Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Document
D-92-28

Ein Modell zur Repräsentation von Nachrichtentypen

Klaus-Peter Gores, Rainer Bleisinger

November 1992

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 20 80
D-6750 Kaiserslautern
Tel.: (+49 631) 205-3211/13
Fax: (+49 631) 205-3210

Stuhlsatzenhausweg 3
D-6600 Saarbrücken 11
Tel.: (+49 681) 302-5252
Fax: (+49 681) 302-5341

Deutsches Forschungszentrum für Künstliche Intelligenz

The German Research Center for Artificial Intelligence (Deutsches Forschungszentrum für Künstliche Intelligenz, DFKI) with sites in Kaiserslautern and Saarbrücken is a non-profit organization which was founded in 1988. The shareholder companies are Atlas Elektronik, Daimler Benz, Fraunhofer Gesellschaft, GMD, IBM, Insiders, Mannesmann-Kienzle, Philips, SEMA Group Systems, Siemens and Siemens-Nixdorf. Research projects conducted at the DFKI are funded by the German Ministry for Research and Technology, by the shareholder companies, or by other industrial contracts.

The DFKI conducts application-oriented basic research in the field of artificial intelligence and other related subfields of computer science. The overall goal is to construct *systems with technical knowledge and common sense* which - by using AI methods - implement a problem solution for a selected application area. Currently, there are the following research areas at the DFKI:

- Intelligent Engineering Systems
- Intelligent User Interfaces
- Intelligent Communication Networks
- Intelligent Cooperative Systems.

The DFKI strives at making its research results available to the scientific community. There exist many contacts to domestic and foreign research institutions, both in academy and industry. The DFKI hosts technology transfer workshops for shareholders and other interested groups in order to inform about the current state of research.

From its beginning, the DFKI has provided an attractive working environment for AI researchers from Germany and from all over the world. The goal is to have a staff of about 100 researchers at the end of the building-up phase.

Prof. Dr. Gerhard Barth
Director

Ein Modell zur Repräsentation von Nachrichtentypen

Klaus-Peter Gores, Rainer Bleisinger

DFKI-D-92-28

Diese Arbeit wurde finanziell unterstützt durch das Bundesministerium für Forschung und Technologie (FKZ ITW-9003 0).

Ein Modell zur Repräsentation von Nachrichtentypen

Klaus-Peter Gores & Rainer Bleisinger

Zusammenfassung

In diesem Papier stellen wir einen Formalismus vor, mit dem eine computergerechte Repräsentation verschiedener Klassen von Geschäftsbriefen möglich ist. Der Ausgangspunkt dieser Bemühungen ist das Projekt ALV (Automatisches Lesen und Verstehen), dessen Ziel das partielle Erkennen einer eingeschränkten Menge von Geschäftsbriefen ist. Für die verschiedenen Klassen von Geschäftsbriefen werden sogenannte Nachrichtentypen entwickelt, die sich aus einzelnen Bausteinen, den Nachrichtenelementen zusammensetzen. Diese werden durch eine modifizierte Conceptual Dependency-Notation definiert. Durch die hierarchische und modulare Definition kann eine breite Anzahl von Geschäftsbriefen modelliert werden. Die Zielrichtung des hier vorgestellten Modells liegt neben der effizienten Modellierung der Nachrichtentypen in der Bearbeitung durch ein Verfahren zur erwartungsgesteuerten Textanalyse in ALV. Um dies zu ermöglichen, wurden zahlreiche Steuerungselemente in das Nachrichtenmodell aufgenommen.

Inhaltsverzeichnis

1 Einleitung

Die bleibende Abhängigkeit vom Informationsmedium Papier und der gleichzeitige Drang zum elektronischen Medium erfordern die Entwicklung intelligenter Schnittstellen zur Rezeption papiergebundener Information. Mit der wissensbasierten Dokumentanalyse am Beispiel von einseitigen Geschäftsbriefen in deutscher Sprache beschäftigt sich das Projekt **ALV** (Automatisches Lesen und Verstehen). Analog zum menschlichen Lesen wird eine enge Verflechtung zwischen automatischem Erkennen und automatischem Verstehen textueller Information in Papierdokumenten verfolgt. Diese Analyse wird in ALV in drei große Phasen aufgeteilt: in die Strukturanalyse, in die Texterkennung und in die partielle Textanalyse (siehe Abb. 1).

Umfassende Beschreibungen, vor allem von Strukturanalyse und Texterkennung, finden sich in [Dengel et al 92a] und [Dengel et al 92b]. Nachfolgend wird nur ein kurzer Überblick gegeben und auf vertiefende Literaturstellen verwiesen.

Ausgangspunkt der Analyse bildet ein gescanntes Papierdokument, das im Rechner als Bit-Matrix kodiert vorliegt. In einem Vorverarbeitungsschritt wird der Drehwinkel des Dokumentes bestimmt ([Dengel & Schweizer 89]) und bei Bedarf (Drehwinkel ungleich 0) wird eine Korrektur dieses Winkels vorgenommen ([Ali 92]).

In der Strukturanalyse wird die formale Gestalt eines Dokumentes ermittelt. Dabei werden innerhalb der Segmentierung die vom Layout vorgegebenen Teile, sogenannte Layoutobjekte, bestimmt (z.B. Text-, Graphik-, Bildbestandteile). Die Textteile werden ihrerseits hierarchisch in Blöcke, Zeilen, Wörter und Zeichen weiter zerlegt ([Fein et al 92]). Aufbauend auf geometrischen Beschreibungen solcher Layoutobjekte werden logische Objekte identifiziert (z.B. Betreffteil, Anrede oder Datum). Dieses Verfahren, das als *logical labeling* bezeichnet wird, ist in [Dengel 92] erläutert.

Die Texterkennung versucht, den Text in den Rasterbildern aufgrund visueller Merkmale zu erkennen. Dazu sollen Zeichen nicht nur isoliert betrachtet, sondern die redundante Information des Wortaufbaus genutzt werden ([Boon 92], [Molter 92]). Die Verwendung eines strukturierten Lexikons, dessen spezifische Struktur auch auf die logischen Objekten zugeschnitten ist, soll dabei die Erkennung des Textes innerhalb der logischen Objekte auf der Wortebene unterstützen ([Dengel et al 92c], [Hoch & Malburg 92]).

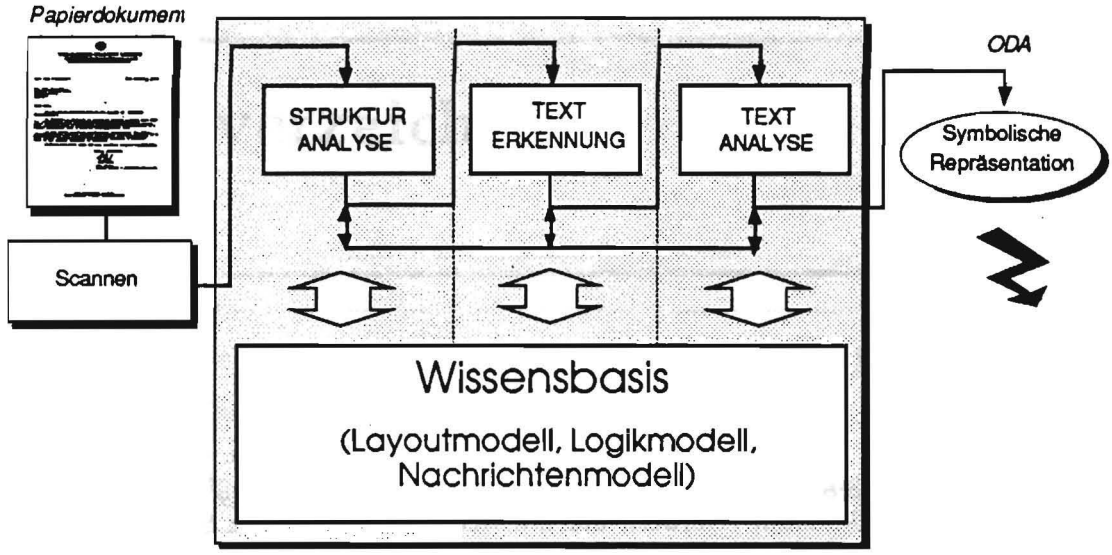

Abbildung 1: Analysephasen in ALV

Bei der Textanalyse soll in Verbindung mit logischen Objekten eine partielle Analyse des erkannten Textes durchgeführt werden. Als Besonderheit ist zu beachten, daß der Text nicht vollkommen korrekt erkannt wird. So entstehen für ein Wort oftmals mehrere Alternativen, oder aber es kann gar kein Vorschlag gemacht werden und es entstehen somit Lücken. Diese "Fehler" in der Texterkennung sind somit von allen Analyseverfahren zu berücksichtigen.

Bei der partiellen Textanalyse bestimmt die Textkomplexität eines logischen Objektes das anzuwendende Verfahren. Bei einzelnen logischen Objekten, wie Datum oder Absender, sind die zulässigen Wörter stark eingeschränkt und es bestehen strenge syntaktische Konventionen, wodurch eine Bedeutungszuordnung der enthaltenen Wörter relativ leicht zu erreichen ist (semantische Grammatiken). Andere Objekte dagegen, wie der Briefrumpf, unterliegen keiner Einschränkung und erlauben den vollen Umfang der natürlichen Sprache, so daß ein vollständiges automatisches Verstehen der enthaltenen Information zum heutigen Zeitpunkt unmöglich ist. Um trotzdem in der Lage zu sein, die zentralen Aussagen eines Dokuments zu erfassen, wird in dem Vorhaben ALV ein erwartungsgesteuertes Analysevorgehen verfolgt ([Gores 92]). Ein solches "oberflächliches" Verstehen wird auch *Text-Skimming* genannt ([DeJong 79]). Das Verstehen eines Dokuments zielt somit auf die Erfassung wichtiger inhaltlicher Aussagen ab.

Für die unterschiedlichen Aufgaben in der Dokumentanalyse wird zur Unterstützung umfangreiches Wissen in Form von Modellen vorgegeben. So existieren ein Layoutmodell und ein Logikmodell, um die Strukturanalyse und die Texterkennung effektiv zu unterstützen (siehe [Bleisinger et al 91]). In diesem Papier beschäftigen wir uns mit der letzten Phase, der erwartungsgesteuerten Textanalyse, insbesondere mit dem dafür zugrundeliegenden Modellwissen. Die wichtigen Aussagen, die das System aus den Brieftexten herausfinden soll, werden durch das Modell der Nachrichtentypen beschrieben. Jede Klasse eines Geschäftsbriefes, z.B. Bestellung oder Angebot, wird als ein Nachrichtentyp angesehen (EDIFACT in [ISO9735]). Und zu jeder Briefklasse werden die wichtigsten oft vorkommenden

Informationen abstrakt in einem Nachrichtentyp aufgelistet. Mit diesem Modell ist es damit möglich, einen großen Teil der Informationen, die in typischen Geschäftsbriefdokumenten enthalten sind, als Erwartungen im Rechner darzustellen.

Die Anforderungen, denen dieses Modell gerecht werden muß, ergeben sich aus zwei wesentlichen Grundbedürfnissen. Zum einen muß das Modell mächtig genug sein, um eine ausreichend große Menge verschiedener Arten von Geschäftsbriefen darstellen zu können. Zum anderen muß dieses Modell dem gewählten erwartungsgesteuerten Analyseverfahren entgegenkommen, d.h. nicht nur die Darstellung inhaltlicher Aspekte des Geschäftsbriefes ermöglichen, sondern auch die von Steuerungs- und Kontrollinformationen der Analysekomponente.

2 Anforderungen

Das Ziel der erwartungsgesteuerten Textanalyse im Projekt ALV ist die Ermittlung der wichtigen, im Modell der Nachrichtentypen abstrakt vorgegebenen Informationen. Damit verbunden ist die richtige Zuordnung eines Briefdokumentes in eine bestimmte Klasse von Briefen, repräsentiert durch die Nachrichtentypen. Das Konzept der erwartungsgesteuerten Textanalyse in ALV ist in Abbildung 2 dargestellt.

