im Rahmen des Demonstrator-Workshops gegeben. Der präsentierte VERBMOBIL-Demonstrator enthielt zwei alternative Spracherkennermodule, die während der Laufzeit wahlweise aktiviert werden konnten.

Weniger offensichtlich wurde im Rahmen eines *Posters* ebenfalls auf dem Demonstrator-Workshop eine Alternative vorgestellt, in der mehrere Module der Sprachverarbeitung ausgetauscht sind. Während im offiziellen Demonstrator das Modul zur syntaktischen Analyse von der Fa. Siemens gestellt wird, die Konstruktion semantischer Information durch ein Modul der Universität Saarbrücken, beinhaltet der *alternative Demonstrator* eine integrierte Komponente zur syntaktisch-semantischen Analyse der Fa. IBM. Dieser alternative Demonstrator soll im Hinblick auf die genannten Ziele der SI vorgestellt und diskutiert werden.

Zunächst erscheint es notwendig, die Architektur des Demonstrators kurz zu skizzieren (Kap. 2). Besonderer Raum ist dabei der Sprachverarbeitungskomponente MDS[2] vorbehalten, die in direkten Vergleich mit der Modulgruppe *Syntax-Semantik-Transfer* (SST) gesetzt werden soll. Als wesentlich wird sich das verwendete Kommunikationsmedium, das *INTARC Communication Environment* (ICE) erweisen [Amtrup, 1994], insbesondere die Festlegung der Schnittstellen zwischen MDS/SST und den anderen Systemkomponenten. Auch die Kommunikation *innerhalb* der linguistischen Komponenten wird angesprochen (Kap. 3). Einer zusammenfassende Bewertung (Kap. 4) folgt der Blick auf den zukünftigen Weg in VERBMOBIL (Kap. 5).

2 Der VERBMOBIL-Demonstrator

Der offizielle VERBMOBIL-Demonstrator (vgl. Abb. 1 u. [Bub, 1995]) erlaubt drei Modi der Verarbeitung — *Erkennung*, *Keywordspotting* und *Buchstabiererkennung* — von denen nur der erste für den Austausch linguistischer Komponenten relevant ist. Der Fluß der Information führt im Modus *Erkennung* von der Spracheingabe über den jeweils aktiven Erkenner, ggf. über die Prosodieanalyse in das MDS (Syntax, Semantikkonstruktion, Semantikauswertung, Transfer), von

[2] Die Herkunft der Bezeichnung „MDS“ ist Geschichte und mittlerweile irrelevant.

dort in die Generierung und weiter in die englische Synthese. Das MDS interagiert außerdem mit dem Modul Dialog.

Betrachtet man die linguistischen Module als Einheit[3], so ergibt sich eine lineare Verarbeitungfolge. Innerhalb des MDS findet ein Informationsfluß auch in entgegengesetzter Richtung statt — zum Abruf zusätzlicher Teillösungen.

Transparent für den Benutzer existieren zwei weitere, im Schaubild nicht vorhandene Prozesse zur Unterstützung der Integration: der *Testbedmanager*, der die Funktion einer Kommunikationsschaltstelle erfüllt, und der *Visualisierungsmanager*, der an den Modulschnittstellen anfallende Daten abgreift, um sie graphisch aufzubereiten.

Der Rahmen der Integration — Anzahl, Art und Bezeichnung der Module, sowie die Kommunikationswege — ist somit fest vorgegeben, s. a. [Auerswald und Amtrup, 1994]. Für den Austausch von Komponenten ist dies teilweise zu berücksichtigen.

2.1 Das MDS

MDS bezeichnet einen Softwareprozeß innerhalb des Demonstrators, der die fachlichen Module Syntax, Semantikkonstruktion, Semantikauswertung und Transfer umfaßt.

Die syntaktische Analyse erfolgt durch einen Parser der Fa. Siemens, der mit einer *Tree-Unification-Grammar* und einem gegenüber dem VERBMOBIL-Lexikon modifizierten Lexikon arbeitet. Als Eingabe liest dieser Parser Worthypothesengraphen [Nöth und Plannerer, 1993] aus einer Datei, erzeugt wird ein HPSG-*Sign* (*Head Driven Phrase Structure Grammar*, [Pollard und Sag, 1987; 1994]). Falls notwendig, steht zur Disambiguierung Information aus dem Domänenmodell im Modul Semantikauswertung zur Verfügung. Die erzeugte Datenstruktur wird von der nachfolgenden Semantikkonstruktion um Einträge semantischer Information ergänzt (Schnittstellendefinition in [Mastenbroek und McGlashan, 1994]). Dabei wird das Verarbeitungsmodell CUF (*Comprehensive Unification Formalism*, [Dörre *et al.*, 1993]) eingesetzt. Eine kompositionale semantische Theorie, die *Diskursrepräsentationstheorie* (DRT), ist mit Hilfe des CUF-Formalismus codiert. Auch die Semantikkonstruktion greift bei Bedarf auf das Domänenmodell im Modul Semantikauswertung[4] zu und fordert von hier den Sprechhandlungstyp

[3]Diese Betrachtungsweise ist durch die Softwarearchitektur motiviert, in der das MDS als ein einziger PROLOG-Prozess realisiert wurde.

[4]Alternativ existiert eine in CUF codierte Version des Domänenmodells, die eine geringere

an, den der Dialog je nach Betriebsmodus bereitstellt [Maier und McGlashan, 1994]. Dazu ist eine Konvertierung der Daten in ein sog. *Information Profile* erforderlich. Beim *Information Profile* handelt es sich um die Zusammenfassung informeller Schnittstellen zwischen CUF- und Nicht-CUF-Prozessen.