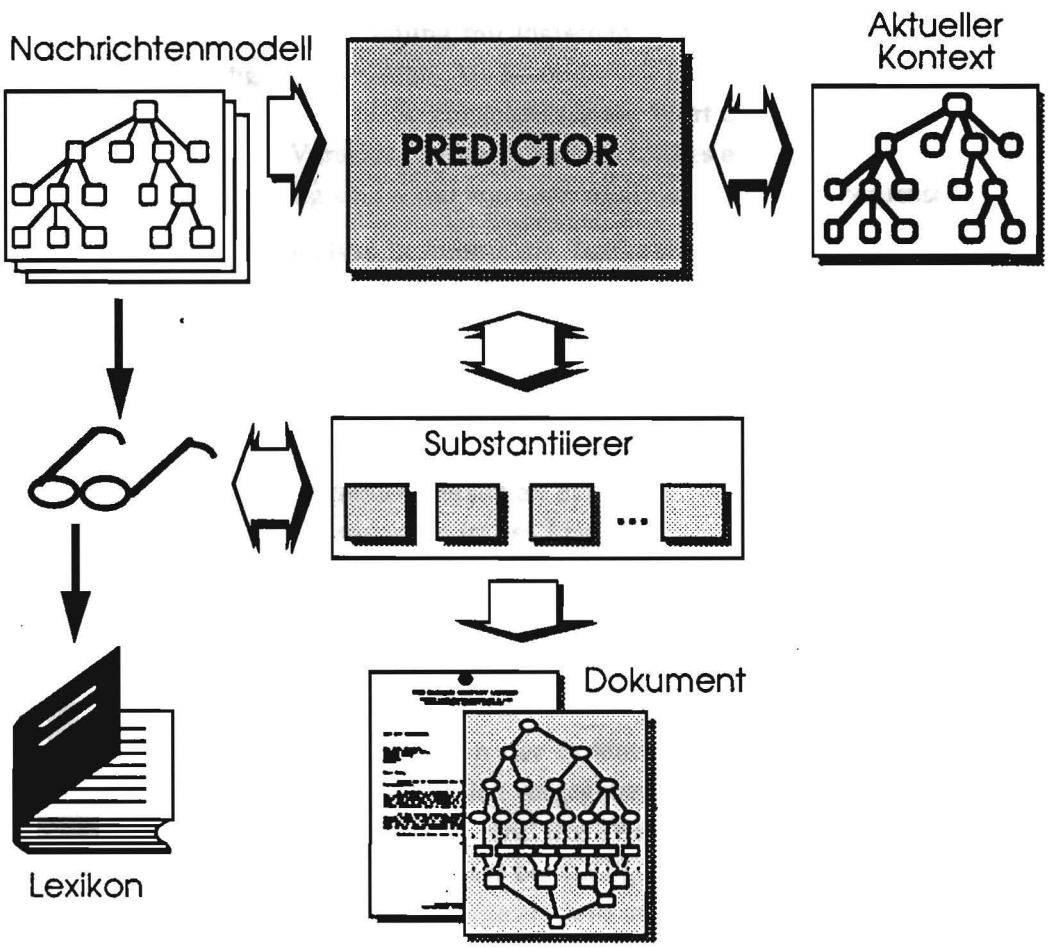

Abbildung 2: Das Text-Skimming Konzept in ALV

Der Predictor ist die zentrale Steuerungskomponente. Dieser interpretiert die Erwartungen im Modell der Nachrichtentypen, um die inhaltliche Analyse, unter Einbezug des aktuellen Kontexts (bisherige Analyseergebnisse), gezielt zu betreiben. Die eigentliche Analysearbeit auf dem Brieftext wird durch die vom Predictor aktivierten Substantiierer durchgeführt, die auf ein durch "lexical views" strukturiertes Lexikon zurückgreifen. Lexical views stellen eine eingeschränkte Sicht auf das Lexikon dar, also eine Teilmenge von Worten des Lexikons.

Die aus diesem Konzept resultierenden sowie die bezüglich der wichtigen Informationen in Briefklassen anfallenden Anforderungen an das Modell der Nachrichtentypen werden im folgenden diskutiert.

Die einzelnen Klassen von Briefdokumenten lassen sich in mehrere Bestandteile zerlegen, diese sind z.B. Adressen des Absenders und Empfängers, Datumsangaben und andere mehr, z.B. wer bestellt, was wird bestellt, warum wird sich beschwert? Mit diesen Bestandteilen als Vorrat können die Nachrichtentypen beschrieben werden. Die Anforderungen an das Nachrichtenmodell bestehen damit aus den folgenden Punkten:

- Bereitstellung einer Syntax, mit der die Bausteine der Nachrichtentypen beschrieben werden können.
- Erweiterung dieser um Kontrollstrukturen, die effiziente und mächtige Definitionen zulassen, z.B. Iteration (`list`) und Auswahl (`cond`) von Bausteinen (Alternativen) und eine hierarchische Struktur.
- Modifikation von Constraints (z.B. ist der Constraint für das Produkt in einem Angebot in Abhängigkeit davon, welche Firma anbietet, von der lexical view `LV-product` (alle möglichen Produkte) zu `LV-product-company` (die des Anbieters) zu ändern. Durch die Modifikatoren wird die Sicht auf einen Nachrichtentyp geändert.

Damit sind die Anforderungen gestellt, die eine Kodierung der für die Dokumentanalyse interessierenden Inhalte eines Geschäftsbriefes ermöglichen. Diese Inhalte werden im folgenden als *Informationseinheiten* bezeichnet. Mit ihnen alleine ist es möglich, das Wesentliche des Briefdokumentes zu beschreiben. Für die Anwendung dieser Darstellung in einem erwartungsgesteuerten Analysesystem müssen noch sogenannte *Steuerungseinheiten* hinzugenommen werden. Dazu gehören

- Direktiven für die Analyseeinheit (*Predictor*), z.B. die Angabe der Wichtigkeit von Bausteinen (Muß ein Baustein im Brief auftauchen? Darf er fehlen?), seiner Bedeutung (Wie charakteristisch ist er für einen Nachrichtentyp?), ihrer erwarteten Reihenfolge und Position im Text (zur Erleichterung der Analyseaufgabe), ihrer Verbindung untereinander (Welche Beziehung haben die Bausteine zueinander, welche Regeln gibt es zwischen ihnen?). Diese Direktiven werden vom Predictor interpretiert, sie steuern dessen Vorgehensweise.
- Hierarchie- und Verwandschaftsangaben. Die Definition der Nachrichtentypen in einer hierarchischen Form erleichtert diese nicht nur, sondern bietet die Möglichkeit zur Anwendung bestimmter Analysealgorithmen des Predictors. Dafür werden weitere

Sichten des Nachrichtenmodells definiert ([Gores 92]), die eine effizientere Analyse erlauben, ohne besonders mächtige Parser als notwendig vorauszusetzen.

Mit einem obigem Anforderungskatalog genügenden Nachrichtenmodell lassen sich - in den Grenzen, die durch unser Modell einer erwartungsgesteuerten partiellen Textanalyse vorgegeben sind - eine Menge von Briefdokumenten analysieren und repräsentieren. Durch die Einführung sogenannter Multi-Nachrichtentypen wird sowohl die Bearbeitung von Dokumenten, die nicht eindeutig einem Nachrichtentyp zuordbar sind, als auch die von mehreren Briefen umfassenden Korrespondenzen ermöglicht.

Im folgenden Kapitel 3 wird zunächst ein Beispiel einer Modellierung des Nachrichtentyps einer Bestellung `MT-order` auf einer abstrakten Ebene angegeben. Die zur konkreten Beschreibung nötigen Anforderungen führen zur *Conceptual Dependency*, die in Kapitel 4 vorgestellt wird, und der Definition des Nachrichtenmodells in Kapitel 5.

3 Die Briefklasse Bestellung

Durch das Nachrichtenmodell soll es möglich sein, die wichtigen Inhalte eines Geschäftsbriefes zu repräsentieren. Betrachtet man verschiedene Briefklassen, stellt man fest, daß Briefe in verschiedene inhaltliche Einheiten gruppierbar sind, die Elemente eines Briefes. Dazu zählen z.B. die Adressen des Absenders und Empfängers, das Datum, der Betreffteil, der Briefrumpf und andere mehr. Viele dieser Elemente treten in allen Briefklassen auf, einige sind speziell für eine bestimmte Klasse. Mit allen Elementen lassen sich durch geeignete Kombination verschiedenartige Klassen von Briefdokumenten beschreiben. Abbildung 3 zeigt einen Bestellbrief.

Die interessanten inhaltlichen Bestandteile dieses Briefes sind Absender, Empfänger, das Datum und die Fakten der Bestellung, nämlich die Produkte. Für diese müssen im Nachrichtenmodell Nachrichtenelemente definiert werden, deren Kombination beschreibt als Nachrichtentyp die Briefklasse Bestellung.

Zur Beschreibung der Klasse der Geschäftsbriefe für Bestellungen wird der Nachrichtentyp Bestellung definiert. Er enthält die Nachrichtenelemente Absender, Empfänger, Datum, und die eigentliche Bestellung. Die meisten dieser Bausteine treten nur einmal auf, einige sind optional, andere obligatorisch. Eine Besonderheit stellt die erwartete Bestellung dar, da sie wiederholt auftreten darf.

Don Hagenbuch
Ötztalstraße 42
4711 Bless-Hohenstein

10.11.1992

Deutsches Forschungszentrum
für Künstliche Intelligenz
z.Hd. Herr Prof. Gerhard Barth
Erwin-Schrödinger-Straße
Postfach 2080
6750 Kaiserslautern

Betreff: Bestellung

Sehr geehrter Prof. Barth,

hiermit bestellen wir

2	ALV-System	423,23 .- DM
1	DFKI-Document DFKI D-92-99: "Die Verachtung der Biologie als musikalische Maßnahme"	99,34 .- DM

Mit freundlichen Grüßen

(Don Hagenbuch)

Anlagen

Abbildung 3: Eine Bestellung

Jedes Nachrichtenelement muß die entsprechenden Informationen des Briefes repräsentieren können. Abbildung 4 zeigt für das Beispiel des Empfängers und einer Bestellaktion das zugehörige Nachrichtenelement und die eingetragenen Inhalte:

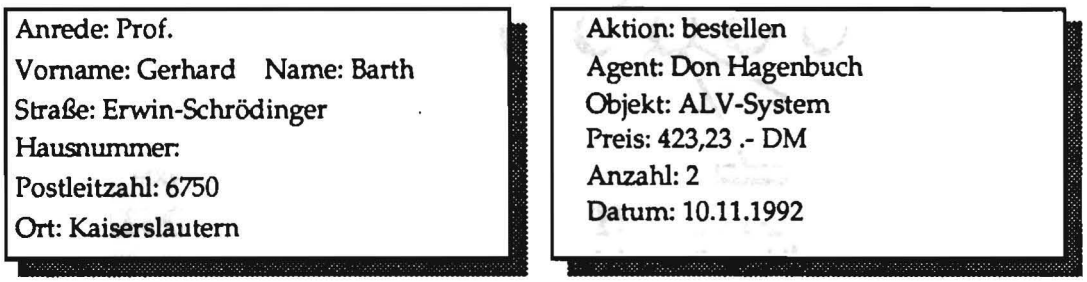

Abbildung 4: Ein statisches und ein aktives Nachrichtenelement

Dabei werden statische Elemente benutzt um Zustände darzustellen, mit aktiven Elementen werden Zustandsänderungen beschrieben. Mit den Bausteinen der Nachrichtenelemente werden die Nachrichtentypen als Elemente der obersten Stufe des Nachrichtenmodells definiert. Sie sind durch die Hinzunahme von Kontrollstrukturen (z.B. zum wiederholten auftreten des Bestellens eines Artikels) jedoch mehr, als nur die Summe ihrer Bestandteile. Die Nachrichtenelemente in der mittleren Stufe des Nachrichtenmodells, z.B. die Empfängeradresse, müssen zum einen die Informationen des Textes aufnehmen können, zum anderen Erwartungen über deren Eigenschaften enthalten. Jeder Slot eines Nachrichtenelementes (z.B. die Anrede, der Name etc.) muß mehrere Aspekte berücksichtigen:

- das im Text gefundene Wort
- Constraints, die die Menge zulässiger Worte beschränken
- Angaben über benötigte Substantiierer
- logische Objekte oder Layoutobjekte, in denen ein Slotfüller gefunden werden kann.

Das Nachrichtenmodell hat damit den folgenden Aufbau: Nachrichtentypen als Repräsentanten einer bestimmten Klasse von Briefdokumenten, die durch die Kombination von Nachrichtenelementen und Kontrollstrukturen gebildet werden. Die statischen oder aktiven Nachrichtenelemente werden durch Angabe mehrerer Aspekte für alle ihre Bestandteile beschrieben.

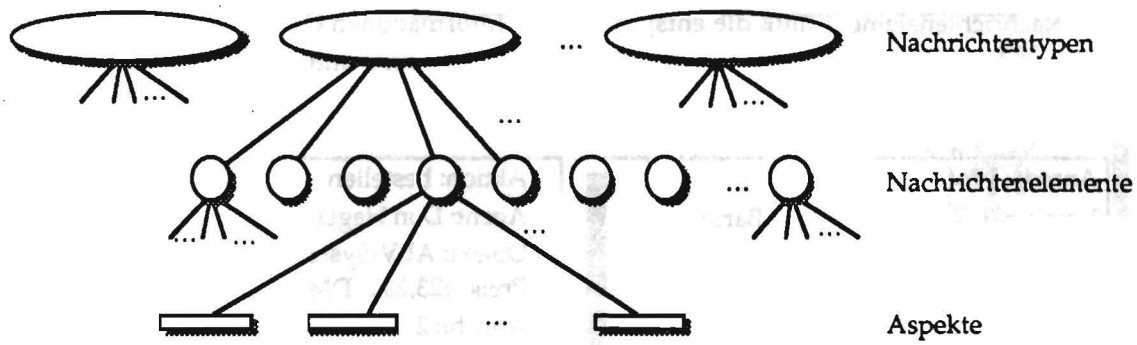

Abbildung 5: Die Inhalte des Nachrichtenmodells

Als Grundlage der Repräsentation der Nachrichtenelemente dient der Formalismus der *Conceptual Dependency*, der im folgenden Kapitel beschrieben wird.

4 Conceptual Dependency

Der Formalismus der Conceptual Dependency bildet die Grundlage der Repräsentation und Interpretation der Nachrichtentypen und beeinflußt - in geringerem Umfang - auch das Lexikon. Conceptual Dependency (CD) ist eine Theorie der (maschinellen) Verarbeitung natürlicher Sprache, die erstmals von Schank ([Schank 1972]) publiziert wurde. Das Ziel war es, Rechnern Sprachverstehen soweit zu ermöglichen, daß eine Zusammenfassung, Paraphrasierung oder gar Übersetzung machbar ist.

Um genau zu verstehen, wie die Nachrichtentypen aufgebaut sind, und um einen Nachrichtentyp konstruieren zu können, muß die Grundlage des Nachrichtenmodells, die Conceptual Dependency bekannt sein.

Sinn der Conceptual Dependency

Der Anspruch der Conceptual Dependency liegt darin, natürlich-sprachlichen Äußerungen (Sätzen) eine Repräsentationsform zuzuweisen, die maschinell verarbeitet werden kann. Sprachliche Äußerungen werden durch Slot/Werte-Paare dargestellt, die *CD-Formen* (*Konzeptualisierungen*). In Abhängigkeit davon, ob der Satz eine Aktion oder einen Sachverhalt beschreibt, wird ihm eine aktive oder statische CD-Form zugewiesen. Die statischen Formen enthalten damit also das im Text genannte faktische Wissen, während die aktiven Formen die Veränderungen repräsentieren. Durch die Möglichkeit zur Kombination aktiver Konzeptualisierungen können mit den Basis-Formen auch komplizierte Satzstrukturen dargestellt werden. Die Interpretation der CD-Formen erfolgt nach einer festen Semantik, die z.B. für die Aktionen deren Konsequenzen festlegt und die Belegungen von Slots einschränkt. Die grundlegenden Fragen zu einem Satz und dessen Konzeptualisierung sind:

- Welche Schlußfolgerungen sind möglich ?
- Wann werden sie gezogen ?
- Woher kommen sie ?

Damit zwei gleichbedeutende Sätze, die eine verschiedene Syntax haben, die gleichen Schlußfolgerungen zulassen, fordert die Conceptual Dependency, daß ihnen die gleiche Konzeptualisierung zugewiesen werden muß. Dazu muß jede im Satz explizit oder implizit vorhandene Information in der Konzeptualisierung explizit enthalten sein.

Für das hier vorgestellte Nachrichtenmodell gilt dabei folgende Einschränkung: da hier nur von erwarteten CD-Formen ausgegangen wird, kann auch nur die erwartete Information explizit gemacht werden. Alles übrige, die Sätze möglicherweise unterscheidende, wird ignoriert. Dadurch kann semantisch verschiedenen Sätzen die gleiche CD-Form zugewiesen werden. Wird etwa eine Konzeptualisierung erwartet, die nur die Belegung der Rollen (Slots) von Agent, Aktion und Objekt vorsieht, so erhalten die Sätze

1) Hagenbuch schlägt Prager im Schach.

2) Hagenbuch schlägt Prager Kontemplation vor.

3) Prager wird von Hagenbuch im Schach geschlagen.

die gleiche Belegung dieser Rollen (also "Hagenbuch" als Agent, "schlagen" als Aktion und "Prager" als Objekt), obwohl lediglich Satz 1) und 3) semantisch gleich sind.

Conceptual Dependency und Nachrichtentypen

Die CD-Formen bilden, erweitert zu Nachrichtenelementen, die Bausteine der Nachrichtentypen, die die Informationen des Briefes aufnehmen sollen. Die rudimentäre Semantik der CDs wird durch weitere Constraints für die Slotbelegungen erweitert. Damit wird ein weitgehendes Verständnis des erkannten Textes durch den Rechner angestrebt. Der Formalismus der Conceptual Dependency wurde gewählt, weil er durch die Bereitstellung der Primitive ausdrucksmächtig genug ist, um die hier interessierende Teilmenge der Schriftsprache, nämlich Geschäftsbriefe auszudrücken. Zur Anpassung an die Bedürfnisse der Domäne wurde die Conceptual Dependency teilweise erweitert und an einem wichtigen Punkt, den instrumentellen Aktionen, gekürzt. Wir verwenden den Formalismus jedoch nur in der Einschränkung, Erwartungen über Sätze zu beschreiben, nicht um beliebigen Sätzen, die im Text gefunden werden, eine CD-Form zuzuweisen. Dies könnte eine Perspektive für die Weiterentwicklung der erwartungsgesteuerten Analyse in ALV sein.

Die CD-Formen sind hier aber nicht nur als Behältnis der passenden Worte sprachlicher Äußerungen vorgesehen. Der Blickpunkt der Nachrichtentypen ist geprägt von den Erwartungshaltungen an den Slot-Füller. Daher werden die Slots in *Aspekte* feinstrukturiert, in denen die Anforderungen (Constraints) an die passenden Worte und Informationen für die Analyse notiert werden.

Conceptual Dependency und Lexikon

Die Anforderungen der Conceptual Dependency an das Lexikon bestehen in einer Zuordnung der Worte in syntaktische, grammatikalische, schwach-semantische oder auch domänenabhängige Kategorien. Alle diese Kategorien werden in den Constraint-Aspekten der CD-Formen durch lexical views angesprochen, die im Lexikon definiert sein müssen. Um einen Slot mit einem Wort des Textes füllen zu können, müssen die erwarteten und dem gefunden Wort zugehörigen Kategorien kompatibel sein. Die lexical views in den Constraints der Erwartung müssen also in der Hierarchie der lexical views über denen des gefundenen

Wortes liegen. Die Verwendung der semantischen Constraints verfolgt die Absicht, den Slots eine Semantik zuzuordnen.

Überblick

Im folgenden wird zunächst die Strukturierung der CD-Slots erklärt, die neben der textuellen Information weitere Constraints und Steuerungshinweise in den Aspekten enthalten. In Kapitel 4.2 werden die statischen CD-Formen und deren Slots beschrieben. Hierzu gehören Objekte und als Spezialfall die Agenten. Im Anschluß daran werden in Kapitel 4.3 die das Herz der Conceptual Dependency bildenden aktiven CD-Formen vorgestellt und die Interpretation, mit denen den Konzeptualisierungen eine Bedeutung zugeordnet werden kann.

4.1 Aspekte von CD-Slots

Die Konzeptualisierungen oder CD-Formen werden durch bestimmte Slot-Werte-Paare beschrieben. Die Werte der CD-Slots werden jetzt weiter differenziert. Aus der Sicht der erwartungsgesteuerten Analyse besteht die wichtige Information eines CD-Slots nicht allein im gefundenen Wort, sondern in den *Aspekten*[1]. Sie enthalten Angaben zur Einschränkung der zulässigen textuellen Information auf lexical views, z.B. syntaktische oder semantische Kategorien. Als Weiteres werden Informationen zur Beeinflussung des Ablaufs angegeben. Die allgemeine Definition eines CD-Slots orientiert sich also an den Bedürfnissen der erwartungsgesteuerten Analyse und zum Teil an der Strukturierung des Nachrichtenmodells. Die Aspekte eines Slots enthalten neben dem Wort oder der Phrase, die als Füller gefunden wurde, Anforderungen an die Substantiierer: Welche Leistungen werden benötigt? Welche syntaktischen und semantischen Constraints müssen erfüllt werden? Welche Regeln können angegeben werden?

Einige dieser Informationen werden auch für die Nachrichtenelemente benötigt. Daher wird die Klasse `cd-slot` durch die Klasse der `cd-constraints` definiert. `cd-constraints` gibt die Aspekte an, die in allen Nachrichtenelementen benötigt werden. Sie enthält nur die Angabe von Steuerinformationen und nimmt keine Information aus dem Text auf. `cd-slot` enthält die Aspekte von `cd-constraints` durch Vererbung und die Aspekte, die nur in Slots sinnvoll sind, durch direkte Definition. Die Klassen haben die Form:

---

1 Der Terminus Aspekte wird hier im Sinne der Definition in [Dilger 1989] benutzt.

```
(defclass cd-constraints()
   ((need-substantiator
      :type list
      :initform '(*general-substantiator*)
      :reader need-substantiator)
   (expected-locations
      :type list :initform '()
      :reader expected-locations))
)


(defclass cd-slot(cd-constraints)
   (  (phrase-found
      :type list
      :initform '()
      :accessor  phrase-found)
      (syntactic-constraints
         :type list
         :initform '()
         :reader syntactic-constraints)
      (semantic-constraints
         :type list :initform '()
         :reader semantic-constraints)
   )
)
```

Diese beiden Klassen können für die individuellen Anforderungen bestimmter Slots in den Constraints erweitert werden, die Struktur der 2 bzw. 5 Slots bleibt jedoch gleich. Die Klasse `cd-slot`, die sich durch die Aspekte zur Aufnahme eines Textelementes und die Constraints unterscheidet, dient als Basis der Definition aller Slots. Dies gilt für alle in den Kapiteln 4.2 und 4.3 definierten CD-Formen. Damit stellt die CD-Form die Kernstruktur des Nachrichtenmodells dar. Im folgenden werden die Bedeutungen der Aspekte eines Slots im einzelnen erklärt:

need-substantiator

Um eine Information aus dem Text und dem aktuellen Kontext zu gewinnen, stehen dem Predictor mehrere Substantiierer zur Auswahl. Der Eintrag für diesen Aspekt gibt eine Auswahl von Substantiierern einer bestimmten Leistungsklasse vor, die für diesen Slot, d.h. das Füllen des `phrase-found`-Aspektes, besonders kompetent sind. Zur Interpretation dieses Aspektes wird davon ausgegangen, daß die Substantiierer verschiedenen Leistungsklassen zugeordnet sind und die Leistungsanforderungen in einem Nachrichtenelement oder dessen Slot vom Predictor auf diese Klassen abgebildet werden können.

Für die Spezialisierung `address-name-slot` der `cd-slot`-Klasse kann die Auswahl der Substantiierer z.B. auf `*sender-substantiator*` und `*recipient-substantiator*` eingeschränkt werden. Alle Adreßnamenslots, die als Belegung Instanzen der Klasse `address-name-slot` haben, erhalten damit die gleichen empfohlenen Substantiierer. Der Beispieleintrag `(*sender-substantiator*` und `*recipient-substantiator*)`

schlägt die Liste der Substantiierer vor, die Absender- und Empfängeradressen erkennen können.

expected-locations

Oft ist der Bereich, in dem eine Information zu finden ist, einschränkbar. Die Einschränkungen beziehen sich dabei sowohl auf logische Objekte wie auch Layoutobjekte, die in vorhergegangenen Analysephasen ermittelt wurden. Dadurch kann die Analyse gezielter vorgehen und Substantiierer einsetzen, die von der Bereichseinengung profitieren. In Verbindung mit einem (vorherigen) logical labeling des Dokumentes gewinnt dieser Eintrag große Bedeutung für die Effizienz der Analyse.