Resultat der Semantikkonstruktion ist ein weiter spezifiziertes HPSG-*Sign* (vgl. [Bos *et al.*, 1994] u. [Bos und McGlashan, 1994]), das direkt von der Transferkomponente verarbeitet werden kann. Im Idealfalle werden die Verarbeitungsschritte Semantikkonstruktion und Transfer in einem CUF-Beweis integriert, um Suchräume frühzeitig beschränken zu können. Die Verfügbarkeit von Zwischenergebnissen kann so jedoch nicht gewährleistet werden. Das Ergebnis des Transfers — wiederum ein HPSG-*Sign*, dessen semantische Informationen jedoch auf das Englische bezogen sind — muß vor der weiteren Verarbeitung durch die englische Generierung noch der Konvertierung in ein weiteres *Information Profile* unterzogen werden. Die resultierende semantische Struktur ist flacher und kompakter als ihr HPSG-Pendant, was auch ihrer Kommunizierbarkeit zugute kommt.

2.2 Die Modulgruppe Syntax-Semantik-Transfer

IBM-Demonstratormodul (IBM-DM) ist die Bezeichnung für die integrierte syntaktisch-semantische Analyse, die die Fa. IBM in Form eines PROLOG-Prozesses bereitstellt. Sie enthält — in dem auf CUF basierenden linguistischen Verarbeitungsmodell STUF III codiert — eine HPSG und die *Minimal Recursion Semantics*[5] (MRS), sowie einen *Chartparser* als externe Inferenzkomponente. Details zur Verarbeitung innerhalb dieses Moduls finden sich in Anhang A.

Der Integration des MDS in den VERBMOBIL-Demonstrator ist in der Projektierungsphase Vorrang eingeräumt worden, da das MDS gegenwärtig in den Punkten Stabilität, Performanz und Abdeckung konkurrenzlos ist. So ergab sich die Notwendigkeit, o. g. Rahmenbedingungen für die Integration der das IBM-DM enthaltenden Modulgruppe Syntax-Semantik-Transfer (SST) zu berücksichtigen, um das Ziel äquivalenten Teilsystemverhaltens als Basis für die gewünschte Austauschbarkeit zu erreichen.

Andererseits ist es langfristig erwünscht, mit der Softwarearchitektur die fachliche Architektur in VERBMOBIL zu reflektieren.[6] Erscheint dies nicht möglich, sollte

Zugriffszeit ermöglichen soll.

[5] Eine Einführung in MRS geben Copestake *et al.* [1994]. Hier werden auch Unterschiede zur Semantikkonstruktion des MDS deutlich.

[6] Angestrebte ist ein isomorphe Relation zwischen fachlichen Modulen und Softwaremodulen: jedes fachliche Modul wird durch genau ein Softwaremodul repräsentiert, das jedoch aus meh-

die fachliche Architektur überdacht werden [Görz und Menzel, 1994].

Bedingungen und Ziele für die Integration der SST lassen sich nun zusammenfassen:

1. Die Architektur des MDS ist zu beachten.
2. Die äußeren Schnittstellen des MDS sind einzuhalten.
3. Es sollen vier eigenständige Prozesse existieren: Syntax, Semantikkonstruktion, Semantikauswertung und Transfer.

Offensichtlich ist es notwendig, das MDS aufzubrechen, um die eigenständige Realisierung der Module Transfer und Semantikauswertung zu erzwingen (Abb. 2). Auszutauschen ist die Komponente Semantikkonstruktion des MDS. Sie wird durch das IBM-DM ersetzt. Das Modul Syntax in der SST erfährt eine Sonderrolle: zur Vorbereitung der Integration inkrementell kommunizierender Erkenner bereitet es den in einer Datei abgelegten Worthypothesengraphen für die Übertragung durch ICE auf und sendet die einzelnen Worthypothesen an das Modul Semantikkonstruktion, das die *inkrementelle* Verarbeitung durch den Parser des IBM-DM erlaubt.

Der Informationsfluß innerhalb der SST ist dem innerhalb des MDS ähnlich. Informationen, die *top-down* kommuniziert werden, dienen lediglich dem Abruf zusätzlicher Lösungen. An der Schnittstelle zwischen den Modulen Syntax und Semantikkonstruktion wird der aufbereitete Worthypothesengraph übergeben, Resultat der syntaktisch-semantischen Analyse ist ein HPSG-*Sign*. Die Angleichung der Datenformate an die Erfordernisse des Transfers ist für die Weiterverarbeitung vorgesehen, ebenso die Auflösung von Ambiguitäten und die Festlegung des Sprechhandlungstyps durch die Semantikauswertung. Interaktionen mit Testbed- und Visualisierungsmanager finden statt. Da die Module durch getrennte Prozesse realisiert wurden, ist eine inkrementelle Verarbeitung unter Ausnutzung von Nebenläufigkeiten vorbereitet.

3 Kommunikation

Die Kommunikationsmechanismen, die Inhalte der Kommunikation und die Festlegung der Kommunikationschnittstellen haben sich für die Integration als we-

reren Prozessen bestehen kann. Relationen (z. B. in der Kommunikation) zwischen fachlichen Modulen müssen auf der Implementierungsebene beachtet werden.

sentlich herausgestellt. Ein Vergleich der konkurrierenden Ansätze MDS und SST muß also hier ansetzen.

Entgegen früher verfolgter Bestrebungen, die einzelnen Softwaremodule als Subprozesse eines sog. *Testbeds* zu starten und so die Variablenübergabemechanismen der Implementierungssprache C++ als Kommunikationsmedium zu verwenden, sollte Kommunikation im Demonstrator durch einen flexiblen Mechanismus auf Betriebssystemebene erfolgen. Wichtige Eigenschaften waren: Verfügbarkeit zu Projektbeginn, Konfigurierbarkeit und Adaptierbarkeit. Die Kriterien wurden durch das *INTARC Communication Environment* (ICE) des TP 15.4 erfüllt [Amtrup, 1994]. Durch die hervorragende Kooperation mit dem Entwickler von ICE war eine Anpassung und Erweiterung nach den Vorstellungen der Systemgruppe innerhalb kürzester Frist möglich. ICE basiert auf dem Kommunikationspaket *Parallel Virtual Machine* (PVM), welches eine stetige Aktualisierung und Verbesserung erfährt und mittlerweile zum Standard für Interprozeßkommunikation avanciert ist [Geist *et al.*, 1994].