Der Aspekt `expected-locations` enthält eine Liste von Empfehlungen für Bereiche, in denen der gesuchte Text sein könnte. Mit dem Beispielwert `((logical-object sender_address (line-position 1 3)) (logical-object recipient_address (line-position 1 3)))` wird der Bereich auf die erste bis dritte Zeile des logischen Objekts Absender- oder Empfängeradresse eingeschränkt. Die Existenz von Zugriffsfunktionen für die Substantiierer ist eine Voraussetzung, um solche Angaben nutzen zu können.

phrase-found

In diesem Aspekt wird die Phrase oder das einzelne Wort, das als Füller zulässig ist, eingetragen. `phrase-found` ist der einzige Aspekt jedes Slots, der durch die Analyse verändert wird. Er tritt nur innerhalb der Klasse `cd-slots` auf, auf der Ebene der Nachrichtenelemente spielt er keine Rolle. Daher wurden die Klassendefinitionen in `cd-constraints` und `cd-slots` getrennt. Der Aspekt wird mit verschiedenen Werten belegt, die in einer Liste gesammelt werden. Dadurch kann die Information jederzeit leicht erweitert werden. Die Belegung dieses Aspektes begründet wesentliche Anforderungen an das Lexikon. Die Elemente der Liste sind

- die Zeichenkette, die im Text gefunden wurde.
- die Wortpositionen dieser Zeichenkette. Sie werden als `(word-position <from>.<to>)` angegeben. Für eine Phrase, die durch andere Worte unterbrochen ist, z.B. „Mit *besonders ehrlichen* freundlichen Grüßen", können die Wortpositionen nicht als zusammenhängender Bereich angegeben werden. Die Notation hat dann die Form `(word-position ({<from>.<to>}+))`.
- die Referenz auf die Lexikoninformation. In einer Liste werden der Verweis auf den Lexikoneintrag und morphologisch syntaktische Angaben, wie das Wort von der Stammform in die Textform abgebildet wurde, eingetragen[2]. Durch die morphologisch syntaktischen Angaben kann Semantik transportiert werden, die anderweitig nicht erkennbar ist. Falls die im Text gefundene Zeichenkette, z.B. durch eine fehlerhafte

[2] Dieser Eintrag wird durch die Lexikonanfrage erzeugt und an die Substantiierer bzw. den Predictor weitergereicht.

Texterkennung, mehrdeutig ist, muß die Referenz für alle Alternativen angegeben werden.

- die Referenz auf ein bestimmtes Objekt der Welt oder einer Wissensbasis, das mit dieser Zeichenkette angesprochen wird. Für mehrdeutige Zeichenketten muß für alle Alternativen die Referenz angegeben werden.

Die Gesamtheit aller gefüllten `phrase-found`-Aspekte - und die Slots, denen sie zugehören - stellen das Ergebnis der Analyse dar. Der Beispielwert `("Hagenbuch" (word-position 3 3 ) (↑Hagenbuch ...) ⇑Hagenbuch)` umfaßt die Textinformation als Zeichenkette, die Phrasenposition durch Beginn und Ende gekennzeichnet, einen (hier unvollständigen) Verweis auf den Lexikoneintrag und das Objekt, das mit dem Namen `Hagenbuch` bezeichnet wird.

syntactic-constraints

Mit syntaktischen Constraints werden die elementaren Einschränkungen der erwarteten Phrase angegeben, nämlich die syntaktischen Kategorien, denen sie zugeordnet sein muß. Die Constraints werden als lexical views angegeben. Die Worte im Lexikon und die Antwort des Substantiierers müssen eine Zuordnung zu einer lexical view haben, damit dieser Constraint geprüft werden kann.

Mit dem Beispielwert `(LV-N LV-NP)` werden nur Nomen oder Nominalphrasen akzeptiert.

semantic-constraints

Semantische Constraints stellen verschärfte Anforderungen an die möglichen Füller des `phrase-found`-Aspektes. Sie fordern die Zugehörigkeit zu einer semantischen Kategorie oder die Gültigkeit eines Prädikates für den Füller. Constraints dürfen sich nicht auf Instanzen (z.B. die Agenten `Barth` und `Wendel`, die im Lexikon notiert sind) beziehen, da solche Einschränkungen domänenspezifisch sind. Stattdessen werden lexical views verwandt. Für obiges Beispiel wird die neue lexical view `LV-geschäftsführer` vorgeschlagen, die für die Domäne DFKI dann nur besagte Agenten enthält.

Die zulässigen Belegungen sind:

- lexical views
  Hiermit werden die zulässigen Füller hinsichtlich der Eigenschaften eingeschränkt, die durch die lexical view bezeichnet wird.
- Logische Operatoren: `(not {lexical-view}+)`, `(or {lexical-view}+)`, `(and {lexical-view}+)`
  Die durch lexical views angegebenen Constraints werden logisch verknüpft. Nur Füller die keiner (`not`), mindest einer (`or`) oder allen (`and`) lexical views angehören, werden akzeptiert. Die Benutzung der Negation kann problematisch sein[3].

---

3 Das hängt davon ab, wie das Wissen der lexical views organisiert und die Negation implementiert ist (closed world assumption, negation as failure).

- Lisp-Prädikate[4].

Abschlußbeispiel:

Mit der folgenden Definition wird eine Klasse erzeugt, deren Instanzen die Erwartung eines Namens beschreiben[5].

```
(defclass address-name-slot (cd-slot)
   ((need-substantiator
      :type list
      :initform
         '(   *sender-substantiator* *recipient-substantiator*
              *address-substantiator*)
      :reader need-substantiator
      :allocation :class)
   (expected-locations
      :type list
      :initform '(   (logical-object
                      sender_address (line-position 1 3))
                     (logical-object
                      recipient_address (line-position 1 3)))
      :reader expected-locations
      :allocation :class)
   (phrase-found
      :type list
      :initform '()
      :accessor phrase-found)
   (syntactic-constraints
      :type list
      :initform '(LV-N LV-NP)
      :reader syntactic-constraints
      :allocation :class)
   (semantic-constraints
      :type list
      :initform '(or LV-customer LV-supplier)
      :reader semantic-constraints
      :allocation :class))
)
```

Eine Instanz dieser Klasse benötigt zur Füllung Substantiierer der Leistungsklassen `*sender-substantiator*`, `*recipient-substantiator*` oder `*address-substantiator*`. Der Füller wird im Bereich der ersten bis dritten Zeile der logischen Objekte Empfänger- oder Absender-Adresse vermutet. Als syntaktische Einschränkungen werden nur Nomen oder Nominalphrasen zugelassen. Die semantischen Constraints erwarten, daß die lexical views `LV-customer` oder `LV-supplier` gelten, der Füller von `phrase-found` muß also ein Kunde oder ein Lieferant sein. Obige Definition gibt im Nachrichtenmodell die Erwartungen über

---

4 Zulässige Prädikate sollen einfach testbar sein, also etwa `(and (> 1000 ) (< 9999))` zur Angabe eines Intervalls.

5 Die Angabe `:allocation :class` legt fest, daß der Wert nur in der Klassendefinition gespeichert ist.

einen Eintrag eines Nachrichtenelementes an (z.B. den Absender), das als Bestandteil den Namen eines Kunden oder Lieferanten enthält. Alle anderen Bestandteile dieses Nachrichtenelementes werden auf analoge Art definiert.

Bemerkung

In den beiden nächsten Kapiteln 4.2 und 4.3 werden die statischen und aktiven CD-Formen definiert, die die Grundlage der in Kapitel 5 definierten Nachrichtentypen bilden. Dabei wird häufig von den Slots dieser Formen gesprochen, als gäbe es nur einen Aspekt, nämlich den füllenden Text in `phrase-found`. Dies ist jedoch nur eine **abkürzende Schreibweise**, jeder der Slots hat die in diesem Kapitel definierte mehrdimensionale Struktur der Aspekte, die Klasse `cd-slot` ist Oberklasse der Slots der aktiven und statischen CD-Formen!

4.2 Statische CD-Formen

Mit statischen CD-Formen werden Erwartungen über Eigenschaften und Zustände von Objekten notiert. Als statische CD-Formen lassen sich viele der Informationen eines Geschäftsbriefdokumentes, wie etwa Adressen, Datum etc. durch Referenz des Objektes, des Zustandes und des neuen Wertes, darstellen. Für die Implementierung des Nachrichtenmodells wurde die Notation statischer CD-Formen leicht modifiziert. Die `state` und `value` Slots wurden um die Angabe der Maßeinheit `measurement` erweitert und zu einem Tripel in der Klasse `prop-class` zusammengefaßt. Instanzen dieser Klasse können beliebig oft innerhalb einer Liste als Belegung des neuen Slots `properties` angegeben werden. Die Syntax einer statischen CD-Form hat damit die Form:

```
( object   <object>
  owner <owner>
  is-a  <super-class>
  properties (( {<prop-class instance>}*) )
)
```

Um die Einordnung der Objekte in einer Klassenhierarchie auszudrücken und Teilebeziehungen zuzulassen, wurden die Slots `is-a` und `owner` zugelassen. Mit `is-a` wird die Oberklasse eines Objektes angegeben, `owner` verweist auf das Objekt, zu dem es gehört.

Die Slots statischer CD-Formen haben die oben angegebene Feinstruktur der Aspekte, durch die die Erwartungen an den Füller definiert werden.

4.3 Aktive CD-Formen

Mit den aktiven CD-Formen werden Erwartungen über Elemente einer sprachlichen Äußerung repräsentiert. Sie beschreiben eine Veränderung des Kontextes durch Aktionen. In der Conceptual Dependency Theorie gibt es eine Basis an primitiven Aktionenklassen, die hier

als lexical views die semantischen Constraints der `action-slot`'s bilden. Genauer gesagt heißt das, daß der `semantic-constraint`-Aspekt des `action-slot`'s mit ein oder mehreren dieser lexical views belegt werden. Die aktiven CD-Formen der Aktionen in Kapitel 4.3.1 werden durch die Angabe der verschieden lexical views als Constraint definiert. Die Besonderheiten der Aktionenklassen werden durch die Angabe von Constraints für die übrigen Slots der Aktionen angegeben. Mit jeder der Aktionen-Klassen sind eine bestimmte Semantik und zusätzliche Regeln verbunden, die angeben, wie die Klassen kombiniert und interpretiert werden können. Für die Domäne der Geschäftsbriefdokumente wurde die Menge der Aktionen erweitert. Der Sinn der Aktionenklassen der CD und unserer Erweiterungen liegt vor allem in der späteren Interpretation der Konzeptualisierungen.

Eine einfache aktive CD-Form besteht aus den Slots:

```
( action         <action-slot>
  agent          <agent-slot>
  object         <object-slot>
  direction_from <direction_from-slot>
  direction_to   <direction_to-slot>
  date           <date-slot>
)
```

Die Belegung dieser Slots besteht aus Instanzen von Slotklassen, deren Oberklasse cd-slot ist. Von entscheidender Bedeutung für eine mögliche Interpretation ist die Art der Aktion einer aktiven CD-Form. Die Primitive, die als `semantic-constraints` des Aktionenslots vorgegeben sind, werden im folgenden Unterkapitel erläutert. Die Füller der Slots `agent`, `object`, `direction_from`, `direction_to` und `date` werden durch geeignete Constraints eingeschränkt, denen durch die Conceptual Dependency keine implizite Bedeutung zugewiesen ist.

4.3.1 Aktionenklassen

Der Aktionen-Slot aktiver CD-Formen dient zur Aufnahme der Aktion, die in einer natürlichsprachlichen Äußerung verwandt wird. Die zulässigen Füller müssen syntaktisch Verben oder Verbphrasen sein (`syntactic-constraint-Aspekt`). Semantisch wird durch den `semantic-constraint`-Aspekt eine Zuordnung zu einem oder mehreren der Primitive `atrans`, `ptrans`, `propel`, `move`, `grasp`, `ingest`, `expel`, `mtrans`, `mbuild`, `speak` und `attend` erwartet. Diese müssen im Lexikon als entsprechende lexical views `LV-atrans`, `LV-ptrans` etc. definiert sein. Den Aktionenklassen ist eine Semantik zugeordnet, die für die Interpretation der Konzeptualisierung und die möglichen Konsequenzen von Bedeutung ist.

Mit den primitiven Aktionen-Klassen der Conceptual Dependency ist es möglich, die wesentlichen Vorgänge, wie sie zur Beschreibung einer linguistischen Proposition nötig sind, zu kodieren. Primitive Aktionenklassen werden durch einfache aktive CD-Formen und die entsprechenden Constraints beschrieben. Komplexere Vorgänge werden durch Erweiterung der Primitive um neue Slots aufgebaut. Für die Domäne typische Aktions-Formen (Primate)

werden als Erweiterung der Primitive hinzugenommen. Auf die Möglichkeit geschachtelter CD-Formen durch sogenannte instrumentelle Aktionen wird verzichtet.

Die Erwartungshaltung der Nachrichtentypen an die Wort-Füller der Aktionen-Slots fordert, daß die Worte des Lexikon nach den folgenden Klassen gekennzeichnet sind, wobei Zuordnungen zu mehreren Klassen nicht nur möglich, sondern die Regel sind. Die Hierarchie der primitiven Aktionen hat vereinfacht die folgende Gestalt:

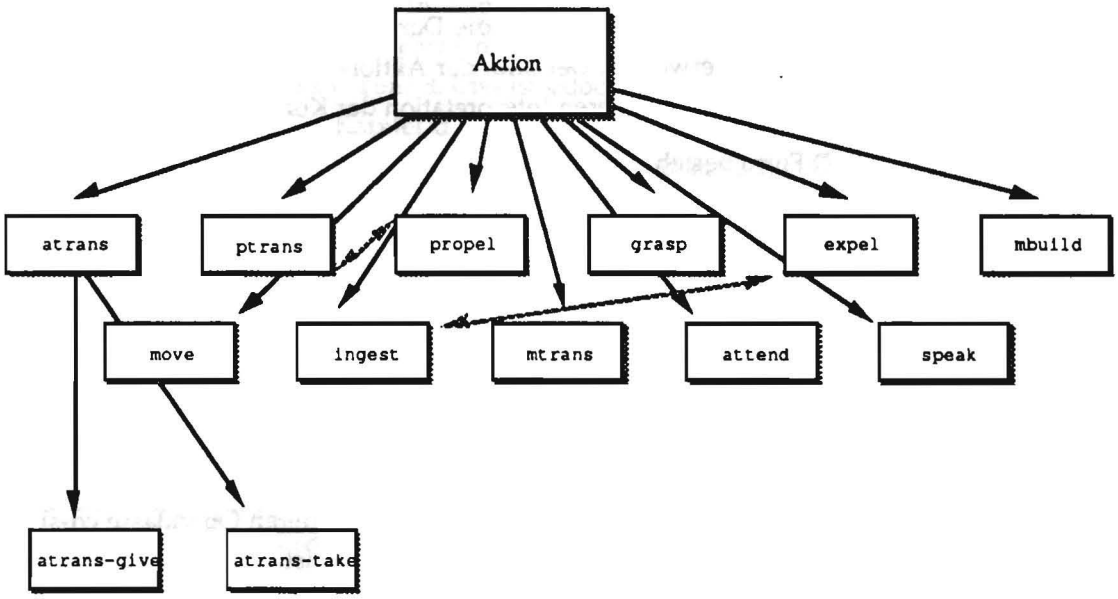

Abbildung 6: Die primitiven Aktionenklassen

Mit den vertikalen Pfeilen wird dabei die übliche Vererbung von Slots und Werten ausgedrückt. Die primitiven Aktionen stehen oft in einer kausalen Beziehung zueinander, z.B. folgt auf ein `propel` gewöhnlich ein `ptrans`, auf ein `ingest` ein `expel`. Diese Beziehungen werden durch die gestrichelten Pfeile angedeutet.

4.3.1.1 ATRANS

Der Klasse `atrans` durch eine lexical view zugeordnete Verben bezeichnen die Übertragung abstrakter Beziehungen, etwa Besitz oder Kontrolle. Solche Verben sind z.B. "nehmen", "schenken", "kaufen". Wie diese Beispiele nahelegen, läßt sich `atrans` als Oberklasse verstehen, für deren spezialisierende Subklassen einschränkende Eigenschaften angegeben werden. Ein typisches Beispiel sind die `atrans`-Klassen, in denen der Agent stets eine bestimmte der beiden Richtungsangaben vorgibt. So sind die meisten Aktionen innerhalb eines Briefes Aktionen des Absenders und gehen auch von diesem aus. Dadurch kann der Aufwand zur Definition der Nachrichtentypen vermindert werden, da die entsprechenden Constraints nicht erneut explizit definiert werden müssen. Für `atrans` lassen sich die in Abbildung 6

eingetragenen Unterklassen `atrans-give` und `atrans-take` angeben, die für den Richtungswert von `direction-from` bzw. `direction-to` den Agenten der Aktion erwarten.

Interpretation

Die Konsequenzen einer `atrans`-Aktion sind die Kenntnis des stattgefundenen Vorganges bei der oder den betroffenen Parteien. Eine Interpretation der durch die erwartungsgesteuerte Analyse aufgebauten Konzeptualisierungen muß dieses Ergebnis liefern können.

4.3.1.2 PTRANS

`ptrans` bezeichnet die Klasse der Aktionen, die Veränderung der topographischen Lage eines Objektes bewirken. Verben dieser Klasse sind geben, liefern, tragen etc., bei denen die physikalische Position des Objektes verändert wird. Oft wird die `ptrans`-Aktion nur impliziert, etwa durch eine Aktion der Klasse `propel`. Um Zusammenhänge dieser Art ausdrücken zu können, wurden in der Klassenhierarchie der Aktionen Querverweise eingeführt, die solche Beziehungen ausdrücken können. Die Klasse der `ptrans`-Aktionen kann ebenfalls in Unterklassen spezialisiert werden.

Interpretation

Die Konsequenzen einer `ptrans`-Aktion sind die Kenntnis des stattgefundenen Vorganges bei den Beteiligten. Die topographischen Koordinaten des Objektes der Aktion ändern sich.

4.3.1.3 PROPEL

Die Anwendung physischer Gewalt auf ein Objekt wird durch Aktionen der Klasse `propel` ausgedrückt. Die Verben "werfen", "schlagen", "stechen" etc. gehören zu dieser Klasse, d.h. sie haben die lexical view `LV-propel`. Innerhalb der Erwartungen zur Geschäftsbriefdomäne tritt diese Klasse nur mittelbar auf, nämlich als Prämisse von Verben der `ptrans`-Klasse wie "zusenden" etc..

Interpretation

Die Konsequenz einer `propel`-Aktion ist die Kenntnis des Vorganges. Die Folge kann eine `ptrans`-Aktion sein. `propel` geht oft eine andere Aktion voraus. Für "werfen", das zur lexical view `LV-propel` gehört, findet eine `grasp`-Aktion ihr Ende .

4.3.1.4 MOVE, INGEST, EXPEL, GRASP

move bezeichnet Aktionen der Bewegung eines (Körper-) Teiles eines Agenten durch diesen selbst. Dies kann in übertragener Bedeutung auch in der Domäne der Geschäftsbriefe verwandt werden. Zum Beispiel kann das Schicken des Kundendienstes durch eine Firma als eine move-Aktion verstanden werden, deren Objekt "Kundendienst" ein Bestandteil des Agenten Firma ist.

ingest und expel bezeichnen Aktionenklassen der Vereinnahmung bzw. Ausstoßung von Objekten in oder aus dem Körper des Agenten. Gewöhnlich ist die Konsequenz einer ingest-Aktion ein expel, d.h. eingenommene Objekte werden wieder ausgestoßen. Neben dem semantischen Constraint des Aktionenslots wird für den Objektslot die gefordert, daß das Objekt der Aktion ein konkretes ist. Das bedeutet, daß der semantic-constraints-Aspekt des Objektslots die Einhaltung einer lexical view LV-concrete-object ist.

```
(defclass ingest-action (...)
   ((action :initform (make-instance 'action-slot
            :semantic-constraint LV-ingest))
    (object :initform (make-instance 'object-slot
            :semantic-constraint LV-concrete-object)
   ...
   )
)
```

Zudem wird die Ausgangsrichtung bei expel und die Zielrichtung bei ingest auf den Agenten festgelegt. Eine Verwendung dieser Aktionenklasse im Nachrichtenmodell für Geschäftsbriefdokumente ist allerdings unwahrscheinlich, zumindest für die Unterdomäne DFKI. Innerhalb der Unterdomäne Schokoladenfabrik-Geschäftsbriefe dagegen können Erwartungen über Aktionen der ingest-Klasse sinnvoll sein.

Zur Klasse grasp gehören die Verben oder Verbphrasen, die die Aktionen des Greifens eines Gegenstandes durch einen Agenten bezeichnen. Dazu muß der gegriffene Gegenstand konkret sein, d.h. der semantische Constraint des Objektslots hat den gleichen Wert wie im obigen Beispiel. Die Zuordnung des Verbs greifen im Satz „Hagenbuch greift den Gedanken auf. " zur grasp-Klasse ist damit unzulässig, die Interpretation als atrans oder mtrans schon plausibler.

Interpretation

Die Konsequenzen von Aktionen dieser Klassen sind wie stets die Kenntnis des Agenten von der Aktion und die Veränderung der Position des Objektes. Diese ist, außer für expel, als mit der des Agenten gleich anzusehen.

4.3.1.5 MTRANS, MBUILD, ATTEND

Diese Aktionen-Klassen dienen zur Repräsentation kognitiver Vorgänge, die in einer sprachlichen Äußerung beschrieben oder die Konsequenz einer der übrigen Aktionen sind.

In Briefen ist oft die Rede von zukünftigen oder bereits stattgefundenen Ereignissen (`ptrans`- oder `atrans`-Klassen), die als Information übermittelt werden. Diese Informationsübertragung stellt eine `mtrans`-Aktion dar. Andere Möglichkeiten der Informationsübertragung bezeichnen die Klassen `speak` und `write`. Deren Objekte müssen akustische (insbesondere gesprochene) bzw. graphische (geschriebene) Äußerungen als Objekt haben.

Mit `mbuild` wird die mögliche Konsequenz-Aktion, die Inferenz von neuem Wissen aus altem durch einen intelligenten Agenten, bezeichnet. Die Klasse `attend` umfaßt Bedeutungen der Ausrichtung eines Sinnesorgans auf eine entsprechende Quelle (hören, sehen etc.). Der gemeinsame Constraint aller mentalen Aktionen ist die Forderung nach einem intelligenten Agenten, bzw. einem Agenten mit intellektuellen Fähigkeiten.

Diese Klassen spielen alle keine Rolle für die Erwartungen, die im Nachrichtenmodell aufgestellt werden. Die im Brief genannten Aktionen werden also stets der entsprechenden Klasse zugeordnet, die Konsequenz der Informationsübertragung als `mtrans` wird nicht beachtet. Daher wird hier auch keine mögliche Interpretation angegeben (siehe dazu [Schank 72]).

4.3.2 Aktionen der Domäne

Bisher wurden die Aktionenklassen vorgestellt, die durch einfache CD-Formen mit den entsprechenden Constraints darstellbar sind. Innerhalb der Domäne gibt es typische Aktionen, die dort gehäuft auftauchen, aber nicht ohne weiteres mit den Primitiven der CD dargestellt werden können. Zur Lösung dieses Problems wurden einige neue Primitive eingeführt, die Primate. Primate sind im wesentlichen Spezialisierungen der Primitive, die um neue Slots erweitert wurden. Die für die Welt der Geschäftsbriefe typischen Aktionen, die durch einfache aktive CD-Formen beschrieben werden können, werden ebenfalls in diesem Kapitel aufgeführt.

Anfragen

Aktionen der Klasse `inquire-act` haben besondere Constraints für das Objekt der Anfrage, z.B. Termine, Preislisten etc. Diese Aktion enthält die gleichen Slots wie eine einfache aktive CD-Form.

Bestellen_allgemein

Die besonderen Constraints, die zur Erwartungshaltung einer Bestellung (`order-act`) gehören, schränken die möglichen Agenten, Objekte, Richtungen und eventuell auch die Zeit

ein. Agenten müssen dem semantischen Constraint `LV-customer` genügen, die Objekte `LV-product`, die Ausgangsrichtung muß mit der Topographie des Agenten konsistent sein, also von diesem ausgehen.

Bestellung_ändern

Die Änderung einer Bestellung muß einen Bezug auf die zu ändernde Bestellung enthalten. Dieser wird in einem neuen Slot (`reference`-Slot) der Ändern-Aktion `change-order-act` eingetragen. Dieser referiert auf einen anderen Nachrichtentyp. Die Konsequenz dieser Aktion kann eine Bestellen-Aktion oder eine Stornieren-Aktion sein.

Bestellung_bestätigen

Die Erwartung des Bestätigens einer Bestellung wird durch die Angabe eines semantischen Constraints `LV-order-acknowledge` für den Aktionen-Slot von `acknowledge-order-act` angeben. Innerhalb der Bestätigung muß es eine Referenz auf die zugehörige Bestellung geben, in welcher Form auch immer. Diese wird im `reference-slot` der Bestätigen-Aktion eingetragen.

Stornieren

Das Primat zur Stornierung einer Bestellung muß, wie in der Bestelländerung, einen Bezug auf die Bestellung enthalten. Dieser wird im `reference`-Slot der Stornieren-Aktion `cancel-order-act` eingetragen.

Liefern

Die Aktionenklasse des Lieferns erhält den zusätzlichen Constraint, daß der Agent der Aktion ein Lieferant, das Objekt ein Produkt sein muß. Um die Erkärung etwas einsichtiger zu machen, seien an dieser Stelle die Definitionen angegeben. Die Aktion Liefern wird durch die einfache aktive CD-Form mit den Slots `action`, `agent`, `object`, `direction_from`, `direction_to` und `date` beschrieben. Die Belegung des `action`-Slots erfolgt durch eine Instanz der Klasse `action-slot`, deren Definition folgende Form hat (hier unwichtige Werte wurden nicht angegeben):

```
(defclass supplie-action (...)
   ((action :initform (make-instance 'action-slot
            :semantic-constraint LV-supplie))
    (agent :initform (make-instance 'agent-slot
            :semantic-constraint LV-supplier))
    (object :initform (make-instance 'object-slot
            :semantic-constraint LV-product)
   )
)
```

In der Instanz wird der `semantic-constraint` mit `LV-supplie` belegt. Für den Agentenslot wird die Klasse `agent-slot` benutzt, die im `semantic-constraints`-Aspekt den Wert

LV-supplier erhält. Analog wird im entsprechenden Aspekt der Klasse object-slot für das Produkt der Wert LV-product angegeben.