Die Systemgruppe hat die Adressen und Kommunikationswege für den Demonstrator, wie auch die Formate der zu übertragenden Nachrichten festgelegt [Auerswald und Amtrup, 1994]. Diese Angaben mußten im Verlauf der Integrationsarbeiten weiter verfeinert werden, eine detaillierte allgemeine Beschreibung enthält [Bub, 1995].

Adressen, die in der Kommunikation mit den sprachverarbeitenden Modulen auftreten, sind:

Adresse	Modul
TESTBED	Testbedmanager
VIM	Visualisierungsmanager
DIALOG	Dialogmodul
ERKENNER_UKA	Erkenner der Universität Karlsruhe
ERKENNER_DB	Erkenner der Fa. Daimler Benz AG
PROSODIE	Prosodiemodul
SYNTHESEDT	deutsche Synthese
GENERIERUNGENGL	englische Generierung

Linguistische Module haben die Adressen:

Adresse	Modul
SYNTAX	MDS: syntaktische Analyse
	SST: Aufbereitung des Worthypothesengraphen
SEMKON	MDS: Semantikkonstruktion
	SST: syntaktisch-semantische Analyse
SEMAUS	Semantikauswertung
TRANSFER	ebd.

Die Vernetzung kann Abb. 2 entnommen werden.

3.1 Äußere Schnittstellen

Entscheidend für die Auslösung der Modulgruppe MDS aus dem Demonstrator und deren Ersetzung durch die Modulgruppe SST sind naturgemäß die Schnittstellen, die den Rand dieses Teilsystems definieren. Sie gilt es unverändert beizubehalten. Im einzelnen sind dies die in Tab. 1 aufgelisteten Verbindungen.

Vorläufigen Charakter haben die Verbindungen zum Visualisierungsmanager. Dieser soll in Zukunft nicht direkt adressiert werden.

3.1.1 Nachrichtenformate

Nachrichten, die über ICE zu übertragen sind, sollten nur in Form nullterminierter Zeichenketten (C-Format) auftreten. Dazu waren z. T. Konvertierungen erforderlich (s. Kap. 3.3). Allgemein wurde das Datenformat

```
"<Kommando>␣{<Parameter>}"
```

bzw. für die Kommunikation mit Testbedmanager, Visualisierungsmanager und Dialog

```
"<Absender>␣<Kommando>␣{<Parameter>}"
```

vereinbart. Alle Module sollten in der Lage sein die folgenden Nachrichten des Testbedmanagers zu verarbeiten:

```
"exit"
"reset"
"guion"
"guioff"
```

Verbindung	übertragene Daten
TESTBED ⟶ alle Module	Ein-/Ausschalten der internen Visualisierung Zurücksetzen, Beenden
TESTBED ⟶ SYNTAX	Datenquelle (Erkenner / Prosodie)
alle Module ⟶ TESTBED	Meldungen (fertig / aktiv / inaktiv) Prozeßfortschrittsinformation Fehlermeldungen zu kontrollierbaren Fehlern
ERKENNER_UKA ⟶ SYNTAX	Dateiname für Worthypothesengraphen
ERKENNER_DB ⟶ SYNTAX	Dateiname für Worthypothesengraphen
PROSODIE ⟶ SYNTAX	Dateiname für Worthypothesengraphen
SYNTAX ⟶ SYNTHESEDT	Wortkette der besten syntaktischen Analyse
SYNTAX ⟶ DIALOG	Scheitern der syntaktischen Analyse
TRANSFER ⟶ GENERIERUNGENGL	semantische Information im *Information Profile*
SEMAUS ⟶ DIALOG	Anforderung des Sprechhandlungstyps
DIALOG ⟶ SEMAUS	Sprechhandlungstyp
SEMKON ⟶ VIM	resultierendes *Sign* im FegramEd-Format
TRANSFER ⟶ VIM	resultierendes *Sign* im FegramEd-Format

Tabelle 1: Äußere Schnittstellen

`exit` veranlaßt das Herunterfahren des gesamten Systems, `reset` die Reinitialisierung des Moduls. Die Nachrichten `guion` / `guioff` bewirken, soweit implementiert, die Aktivierung einer modulinternen Visualisierung (*Graphical User Interface*). Im Gegenzug versorgen die einzelnen Module den Testbedmanager mit Hilfe der Nachrichten

```
"<Absender>␣ready"
"<Absender>␣active"
"<Absender>␣inactive"
"<Absender>␣pfi␣<N>"
```

mit Information über den aktuellen Modulzustand, soweit dies möglich ist. In einzelnen bedeutet die Nachricht `ready`, daß das Modul arbeitsbereit ist (nur nach Systemstart oder Reinitialisierung), `active`, bzw. `inactive` beschreiben den Aktivierungszustand des Moduls, `pfi` (Prozeßfortschrittsinformation) indiziert die Übermittlung des prozentualen Prozeßfortschritts `N`.

Kontrollierbare Fehler können an den Testbedmanager zur Einleitung einer Systemrückfrage (über das Modul Dialog) oder einer verbalisierten Fehlerausgabe gemeldet werden:

```
"<Absender>␣error␣[error|fatal|warning]"
```

Eine Fehlerklassifikation ist damit vorbereitet.