Kaufen

Die Aktionen-Klasse des Kaufens kann mit einer einfachen aktiven CD-Form nicht dargestellt werden. Um die Erwartungshaltung einer Kaufen-Aktion in dem Nachrichtenmodell zu beschreiben, wird eine erweiterte CD-Form eingeführt. Sie enthält die Slots:

action
agent
object
quantity
price
direction_from
direction_to
date

action2
agent2
object2
price2
date2
direction_from2
direction_to2

Diese CD-Form zur Repräsentation des Kaufens enthält als semantischen Constraint des action-Slots die lexical view LV-buy. Den neuen Slots werden durch entsprechende Constraints die Bedeutungen des Bezahlens, des Käufers, des Zahlungsmittels, des Betrages und des Zahlungsdatums für action2, agent2, object2, price2 und date2 zugewiesen. Für diese spezielle Aktion gelten Regeln, die z.B. sicherstellen, daß die zweiten Richtungsangaben bis auf das Vertauschen von Quelle und Ziel gleich sind, das Datum date2 auf date folgt und insbesondere die Richtungsangabe direction_to mit den topographischen Daten des Kunden übereinstimmen muß, wie auch die direction_to2 mit denen des Verkäufers. Diese Regeln werden auf die in Kapitel 5.1.2.4 angegebene Weise notiert und müssen durch den Predictor getestet werden.

4.4 Zusammenfassung

Mit der Conceptual Dependency und den vorgestellten Erweiterungen steht nun die Basis, mit der das Nachrichtenmodell definiert werden kann. Mit der Definition statischer CD-Formen werden die Erwartungen von Elementen eines Briefes, wie z.B. der Adresse oder dem Datum, beschrieben. Mit den aktiven CD-Formen werden die erwarteten Beschreibungen von Aktionen im Briefdokument aufgestellt. Aus diesen zwei Arten von CD-Formen werden die Nachrichtenelemente gebildet.

5 Das Nachrichtenmodell

In diesem Kapitel wird die Datenstruktur der Nachrichtentypen festgelegt. Die Nachrichtentypen enthalten das gesamte Domänenwissen über Klassen von Geschäftsbriefen, die *Informationseinheiten*. Die zusätzlichen Steuerungsdaten für die Analyse durch den Predictor werden in den *Steuerungseinheiten* angegeben.

Die elementaren syntaktischen Elemente, die CD-Formen, wurden bereits in Kapitel 4 erklärt. Hier werden die darauf aufbauenden Strukturen beschrieben.

Die den Nachrichtentypen zugeordnete Interpretation bestimmt die Semantik, die Möglichkeiten der (und für die) Analyse und das Ausmaß an modellierbaren Vorgängen. Die Semantik der Nachrichtentypen, insbesondere der Steuerungseinheiten, bestimmt im wesentlichen den Analyseablauf.

In diesem Kapitel wird die Syntax der Nachrichtentypen vorgestellt. Dabei werden zunächst die Bestandteile vorgestellt, die Informationen aus dem Text aufnehmen sollen, die *Informationseinheiten*. Im Anschluß werden die zur Steuerung der Analyse nötigen *Steuerungseinheiten* vorgestellt.

5.1 Syntax der Nachrichtentypen

Das Nachrichtenmodell ist als eine hierarchische Klassenstruktur organisiert. Sie fußt in den CD-Slots, aus denen *Message Elements* (Nachrichtenelemente) in Conceptual Dependency-Notation konstruiert werden. Mehrere dieser Elemente bilden einen *Message Type* (Nachrichtentyp), zusammen mit anderen entsteht daraus das Konstrukt eines *Multi-Message Types* (Multi-Nachrichtentyp)[6].

6 Die englischsprachigen und deutschsprachigen Begriffe werden im Text synonym verwandt.

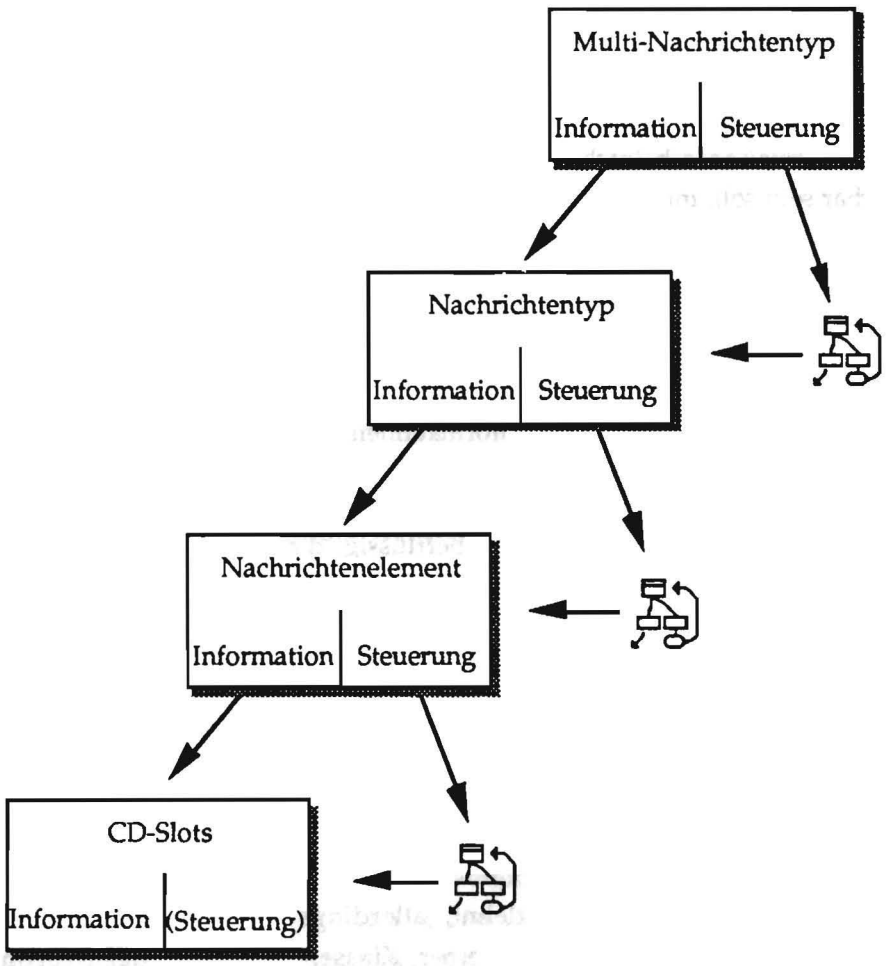

Abbildung 7: Der Aufbau des Nachrichtenmodells

Dabei wird hier zwischen solchen Einheiten unterschieden, die dazu bestimmt sind, im Text vorhandene Informationen aufzunehmen (*Informationseinheiten*), und jenen, die dem Predictor zur Unterstützung und als Direktive der Steuerung dienen (*Steuerungseinheiten*). Die Informationseinheiten der Multi-Nachrichtentypen sind in den Nachrichtentypen enthalten, deren Informationseinheiten in den Nachrichtenelementen, die durch einige Aspekte der CD-Slots bestimmt werden. Die Steuerungseinheiten in einem Element der Hierarchie bestimmen, wie in der Analyse das nächste untergeordnete Element aufgebaut wird. Die Steuerung in den CD-Slots selbst ist implizit vorgegeben.

Im folgenden Kapitel 5.1.1 werden zunächst die informationstragenden Elemente des Nachrichtenmodells vorgestellt. Anschließend wird in Kapitel 5.1.2 auf die Steuerungseinheiten eingegangen, die, sieht man den Predictor in der Rolle eines Interpreters der Nachrichtentypen, als das Programm aufgefaßt werden können.

5.1.1 Syntax der Informationseinheiten

In diesem Abschnitt soll die Datenstruktur der Nachrichtentypen nur als Behälter für im Text vorhandene Informationen betrachtet werden. Jede Informationseinheit des Dokumentes, die repräsentierbar sein soll, muß als Element des Nachrichtentyps definiert werden.

Informationen im Text, für die hier kein Behältnis vorliegt, können daher vom Analysesystem nicht erkannt werden und gehen verloren. Zur Bestimmung der Elemente, die einem Nachrichtentypen zugehören sollen, muß daher der Definition eine genaue Analyse der Inhalte von Dokumenten dieser Klasse vorangegangen sein. Es muß völlig klar sein:

- Welche (explizite und implizite) Informationen sind in einem Dokument enthalten?
- Wie können diese Informationen später genutzt werden?
- Welche dieser Informationen sind überflüssig, d.h. im Rahmen der pragmatischen Ansprüche der Analyse belanglos ?

5.1.1.1 Überblick

Aus den CD-Slots, die unter anderem dazu dienen, die im Text gefundene Information aufzunehmen, bauen sich die *Nachrichtenelemente* auf. Die CD-Slots sind von ihrer Syntax (und auch der Namensgebung) und der damit suggerierten Semantik der Conceptual Dependency-Notation entlehnt, allerdings sind vielfältige Erweiterungen vorgesehen. So sind CD-Slots Objekte einer Klasse, die neben der Wortinformation zusätzliche Werte, insbesondere Constraints, aufnehmen können; siehe dazu auch Kapitel 4. Mehrere Slots bilden eine Konzeptualisierung oder CD-Form, die die Nachrichtenelemente darstellen. Mit diesen Bausteinen (etwa Elemente zur Repräsentation einer Adresse, des Datums oder einer Bestellen-Aktion) werden die Nachrichtentypen konstruiert.

Zur Modellierung von Vorgängen und der Erweiterung der Menge von verstehbaren Briefen - insbesondere solcher, die sich auf zeitlich vorangegangene Dokumente beziehen oder mehr als einen Nachrichtentyp enthalten - dient die Datenstruktur der *Multi-Nachrichtentypen*. Sie sind die Inkarnation des Bezugs zwischen Nachrichtentypen, der durch die Slots zur impliziten Aktivierung (s.u.) aufgebaut wird.

5.1.1.2 Informationseinheiten in CD-Slots

Die elementaren Slots zur Darstellung aktiver und statischer CD-Formen wurden bereits in Kapitel 4 vorgestellt. Es sind

```
action
agent
object
direction_from
direction_to
date
```

für aktive CD-Formen und

```
object
owner
is-a
properties
```

für statische CD-Formen. Alle Slots werden mit Instanzen von `cd-slot` oder einer Spezialisierung dieser Klasse belegt. Der `properties`-Slot wird mit einer Liste von Instanzen der Klasse `prop-class` belegt, die die Slots `state`, `value` und `measurement` enthalten. Diese werden durch Instanzen von `cd-slot` oder einer Spezialisierung dieser Klasse belegt. Damit haben also alle Slots im wesentlichen die gleiche Grundstruktur der Aspekte.

Struktur der CD-Slots: Aspekte von Slots

Die Klasse `cd-slot` enthält direkt oder durch Vererbung von `cd-constraints` die Aspekte `phrase-found`, `need-substantiator`, `expected-locations`, `syntactic-constraints` und `semantic-constraints`. Die Klassendefinition hat die bereits in 4.1 angegebene Gestalt:

```
(defclass cd-slot (cd-constraints)
   ((phrase-found
      :type list
      :initform '()
      :accessor phrase-found)
    (syntactic-constraints
      :type list
      :initform '()
      :reader syntactic-constraints)
    (semantic-constraints
      :type list :initform '()
      :reader semantic-constraints)
   )
)
```

Spezialisierungen dieser Klasse, also Subklassen, werden nur durch Angabe anderer Constraints definiert: Die Klasse der Slots, die nur Phrasen aufnehmen soll, die Produkte bezeichnen, wird durch die spezialisierende Klassendefinition

```
(defclass product-slot (cd-slot)
   ((semantic-constraints
      :initform '(LV-product)
      :allocation :class)
    (syntactic-constraints
      :initform '(LV-NP)
      :allocation :class)
   )
)
```

beschrieben. Dabei wird natürlich vorausgesetzt, daß den Symbolen LV-PRODUCT und LV-NP durch das Lexikon eine Interpretation zugeordnet ist, die eine Auswertung durch den Predictor bzw. die Substantiierer ermöglicht. Die Informationseinheiten finden sich in den Slots der Klasse cd-slots.

5.1.1.3 Informationseinheiten in Nachrichtenelementen

Die Bestandteile, in die sich ein Geschäftsbrief sinnvoll zerlegen läßt, werden durch Nachrichtenelemente beschrieben. Dies sind aktive oder statische CD-Formen, die um zusätzliche Slots erweitert sein können. Typische Bestandteile, die als Nachrichtenelement definiert werden, sind z.B. die Adresse und das Datum (als statische CD-Formen) oder eine Bestellung ("Wir bestellen...") im Briefrumpf (als aktive CD-Form). Die in Kapitel 4 ausführlich definierten CD-Formen werden hier kurz wiederholt:

Aktive CD-Formen

Eine aktive CD-Form besteht aus den obligatorischen Slots, die schon im obigen Beispiel erwähnt wurden:

```
action   :type action-slot
         :initform (make-instance 'action-Slot)
agent    :type agent-slot
         :initform (make-instance 'agent-Slot)
object   :type object-slot
         :initform (make-instance 'object-Slot)
```

Dabei wird die Menge zulässiger Rollenfüller zweifach eingeschränkt: Zum einen durch das :type-Konstrukt, welches die zugehörige cd-slot-Klasse angibt, zum anderen durch die in dieser Klasse als Aspekte angegebenen Constraints. Diese dienen zur Überwachung, damit tatsächlich nur die gemäß dem Lexikon als Agenten markierten Worte als Füller akzeptiert werden. Diese Worte müssen der lexical view direkt oder durch Vererbung angehören.

Ergänzend sind als weitere grundlegende Slots zur Repräsentation der Richtungen und Zeit einer Aktion

```
direction_from
direction_to
date
```

vorgesehen, die mit Instanzen ihrer assoziierten `cd-slot`-Klassen belegt werden. In Kapitel 4 wurden Erweiterungen der einfachen aktiven CD-Formen um weitere Slots definiert. Die Informationseinheiten aller aktiven CD-Formen sind in den Aspekten der Klasse `cd-slots` enthalten, die Oberklasse aller zulässigen Belegungen der Slots aktiver CD-Formen ist.

Statische CD-Formen

Statische CD-Formen werden durch die Slots `object`, `owner` und `is-a` und `properties` definiert. Dabei sind die Klassen, deren Instanzen die Werte der Slots bilden, so zu definieren, daß die folgenden Belegungen möglich sind:

- `object`: Objekte (bzw. Instanzen von Objekten) gemäß Lexikon, etwa: Produkte, Kunden, Mitarbeiter. Dies wird z.B. durch den semantischen Constraint `LV-object` erreicht.
- `owner`: andere Objekte, hiermit wird eine `is-part-of`-Beziehung ausgedrückt.
- `is-a`: die Superklasse, zu der das angegebene Objekt gehört.
- `properties`: in dieser Liste wird eine unstrukturierte Menge von Eigenschaften des Objektes als Instanzen der Klasse `prop-class` definiert Diese enthält die Slots:
  - `state`: alles, was gemäß Lexikon einen Zustand direkt bezeichnet, bzw. sich auf einen Zustand bezieht (z.B. Gewicht).
  - `value`: der Wert, den der angegebene Zustand erhält (z.B. 52). Dieser erhält durch `measurement` eine Maßeinheit.
  - `measurement`: die (physikalische) Maßeinheit obiger Größe (z.B. [kg]).

Zweckmäßigerweise ist die Notation der Objektstruktur durch diese vier Slots nicht mit der tatsächlichen Repräsentation der Objekte in einer Hierarchie identisch, sie muß also konvertiert werden.

Nachrichtenelemente

Die Nachrichtenelemente sind CD-Formen oder Erweiterungen der CD-Formen um Slots, die für die Domäne der Geschäftsbriefe nötig sind. Diese sind z.B. für eine Aktion des Bestellens die Slots `price` und `quantity`, die zur Aufnahme der Preis- und Mengenangaben dienen.

5.1.1.4 Informationseinheiten in Nachrichtentypen

Nachrichtentypen werden aus einer Menge von Nachrichtenelementen zusammengestellt. Die Elemente bilden also die Informationseinheiten, mit denen alle Typen aufgebaut werden sollen. Die Menge der Nachrichtentypen läßt sich durch diesen modularen Aufbau als *Nachrichtenbaum* darstellen. Diese Bezeichnung ist nicht ganz korrekt, die Struktur ist eher ein gerichteter Graph, den allerdings nur wenig von einem Baum unterscheidet. Abbildung 8 zeigt die Struktur dieses Baumes.

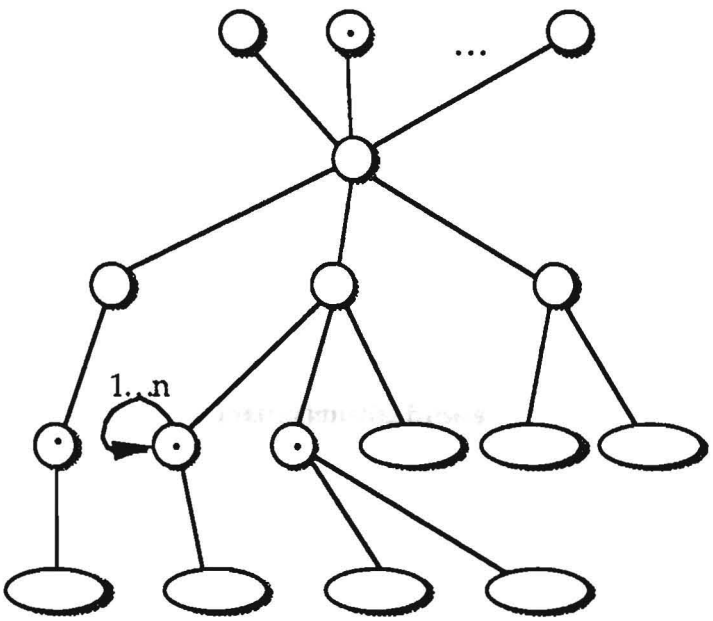

Abbildung 8: Die Nachrichtentypen in der Sicht der Definition[7]

Die Wurzelknoten des Nachrichtenbaumes bilden die allgemeinsten Nachrichtenelemente, die in allen Nachrichtentypen vorkommen, z.B. Adressen, Datum etc.. An den Blattknoten sind die genau einen Nachrichtentyp charakterisierenden Elemente enthalten, hier ist das Einzigartige eines Nachrichtentyp definiert. Die Gesamtheit der Nachrichtenelemente entlang eines Pfades (genauer: eines Astes), von den Wurzeln zum Blatt, definiert einen Nachrichtentyp.

Nachrichtentypen werden als Klassen definiert, die mehrere Nachrichtenelemente, die Bausteine des Typen, enthalten. Die Schreibweise des Nachrichtentyps einer Bestellung hat damit als Klassendefinition die Form:

7 Elemente, die mit einem • markiert sind, haben für die Analyse eine ausgezeichnete Bedeutung.

```
(defclass MT-order (...)
   ((me-sender     :initform (make-instance 'ME-sender)
                   :accessor me-sender)
    (me-recipient :initform (make-instance 'ME-recipient)
                   :accessor me-recipient
    (me-date       :initform (make-instance 'ME-date)
                   :accessor me-date)
    (me-order      :initform (make-instance 'ME-order
                   :accessor me-order)
   )
)
```

Diese Definition legt die Klasse des Nachrichtentyps `MT-order` für Bestellungen durch die Angabe seiner Elemente fest. Dies sind im Beispiel die Nachrichtenelemente Sender, Empfänger, Datum und bestellen. Da aber das Element `ME-order` nur einen Bestellvorgang repräsentieren kann, muß zur Realisierung von Bestellungen mehrerer Artikel dieses Nachrichtenelement mehrfach im Nachrichtentypen auftauchen.

Dazu wird die Definition des Nachrichtentyps um den ausgezeichneten Slot `list-<message-element>` erweitert, der die Liste der Nachrichtenelemente vom Typ `<message-element>` aufnimmt. Damit sieht die Definition von `MT-order` nun so aus:

```
(defclass MT-order (...)
   ((me-sender     :initform (make-instance 'ME-sender)
                   :accessor me-sender)
    (me-recipient :initform (make-instance 'ME-recipient)
                   :accessor me-recipient
    (me-date       :initform (make-instance 'ME-date)
                   :accessor me-date)
    (list-me-order:initform '(1 42 (make-instance 'ME-order)))
                   :accessor list-me-order)
   )
)
```

Mit der Initialbelegung `initform` werden Angaben über die Länge der Liste gemacht. Diese werden vom Predictor zur Fehlererkennung benutzt. Im Beispiel darf die Liste der Bestellungen bis zu 42 Positionen enthalten, aber nicht leer sein. Für die Sicht der Nachrichtentypen als Pfad im Nachrichtenbaum bedeutet die Erweiterung, daß bestimmte Knoten mehrfach besucht werden dürfen (siehe Abbildung 8). Die Wiederholung wird als eine Eigenschaft des Pfades, nicht des Baumes angesehen. Dadurch bleibt die Struktur des Nachrichtenbaumes, die für die Vorgehensweise des Predictors, insbesondere in der Startphase, einige Vorteile hat, unverändert.

5.1.1.5 Informationseinheiten in Multi-Nachrichtentypen

Mit dem Konstrukt der Multi-Nachrichtentypen werden semantische Beziehungen zwischen einzelnen Nachrichtentypen dargestellt. Multi-Nachrichtentypen repräsentieren das Wissen, daß bestimmte Nachrichtentypen innerhalb eines Kontextes, nicht notwendigerweise eines Briefes, auftauchen können. Ein Multi-Nachrichtentyp zur Darstellung eines kompletten Bestellen/Liefern-Vorganges `MMT-order-process` besteht aus mindestens einem der Nachrichtentypen `MT-advertising`, `MT-offer`, `MT-order`, `MT-change-order`, `MT-deliver` und `MT-invoice`.

```
(defclass MMT-order-process ()
   ((advertising     :type MT-advertising)
    (offering        :type MT-offer)
    (ordering        :type MT-order)
    (changingorder   :type MT-change-order)
    (delivering      :type MT-deliver)
    (invoicing       :type MT-invoice)
   )
)
```

Inhaltlich wird ein Multi-Nachrichtentyp durch die impliziten Aktivierungsinformationen, eine spezielle Steuerungsinformation seiner Nachrichtentypen aufgebaut. In Abbildung 9 ist ein Multi-Nachrichtentyp aufgezeichnet, der aus mehreren Nachrichtentypen besteht, die wiederholt oder alternativ auftauchen können. Jeder implizite Verweis bildet eine Kante eines Multi-Nachrichtentyps, die Knoten enthalten Zeiger auf die Nachrichtentypen und Steuerungsinformationen, die in Kapitel 5.1.2.5 erläutert werden.

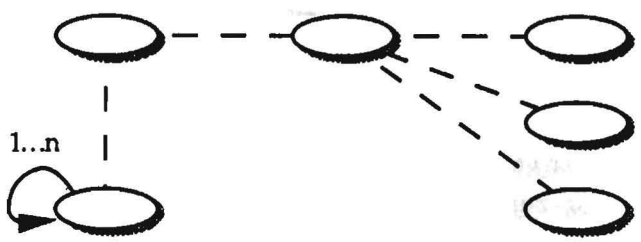

Abbildung 9: Ein Multi-Nachrichtentyp

Wiederholte zulässige Elemente werden syntaktisch wie die Wiederholung von Nachrichtenelementen als Liste definiert, z.B. `(list-MT-order :initform '(1 2 (make-instance 'MT-order))`.

5.1.2 Syntax der Steuerungseinheiten

Die Steuerungseinheiten weisen den durch die Namenswahl mit suggerierter Semantik belegten CD-Slots, Nachrichtenelementen und Nachrichtentypen eine für den Predictor verständliche Semantik zu.

Für alle im Kapitel Informationseinheiten genannten Elemente gibt es die entsprechenden Steuerungs-Pendants, die innerhalb der jeweiligen Definition der Datenstruktur definiert werden. Die Steuerungseinheiten legen als Elemente der Nachrichtentypen die Wichtigkeit `importance` und die Reihenfolge `order` zum Füllen der Slots fest. `rule-default`, `rule-can` und `rule-must` geben Regeln an, die Standardbelegungen liefern oder optional bzw. obligatorisch angewendet werden, um Belegungen herzuleiten. Durch die Oberklasse `cd-constraints` können Angaben über den benötigten Substantiierer (`need-substantiator`) und die erwartete Position im Eingabedokument (`expected-location`) gemacht werden. Diese sieben Slots ergänzen damit die Constraints für zulässige Slot-Füller in den atomaren Bausteinen `cd-slot`.

Innerhalb der Nachrichtentypen werden drei zusätzliche Slots zur Art der Aktivierung (explizit, elementinduziert oder implizit) angegeben. Die Multi-Nachrichtentypen werden um Angaben erweitert, wie die Kombination ihrer Nachrichtentypen auszusehen hat, dies erfolgt durch die Angabe von obligatorischen `rule-must`-Regeln.