Modulspezifische Nachrichtenformate für den Informationsaustausch an den äußeren Schnittstellen von SST/MDS lassen sich an der bereits beschriebenen Kommunikationsstruktur festmachen:

TESTBED ⟶ SYNTAX

```
"lattice_input␣erkenner_uka"
"lattice_input␣erkenner_db"
"lattice_input␣prosodie"
```

ERKENNER_UKA ⟶ SYNTAX

```
"<Dateiname>"
```

Zeichenkette, die den relativen Dateinamen repräsentiert.

ERKENNER_DB ⟶ SYNTAX

```
"<Dateiname>"
```

PROSODIE ⟶ SYNTAX

```
"<Dateiname>"
```

SYNTAX ⟶ SYNTHESEDT

```
"<Satz>"
```

Zeichenkette, die, getrennt durch Leerzeichen, die Wörter des Satzes enthält, der nach der syntaktisch-semantischen Analyse aus dem Worthypothesengraphen extrahiert und analysiert werden konnte.

SYNTAX ⟶ DIALOG

```
"syntax␣parser_failed_1st"
"syntax␣parser_failed_nth"
```

TRANSFER ⟶ GENERIERUNGENGL

```
"<Information Profile>"
```

Das Format wurde direkt aus dem MDS übernommen. Es unterliegt dort einer kontinuierlichen Anpassung.

SEMAUS ⟶ DIALOG

```
"<Information Profile>"
```

Dieses Format gehört ebenfalls zum *Information Profile*, unterscheidet sich aber wesentlich vom vorstehenden.

DIALOG ⟶ SEMAUS

```
"<Information Profile>"
```

SEMKON ⟶ VIM

```
"<FegramEd Term>"
```

Konvertierung des HPSG-*Signs* in eine Zeichenkette, die als Eingabe für den Editor FEGRAMED dienen kann[7].

TRANSFER ⟶ VIM

```
"<FegramEd Term>"
```

Nicht näher spezifizierte Formate befinden sich in vorläufigem Stadium.

[7]Konverter: K. Czuba, Systemgruppe; FEGRAMED: Bernie u. Tom, DFKI.

Verbindung	übertragene Daten
SYNTAX ⟶ SEMKON	Worthypothesen
SEMKON ⟶ SYNTAX	beste Wortkette des Resultats der syntaktisch-semantischen Analyse
SEMKON ⟶ SEMAUS	zu disambiguierende Struktur im *Information Profile*
SEMAUS ⟶ SEMKON	disambiguierte Struktur im *Information Profile*
SEMKON ⟶ TRANSFER	HPSG-*Sign* als Resultat der syntaktisch-semantischen Analyse
TRANSFER ⟶ SEMKON	Anforderung weiterer Lösungen / Bestätigung der positiven Verarbeitung

Tabelle 2: Innere Schnittstellen

3.2 Innere Schnittstellen

Die inneren Schnittstellen des MDS sind virtuell, da die verschiedenen fachlichen Module in einem Softwareprozeß zusammengefaßt sind. Datentransfer erfolgt somit durch Unifikation auf PROLOG-Ebene. Strebt man im Gegensatz dazu die bereits propagierte Isomorphie zwischen fachlicher Architektur und Softwarearchitektur an, ergibt sich sowohl die Notwendigkeit, Kommunikation explizit zu machen, als auch die Freiheit, sie ohne Restriktionen zu gestalten. Selbstverständlich sollte als Medium ausschließlich ICE zum Einsatz kommen.

Die realisierten Kommunikationswege sind im einzelnen aus Tab. 2 zu ersehen (vgl. auch Abb. 2).

Eine bidirektionale Verbindung TRANSFER ⟷ SEMAUS ist bei Bedarf leicht zu erstellen, z. Z. aber noch nicht notwendig.

3.2.1 Nachrichtenformate

Neben den allgemeinen Nachrichten (s. Kap. 3.1.1) existieren wiederum modulspezifische Nachrichtenformate:

SYNTAX ⟶ SEMKON

```
"<Worthypothesen>"
```

Einträge der vom Erkenner / Prosodie übermittelten Datei zeilenweise, ohne Kommentare, einschließlich `BEGIN_LATTICE` und `END_LATTICE`.

SEMKON ⟶ SYNTAX

`"<Satz>"`

Wie zuvor.

SEMKON ⟶ SEMAUS

`"<Information Profile>"`

SEMAUS ⟶ SEMKON

`"<Information Profile>"`

SEMKON ⟶ TRANSFER

`"<Sign>"`

HPSG-*Sign* im CUF-externen Format.

TRANSFER ⟶ SEMKON

`"no_more_solutions"`
`"next_solution"`

In Abhängigkeit davon, ob der Transfer mit den angebotenen Daten erfolgreich arbeiten konnte.

3.3 Konvertierung

Einzig zugelassener Datentypen für die ICE-Kommunikation war `IDL_String`. Dies entspricht in den unterschiedlichen Programmiersprachen:

Sprache	Datentyp
C/C++	`char*`
LISP	`string`
PROLOG	`atom`
Tcl/Tk	`string`

Verbindung	Konvertierungstyp
syntax ⟶ semkon	coded
syntax ⟶ synthesedt	string
semkon ⟶ syntax	coded
semkon ⟶ transfer	ft
semkon ⟶ semaus	coded
semkon ⟶ vim	string
semaus ⟶ semkon	coded
transfer ⟶ semkon	coded
transfer ⟶ generierungengl	coded
transfer ⟶ vim	string
generierungengl ⟶ synthesedt	string
alle anderen Verbindungen	uncoded

Tabelle 3: Konvertierungen

(vgl. [Auerswald und Amtrup, 1994]). Alle Module innerhalb der SST sind in PROLOG implementiert, die externen Kommunikationspartner in den drei anderen genannten Programmiersprachen. Teilweise sind Daten von komplexerer Struktur (Terme) von PROLOG aus zu übertragen. Eine fallbasierte Konvertierung der ICE-Nachrichten ist also unumgänglich. Die zur Codierung verwendete Tabelle (Tab. 3) enthält vier verschiedene Konvertierungstypen: `coded`, `string`, `ft` und `uncoded`, die auf die *automatische* Konvertierung von Nachrichten vor dem Senden und nach dem Empfangen durch ICE Einfluß nehmen (Tab. 4).

Die Konvertierung erfolgt i. d. R. in den Schritten

1. Schreiben eines Terms in eine Zeichenkette
2. Konvertierung der Zeichenkette in ein Atom

bzw. umgekehrt. Der erste Schritt entfällt für `string`.

Diese Vorgehensweise unterliegt technischen Restriktionen. So kann ein Atom in Quintus-PROLOG 3.2.1 nur die max. Länge von 1 024 Zeichen haben (gilt nicht für die Version 3.2). Die max. Länge einer Zeichenkette ist aufgrund der internen Repräsentation identisch mit der max. Länge einer Liste: 64 KB. Diese Grenze wurde in einigen Testläufen überschritten, so daß die beschriebene Lösung als vorläufig angesehen werden muß.

Konvertierungstyp	Effekt	
`coded`	Senden:	beliebiger Datentyp wird in Atom konvertiert
	Empfangen:	Rekonvertierung des ursprünglichen Datentyps aus dem übertragenen Atom
`string`	Senden:	direkte Konvertierung einer Zeichenkette in ein Atom — effizienter als mit `coded`
	Empfangen:	wie `coded`
`ft`	Senden:	direkte Konvertierung eines HPSG-*Signs* in ein Atom — effizienter als mit `coded`
	Empfangen:	wie `coded`
`uncoded`	es erfolgt keine Konvertierung (nur sinnvoll für Atome in PROLOG)	

Tabelle 4: Auswirkungen der Konvertierung

4 Resümee

Es ist gelungen, den praktischen Nachweis der in Kap. 1 eingeforderten Flexibilität des VERBMOBIL-Demonstrators zu geben, indem der Austausch des Teilsystems der linguistischen Verarbeitung durchgeführt wurde — ein komplexeres Szenario als der Austausch einzelner Module (z. B. des Erkenners).

Die nach obiger Analyse erstellte Implementierung der Modulgruppe *Syntax-Semantik-Tranfer* (SST), die das *IBM-Demonstratormodul* (IBM-DM) enthält, erlaubt die Realisierung des alternativen Demonstrators bei ansonsten gleicher Konfiguration des Gesamtsystems auf einfache Art: statt dem initialen Start des Prozesses MDS sind die SST-Prozesse Syntax, Semantikkonstruktion, Semantikauswertung und Transfer zu starten.

Angesichts der genannten Probleme kann das Erreichte nur als Erfolg und Bestätigung der Strategie der Systemgruppe gewertet werden. Der objektorientierte Integrationsansatz ist durchführbar, eine *Breadboard*-Funktionalität kann erreicht werden.

Die resultierenden Gesamtsysteme sind anhand typischer Eigenschaften der konkurrierenden Teilsysteme MDS und SST vergleichbar: Der augenfälligste Unterschied, die Kapselung der Module in einem Softwareprozess im MDS und die Aufteilung auf vier Softwareprozesse bei der SST bestimmt den Bedarf an inter-

ner Kommunikation. Das MDS wartet so mit einem geringen Ressourcenbedarf auf. Durch eine bestehende Vorabintegration[8] sind Tests leicht durchzuführen. Eventuell vorhandene Nebenläufigkeiten, bspw. in der Konvertierung und Übertragung von Daten zur Visualisierungskomponente, können jedoch nicht ausgenutzt werden. Die internen Schnittstellen sind nur durch Manipulationen am bereits integrierten Teilsystem MDS abzugreifen. Bei der Existenz einer expliziten Kommunikationsschnittstelle ist die kurzfristige Modifikation der Formate zu übertragender Daten, die bei der Integration des Demonstrators zu Problemen geführt hat, kaum möglich.

Die Abbildung der fachlichen Architektur auf die Softwarearchitektur entspricht einer oft geäußerten Forderung. Nachvollziehbar inkrementelles Systemverhalten, Ausnutzung von Nebenläufigkeit und die Flexibilität bei der Systemevaluierung sind vorbereitet. Der Zwang, Schnittstellen zu spezifizieren, führt zur empirischen Validierung der fachlichen Architektur.

Bedingt durch den späten Einstieg in die Entwicklungsarbeit befindet sich das IBM-DM noch in einer experimentellen Phase. So war es bis zum Termin der Demonstratorabnahme nicht gelungen, die Schnittstelle zum Transfer anzupassen, so daß ein voll funktionsfähiges Gesamtsystem nicht vorgeführt werden konnte. Der bislang noch nicht optimierte *Chartparser* hat zudem einen unverhältnismäßigen Ressourcenbedarf.

Eine feinkörnige Integration der Module Syntaxanalyse und Semantikkonstruktion, wie sie durch die Verwendung von HPSG intendiert und im IBM-DM realisiert ist, kann kritisch betrachtet werden (vgl. Kap. 5). Die Verwendung des Basisformalismus STUF III ist innerhalb des Projektes z. T. umstritten.

5 Ausblick

Eine die Systemintegration betreffende Zusammenfassung von Standpunkten zur zukünftigen Entwicklung der Architektur in VERBMOBIL bieten Görz und Menzel [1994]. Die dort aufgestellten Forderungen und Vorschläge haben ihre Realisierung z. T. mit dem Demonstrator, bzw. mit dem alternativen Demonstrator erfahren. Es findet sich ebenso die Anregung, auf die Kommunikation ICE aus TP 15 zurückzugreifen, wie auch die Forderung, auf konzeptionelle Unterschiede zwischen fachlicher Architektur und Systemarchitektur zu verzichten. Besondere Bedeutung wird der Inkrementalität der Verarbeitung zugemessen.

[8] S. McGlashan, Univ. Saarbrücken.

Für ein Übersetzungssystem sollte auch die tiefe inhaltliche Analyse der syntaktischen und semantischen Strukturen hinter einer auf den Transfer ausgerichteten Vorgehensweise zurückstehen (die *Minimal Recursion Semantics* des IBM-Demonstratormoduls erhebt diesen Anspruch). Die bestehende Architektur sollte nicht als fest angesehen, sondern bei Bedarf angepaßt werden.

Darüber hinaus fordern die Autoren eine Verlagerung vom text- zum sprachorientierten Grammatikansatz. Eine Trennung von syntaktischer und semantischer Analyse wird propagiert.

Andere Arbeiten beschäftigen sich mit den Problemen der Kontrolle bei verschiedenen Formen der inkrementellen Verarbeitung [Kesseler, 1994] oder neuartigen Konzepten des Systemverhaltens (*Anytime*-Verhalten, [Görz und Kesseler, 1994]).

Die Diskussionen, die auf den jüngsten Treffen (Arbeitstreffen Syntax-Semantik-Transfer-Generierung am 27.03. in Stuttgart, Workshop INTARC am 05.–07.04. in Hamburg) geführt wurden, verdeutlichen den Bedarf an langfristigen Architekturentscheidungen, besonders im Hinblick auf den Forschungsprototypen. Exemplarisch ist hier der Konsens bei Verwendung flacher semantischer Strukturen, die Frage nach sequentieller oder inkrementell interaktiver Ablaufstrategie oder die Festlegung von Datenformaten zu nennen.

Innerhalb der Systemgruppe gibt es Überlegungen, die Softwareentwicklung unter Einbeziehung der Projektpartner durch geeignete Methoden des *Softwareengineerings* zu standardisieren und zu koordinieren, um von wissenschaftlich-experimentellen Implementierungen zu effizienten, dokumentierten und leicht wartbaren Modulen zu kommen. Gegenwärtig steht eine Version des Demonstrators zur Diskussion, die eine Verteilung der zu integrierenden Module über mehrere Rechner erlaubt. Das sog. *Verteilte VERBMOBIL* soll über Internetverbindungen kommunizieren.

Anh. A diskutiert weitere, speziell das IBM-DM betreffende zukünftige Aufgaben.

Literatur

[Abb und Buschbeck-Wolf, 1995] Bernd Abb und Bianca Buschbeck-Wolf. MINT — Minimal Transfer. VERBMOBIL Report, 1995. im Erscheinen.

[Amtrup, 1994] Jan Willers Amtrup. ICE — INTARC Communication Environment. Users Guide and Reference Manual. VERBMOBIL Technisches Dokument Nr. 14, November 1994. TP 15.4 — Universität Hamburg.

[Auerswald und Amtrup, 1994] Marko Auerswald und Jan Willers Amtrup. Kommunikationsarchitektur für den Demonstrator. VERBMOBIL Technisches Dokument Nr. 15, Dezember 1994. TP 16 — Systemintegration, DFKI GmbH, Kaiserslautern / TP 15.4 — Universität Hamburg.

[Bos *et al.*, 1994] Johan Bos, Elsbeth Mastenbroek, Scott McGlashan, Sebastian Millies und Manfred Pinkal. The VERBMOBIL Semantic Formalism (Version 1.3). VERBMOBIL Report Nr. 6, Januar 1994. TP 8 — Universität des Saarlandes.

[Bos und McGlashan, 1994] Johan Bos und Scott McGlashan. Extensions to the VERBMOBIL Semantic Formalism and Analysis of the Referenzdialog. VERBMOBIL Memo Nr. 43, August 1994. TP 8 — Universität des Saarlandes.

[Bub, 1995] Thomas Bub. Der VERBMOBIL-Demonstrator. VERBMOBIL Report, Juni 1995. TP 16 — Systemintegration, DFKI GmbH, Kaiserslautern.

[Copestake *et al.*, 1994] Ann Copestake, Dan Flickinger, Rob Malouf, Susanne Riehemann und Ivan A. Sag. Transfer and Minimal Recursion Semantics. draft version, Oktober 1994.

[Dörre *et al.*, 1993] Jochen Dörre, Michael Dorna, Suresh Manandhar und Chris Brew. Computational Aspects of Constraint-Based Linguistic Description I. Universität Stuttgart, August 1993.

[Geist *et al.*, 1994] Al Geist, Adam Beguelin, Jack Dongorra, Weicheng Jiang, Robert Manchek und Vaidy Sunderam. *Pvm3 User's Guide and Reference Manual.* Oak Ridge National Laboratory, Oak Ridge, Te., Mai 1994.

[Görz und Kesseler, 1994] Günther Görz und Marcus Kesseler. Anytime Algorithms for Speech Parsing? VERBMOBIL Report Nr. 13, März 1994. TP 15.7 — Friedrich-Alexander-Universität Erlangen-Nürnberg.

[Görz und Menzel, 1994] Günther Görz und Wolfgang Menzel. Diskussionsbeitrag zur Systemarchitektur von VERBMOBIL. VERBMOBIL Memo Nr. 45, August 1994. TP 15 — Friedrich-Alexander-Universität Erlangen-Nürnberg / Universität Hamburg.

[Kesseler, 1994] Marcus Kesseler. Distributed Control in VERBMOBIL. VERBMOBIL Report Nr. 24, August 1994. TP 15.7 — Friedrich-Alexander-Universität Erlangen-Nürnberg.

[Maier und McGlashan, 1994] Elisabeth Maier und Scott McGlashan. Semantic and Dialogue Processing in the VERBMOBIL Spoken Dialogue Translation System. VERBMOBIL Report Nr. 51, September 1994. TP 8 — DFKI GmbH, Saarbrücken / Universität des Saarlandes.

[Mastenbroek und McGlashan, 1994] Elsbeth Mastenbroek und Scott McGlashan. The VERBMOBIL Syntax Semantics Interface — Version 1.2. VERBMOBIL Memo Nr. 41, August 1994. TP 8.1 — Universität des Saarlandes.

[Nöth und Plannerer, 1993] Elmar Nöth und Bernd Plannerer. Schnittstellendefinition für den Worthypothesengraphen. VERBMOBIL Memo Nr. 2, Dezember 1993. TP 3 — Friedrich-Alexander-Universität Erlangen-Nürnberg.

[Pollard und Sag, 1987] Carl Pollard und Ivan A. Sag. *Information Based Syntax and Semantics.* CSLI Lecture Notes No. 13. Center for the Study of Language and Information, Stanford University, 1987.

[Pollard und Sag, 1994] Carl Pollard und Ivan A. Sag. *Head Driven Phrase Structure Grammar.* Center for the Study of Language and Information, Stanford University, 1994.

[Shieber, 1985] Shieber. Using restrictions to extend parsing algorithms for complex feature-based formalisms. In *Proceedings of the 22nd annual meeting of the ACL*, 1985.

[Weber, 1995] Hans Weber. *LR-inkrementelles, probabilistisches Chartparsing von Worthypothesenmengen mit Unifikationsgrammatiken: Eine enge Kopplung von Suche und Analyse.* Dissertation, eingereicht an der Universität Hamburg, Informatik, 1995.

A Chartparsing für HPSG

A.1 Die linguistischen Wissensquellen im IBM-DM

Wie bereits angedeutet faßt das *IBM Demonstratormodul (IBM-DM)* die Bereiche Syntax und Semantikkonstruktion zu einem integrierten Modul zusammen. Für eine solche Integration verschiedener linguistischer Domänen steht mit der HPSG ein Rahmen zur Verfügung, der die Beschreibung grammatikalischen Wissens aus unterschiedlichen Domänen unterstützt und deren Repräsentation in einer formalen Sprache mit wohlbekannten Eigenschaften — den getypten Merkmalsstrukturen — erlaubt. Das Zusammenwirken verschiedener linguistischer „Module" wie Syntax und Semantik[9] stellt sich bei dieser Herangehensweise nicht als das Ergebnis der Kommunikation verschiedener eigenständiger Softwarekomponenten dar, die über definierte Schnittstellen miteinander kommunizieren, sondern als eine Sammlung von Wohlgeformtheitsbeschränkungen über die mögliche Struktur sprachlicher *Zeichen*.

Interaktion zwischen verschiedenen Domänen läßt sich in diesem Ansatz im Zeichen selbst, in Form von *Strukturteilung* und Pfadgleichungen modellieren. Syntaktischer Strukturaufbau erfolgt somit unter gleichzeitiger Beachtung der entsprechenden semantischen Beschränkungen — etwa in Form von Selektionsrestriktionen bei Komplementation oder Modifikation. Ebenso erfordert eine ernsthafte Beschreibung des Zusammenhangs von syntaktischen und semantischen Eigenschaften im Lexikon, dem sog. *Linking* eine uniforme Repräsentation der beteiligten linguistischen Domänen.

Die Implementierung großer Grammatiken in diesem durch die HPSG unterstützten und explizit gemachten *constraint-pool* Ansatz profitiert dabei insbesondere von drei Eigenschaften der gewählten Repräsentationssprache, den getypten Merkmalsstrukturen:

- Eine rein deklarative Modellierung erlaubt es, von der konkreten Verarbeitung zu abstrahieren. In welcher Reihenfolge Beschränkungen aus dem *constraint-pool* angewendet werden, hat keinen Einfluß auf die durch eine Grammatik lizensierten Zeichen.
- Durch strikte Monotonie ist sichergestellt, daß die Hinzunahme beliebiger Information in Form weiterer Beschränkungen niemals eine als nicht-wohlgeformt gekennzeichnete Struktur wieder wohlgeformt wird.

[9] und evtl. Pragmatik, Morphologie und weiterer Domänen

- Ein mächtiges Typenkonzept erlaubt multiple Vererbung und Unterspezifikation.

Ein Repräsentationsformalismus mit den genannten Eigenschaften ist STUF-III. Die Spezifikation der linguistischen Wissensquellen im IBM-DM, der Syntax, der kompositionellen Semantik, des Lexikons und der Linking-Bedingungen, fand deshalb ausschließlich in STUF statt.

Während somit eine verteilte, abstrakte und linguistisch anspruchsvolle Implementierung unterstützt wird, ergeben sich hinsichtlich der Notwendigkeit einer effizienten Verarbeitung Probleme: Eine hinreichende Effizienz im Paradigma *Parsing as Deduction* erscheint mit den derzeit verfügbaren Mitteln nicht möglich, bzw. würde eine massive Annotierung des deklarativen Teils der Grammatik mit rein verarbeitungsorientierten Konstrukten erfordern, was auf Kosten der Wartbarkeit und Transparenz ginge.

Der im IBM-DM eingeschlagene Weg basiert daher auf einer Abbildung der für die Spezifikation gewählten Strukturen auf ein Compilat, für das eine effizientere Verarbeitung ermöglicht werden kann. Das Verarbeitungsmodul stellt dabei der von TP 15 (FAU Erlangen-Nürnberg) entwickelte Chart-Parser dar[10], die Abbildung der HPSG-Grammatik erfolgt demzufolge in eine kontext-freie Grammatik. Abbildung und gewähltes Format werden in Abschnitt A.2 detaillierter beschrieben.

Das resultierende Modul IBM-DM weist damit zwei externe Schnittstellen auf: Auf der *Speech*seite erwartet der Parser eine Lattice gemäß [Nöth und Plannerer, 1993], während nach oben HPSG-Zeichen mit den in [Abb und Buschbeck-Wolf, 1995] beschriebenen Eigenschaften in den Transfer einfließen.

A.2 Die Vorverarbeitung der IBM-Grammatik

Grundvoraussetzung für eine effiziente Verarbeitung einer großen HPSG ist, daß die sehr „teuren“ Regelanwendungen so selten wie möglich, d.h. nur in plausiblen Fällen stattfinden. Einer jeden solche Regelanwendung ist daher ein „billiger“ Test voranzustellen, der die Anwendung der aufwendigeren Operationen aufschiebt. Ein solcher Test erfolgt mit einem Teil der in den beteiligten Strukturen kodierten Information, der sog. *restriction* ([Shieber, 1985]). Im IBM-DM nimmt diese Restriktion die Form einer kontext-freien Grammatik an, die aus der HPSG compiliert wird. Dieses Vorgehen erlaubt zum einen, eine lediglich schwache Äqui-

[10] vgl. [Weber, 1995].

valenz zwischen der durch die Restriktion beschriebenen Sprache und der vollen HPSG zu fordern, da jede durch die Restriktion lizensierte Struktur nachträglich gegen die gesamten in der Grammatik kodierten Beschränkungen getestet wird. Zum anderen besteht die Möglichkeit, den Zeitpunkt zur Durchführung des zweiten Tests frei zu wählen. Mögliche Vorgehensweisen hierbei können ein sofortiges Durchlösen der Strukturen nach einem Erfolg der Restriktion sein, oder ein Aufschieben bis nach einem erfolgreichen Strukturaufbau über die gesamte Äußerung ebenso wie ein Durchlösen nach Aufbau einer definierten Teilkonstituente.

Sowohl hinsichtlich dieser Frage, als auch bezüglich der in die Restriktion aufzunehmenden Informationen befindet sich das IBM-DM derzeit noch im Stadium des Experimentierens.

Gegenwärtig besteht die Restriktion aus den syntaktischen Head-Werten, sowie denen der Valenz- und Modifikationsmerkmalen und den Merkmalen zur Kodierung nicht-lokalen Abhängigkeiten. Letzteres ermöglicht eine sehr restriktive Behandlung leerer Elemente, da bereits die in der Restriktion kodierte Information Aussagen darüber macht ob, und wenn ja, mit welcher Art Spur ein Zeichen kombiniert werden kann.

Jedes dieser Merkmale wird als Vektor kodiert, dessen bits jeweils eine von elf paarweise disjunkten Klassen von syntaktischen Head-Typen repräsentieren und in welchem sich Kompatibilität, bzw. Nichtkompatibilität mit dieser Klasse ausdrücken.

```
Beispiel fuer einen Head-Vektor in der Restriktion:
(no,no,no,no,X,X,no,no,no,no,no)
```

In dem obigen Beispiel wird Kompatibilität mit lediglich zweien der elf Klassen kodiert (die bits 5 und 6 beispielsweise stehen für finite bzw. infinite Verben). Auf diese Art erlaubt die Kodierung der Restriktion Unterspezifikation — die Anzahl der kontext-freien Regeln wäre ohne diese Möglichkeit unhandhabbar groß.

Diese Art der Kodierung wird nun an zwei unterschiedlichen Stellen angewendet:

- Zum einen werden die durch die Grammatik lizensierten lokalen Bäume als Restriktion kodiert. Die Anzahl der in diesem Schritt gebildeten Bäume bewegt sich in der Größenordnung 50 - 60. Jeder dieser Bäume wird als Regel interpretiert. Diese Regeln stellen wie oben erwähnt detailierte Anforderungen an Kategorie, Valenz- und Modifikationsinformation der beteiligten Strukturen, sowie hinsichtlich Art und Anzahl der beteiligten Spuren.

Darüberhinaus erlauben sie einen effizienten Subsumtionscheck, der für die Verarbeitung im Chart-Parser von großer Wichtigkeit ist.

- In ähnlicher Weise wird das Lexikon abgebildet auf eine Menge von vollen HPSG-Strukturen und den dazugehörigen Restriktionen. Da sich in HPSG sehr detailierte Angaben über die Struktur leerer Elemente, die ein lexikalischer Eintrag evtl. lizensiert, machen lassen, wird auch diese Information off-line berechnet und zusammen mit den den Spuren zuzuordnenden Restriktionen im Lexikon kodiert.

A.3 Ausblick

Das beschriebene Vorgehen bietet sich in den folgenden Punkten als Grundlage für weitere Entwicklungen an:

- Die entstehende Grammatik bietet sich für eine Erweiterung um probabilistische Bewertungen der Regelanwendungen an.
- Es besteht die Möglichkeit, daß bereits eine Teilmenge der in der HPSG kodierten Beschränkungen ausreichen, um auf den durch die Restriktion lizensierten Bäumen die für die Weiterverarbeitung im Transfer notwendigen Strukturen zu erzeugen.
- Des weiteren ist zu ermitteln, inwieweit sich ähnliche Eigenschaften innerhalb des Paradigmas *Folding/Unfolding* von Merkmalsstrukturen erzielen lassen.
- Schließlich erscheint es als wünschenswert, die Wahl der in der Restriktion enthaltenen Merkmale durch objektiv ermittelbare Kriterien — dem Maß ihrer Restriktivität — zu steuern.

B Abdeckung und Performanz

Eine Abschätzung der Abdeckung und Performanz war für die experimentelle Version des IBM-DM nicht sinnvoll, da diese nicht optimiert war (keine Berücksichtigung von statistischen Einträgen im Worthypothesengraph, keine *Agenda-Pruning*, kein *Time-Mapping*). Der Einsatz unter realen Bedingungen ist erst mit einer überarbeiteten Version, insbesondere mit der neuen Parserkomponente, sinnvoll.

C Abbildungen

Abbildung 1: Die Vernetzung der Module im Demonstrator

Abbildung 2: Die linguistische Komponente im alternativen Demonstrator