5.1.2.1 Überblick

`importance`-Einträge bezeichnen die Wichtigkeit eines Eintrages, die unter mehreren Aspekten bewertet werden kann. Für den Analysefortgang: ob und unter welchen Umständen ist er möglich? Für die Interpretation des Ergebnisses, falls dieser Eintrag fehlt: kann es noch sinngebend gedeutet werden? Mit `importance` werden die Slots von Nachrichtenelementen und Nachrichtenelemente in Nachrichtentypen bewertet. Für die Aspekte einzelner CD-Slots gibt es keine `importance`-Einträge.

Durch Angabe von `order` wird die Reihenfolge wie Vorhersagen generiert werden sollen vorgegeben. Davon ist i.w. die Reihenfolge innerhalb eines Nachrichtenelementes betroffen, genauer die Reihenfolge innerhalb einer logischen Gruppe mehrerer Nachrichtenelemente. Außerhalb dieser Gruppe erfolgt die Analyse in der Reihenfolge der ersten Aktivierung, also vom Dokument abhängig. Die `order`-Angabe taucht in Nachrichtenelementen und Nachrichtentypen auf.

Regeleinträge stellen drei verschiedene Möglichkeiten bereit. Zum einen geben sie den Standardwert eines Slots an, wenn dieser aus dem Text nicht gefüllt werden konnte. Sie geben Alternativen zum Füllen der Slots an, falls der direkte Weg über einen Substantiierer keinen Erfolg gebracht hat, z.B. durch Angabe eines anderen Slots. Sie können aber auch Verbindungen zwischen Einträgen aufbauen, die das testen von Constraints beinhalten. Regeln werden zwischen den Slots eines Nachrichtenelementes, den Slots verschiedener

Nachrichtenelemente eines Nachrichtentyps, Nachrichtenelementen eines Nachrichtentyps und in Multi-Nachrichtentypen zwischen Nachrichtentypen, Nachrichtenelementen und Slots aufgestellt.

Zur Aktivierung von Nachrichtentypen, d.h. zur Entscheidung des Predictors für eine Erwartungshaltung, enthalten diese die ausgezeichneten Slots zur expliziten, elementinduzierten und impliziten Aktivierung.

Die Steuerungseinheiten können sich auf Slots, Nachrichtenelemente oder Nachrichtentypen durch die Angabe der jeweiligen Zugriffsfunktionen beziehen. Im Nachrichtenmodell wird gefordert, daß die Zugriffsfunktion mit dem Slot-Namen identisch ist, der referiert wird.

Innerhalb der CD-Slots werden keinerlei weitere Steuerungseinheiten angegeben, die Bearbeitung der Aspekte verläuft nach einer im Predictor festgelegten Reihenfolge und Wichtigkeit statt. Diese wird in [Gores 92] unter der Diskussion der Interpretation der Nachrichtentypen motiviert.

5.1.2.2 `order` und `importance`

Der Sinn der Einträge zur Reihenfolge der Abarbeitung liegt darin, der Analysekomponente Hinweise zu geben, welche der möglichen Erwartungen die nächste erfüllbare sein könnte. Sie spiegelt also eine Reihenfolge des Auftretens der Füller für Slots in Nachrichtenelementen, bzw. der Nachrichtenelemente in einem Nachrichtentyp wieder. Die Bewertung der Wichtigkeit wird benötigt, um ein Scheitern der Analyse beurteilen zu können.

Nachrichtenelemente

Für die statischen und aktiven CD-Formen der Nachrichtenelemente läßt sich jeweils eine standardisierte Steuerung angeben, die für alle gleichermaßen sinnvoll ist. Dazu wird eine Oberklasse definiert, die nur aus diesen Steuerungseinheiten besteht, und sie an alle Unterklassen vererbt. Für die Bewertung der Reihenfolge wird davon ausgegangen, daß die Aktionen die zentrale Rolle spielen, gefolgt von Agenten und Objekten. Je nach der Strukturierung der Domäne kann diese Prioritätenreihenfolge in anderer Ordnung sinnvoller sein, wobei dem am stärksten einschränkenden Slot die höchste Priorität zugewiesen wird. Die Notation dieser Standardreihenfolge hat für aktive CD-Formen die Syntax

```
(order :initform  '((action.1) (agent.2) (object.3)
                    (direction_from.4) (direction_to.4) (date.5)
                    )
)
```

und für statische

```
(order :initform '((object.1) (owner.2) (is-a.2) (properties.2)))
```

Der Liste der Eigenschaften `properties` kann keine initiale Reihenfolge und Bedeutung zugeordnet werden.

Auf ähnliche Weise wird die Standard-`importance` der Slots angegeben. Dabei bezeichnen, wie auch für die Reihenfolge, kleinere Zahlen die höhere Ordnung:

```
(importance :initform  '((action.1) (agent.1) (object.2)))
```

und

```
(importance :initform  '((object.1) (owner.3) (is-a.3)))
```

Während sich die Reihenfolgeangabe `order` über einen Bereich der Größenordnung 1 bis n erstrecken kann, wobei eine sinnvolle Obergrenze für n die maximale Anzahl von Slots in einem Nachrichtenelement ist (Größenordnung ca. 5-15), genügen für die Wichtigkeit `importance` drei[8] Werte, denen die Interpretationen von *sehr wichtig* bis *unwichtig* zugeordnet werden:

- *sehr wichtig*: ohne diese Information kann das übergeordnete Element (Nachrichtenelement) nicht partiell instantiiert werden, ihre Existenz ist unbedingt nötig.
- *wichtig*: das Fehlen dieser Information läßt eine partielle *Instantiierung* zu, und damit auch den Fortgang der Analyse auf dem gewählten Weg. Liegt diese Information vor, begünstigt dies die Analyse.
- *unwichtig*: Elemente dieser Art stellen weder durch ihr Fehlen, noch durch ihre Anwesenheit die erfolgreiche Instantiierung in Frage, bestenfalls dienen sie dazu, ergänzende Daten aus dem Brief zu extrahieren, mit der die Konzeptualisierung ausdrucksstärker werden kann.

Nachrichtentypen

In den Nachrichtentypen wird durch `order` die erwartete Reihenfolge der Nachrichtenelemente angegeben. Dies ist eine Information, die sich von der Angabe der Layoutinformation in den Aspekten der Slots der Nachrichtenelemente unterscheidet, sie gibt lediglich Hinweise auf die Abarbeitungsreihenfolge. Die Notation der Ordnung von Nachrichtenelementen erfolgt durch den Slot `order` im Nachrichtentyp, der für eine Bestellung die Belegung

```
(order :initform  '((list-me-order.1) (me-sender.1) (me-recipient.1)
                    (me-date.1))
)
```

[8] Eine subtilere Differenzierung mag mit den wachsenden Fähigkeiten des Predictors, auf Ausnahmen zu reagieren, einhergehen.

haben könnte. Auf analoge Weise wird die Wichtigkeit angegeben.

Multi-Nachrichtentypen

In Multi-Nachrichtentypen gibt es keine Angaben für `order` oder `importance`. Diese Angaben machen hier keinen Sinn.

5.1.2.3 Regeln

Das Nachrichtenmodell enthält drei verschiedene Arten von Regeln. Sie können in Multi-Nachrichtentypen, Nachrichtentypen oder Nachrichtenelementen definiert werden. Es sind dies `rule-default`, `rule-can` und `rule-must`. Erstere legen einen Defaultwert fest, der verwendet wird, wenn kein anderer Wert vorliegt. Mit `rule-can` werden optionale Werte angeboten. Mit der letzten Regelart werden zwingende Bedingung aufgestellt, deren Verletzung der Analyse einen Fehler signalisiert. Die Syntax der Regeln ist

```
(rule-must
   '({(<LISP predicate>)}*)
)
```

bzw.

```
(rule-[must|can|default]
   '({(if (<LISP predicate>)
          <then>
          [<else>])}*
    )
)
```

Mit der ersten Schreibweise wird unabhängig von Nebenbedingungen eine durch das Lisp-Prädikat getestete Bedingung gefordert. Die zweite Schreibweise gibt obligatorische, optionale oder default- Regeln an, die abhängig von der Gültigkeit eines Prädikates eine Lisp-Form im `then` oder `else`-Zweig der Regel evaluieren. Für die Regeln gelten die folgenden Vorschriften:

- Eine Regelmenge besteht aus null oder mehr Regeln.
- Eine Regel wird durch das Prädikat (`rule-must`) oder durch ein Prädikat und seine Konsequenzen (`rule-must`, `-can` und `-default`) angegeben. Das Prädikat kann jedes Lisp-Prädikat sein[9].
- Als Argumente des Prädikates sind die im Kontext der Regel sichtbaren Bestandteile des Nachrichtenelementes, Nachrichtentyps oder Multi-Nachrichtentyps erlaubt.

---

9 Um die Analyse durch aufwendige Prädikate zu entlasten, werden nur einfache Prädikate (wie Vergleiche) benutzt.

- Die spezielle Variable `*self*` kann als Argument in Prädikaten ,`else` und `then`-Zweig benutzt werden. Sie bezeichnet die Instanz in der die Regel definiert ist.
- In den Prädikaten, sowie in `else` und `then`-Zweig, können die im Kontext der Regel sichtbaren Slots durch ihre Zugriffsfunktionen in dieser Instanz angesprochen werden.
- Der `then` und `else`-Teil einer Regel stellt eine LISP-Form dar, die der Predictor zum Zeitpunkt der Analyse durch Lisp evaluieren läßt.
- Im `then` bzw. `else`-Zweig wird die Wertebelegung eines Bestandteiles des die Regel enthaltenden Nachrichtenelementes, Nachrichtentyps oder Multi-Nachrichtentyps in einer Form angegeben (`setf`), die evaluiert werden kann und einen Schreibzugriff erlaubt. Die Form ist ebenso wie das testende Prädikat nicht beschränkt, sollte aber nicht rechenintensiv sein.

Die obigen Regeln können kombiniert werden.

Nachrichtenelemente

In den Nachrichtenelementen stellen die Regeln Anforderungen an Slots und deren Belegungen auf. Die als Argumente zulässigen Elemente sind damit die Slots dieses Nachrichtenelementes. In einem Nachrichtenelement wird mit der Regelmenge

```
(rule-default
   '( (if   (not (slot-boundp *self* direction_from))
          (setf (direction_from *self*) (agent *self*))
      )
    )
)
```

angegeben, daß der Slot `direction_from` als Default mit dem Wert des Agenten-Slots belegt wird, wenn er nicht bereits gebunden ist[10].

Nachrichtentypen

In den Nachrichtentypen werden die Regeln auf die gleiche Art beschrieben, da sie aber Beziehungen zwischen Nachrichtenelementen ausdrücken, sind die in den Prädikaten bzw. `then` und `else`-Zweigen zulässigen Argumente Informationseinheiten von Nachrichtentypen, also Nachrichtenelemente und deren Slots. Eine denkbare Muß-Regel in dem Nachrichtentyp Bestellung ist die Forderung, daß der Agent der Bestellung mit dem Absender identisch ist. Sie wird durch

[10] Dabei sind `last-name` und `agent` die Zugriffsfunktionen auf die gleichnamigen Slots.

```
(rule-must
   '( (eq   (last-name (*self* me-sender))
            (agent (*self* me-order))
      )
    )
)
```

ausgedrückt. Vereinfachend wird hierbei nur der Nachname der Absenderadresse gegen den `agent`-Eintrag getestet.

Multi-Nachrichtentypen

In Multi-Nachrichtentypen stellen Regeln die Beziehungen zwischen den beteiligten Nachrichtentypen auf. Sie dienen z.B. zum Test der Konsistenz, um festzustellen, ob ein Nachrichtentyp zu einem Multi-Nachrichtentyp paßt. Hierzu muß die Belegung von Elementen getestet werden, z.B. ob die Datumsangaben (`me-date`) eine aufsteigende Reihenfolge haben.

5.1.2.4 Aktivierungen

Ausschließlich in den Nachrichtentypen werden als Sonderfall der Steuerinformation Schlüsselelemente benannt, denen eine besondere Bedeutung für die Aktivierung zukommt. Sie werden in den Aktivierungs-Slots `explicit-word-reference`, `primary-list`, `secondary-list`, `tertiary-list`, `explicit-logical-object-reference`, `explicit-layout-reference`, `implicit-reference`, `element-induced-activation`, `logical-object-induced-activation`, `layout-induced-activation` aufgeführt. Die Slots zur Aktivierung stellen die Information bereit, die es dem Predictor zu Beginn der Analyse ermöglicht, sich effizient und schnell für einen Nachrichtentyp zu entscheiden und dessen Erwartungen durch die Substantiierer zu beweisen.

`explicit-word-reference`

In dem Slot `explicit-word-reference` eines Nachrichtentyps werden in einer Liste die Worte angegeben, die eine so starke Semantik haben, daß ihr Auftauchen das Vorliegen dieses Nachrichtentyps anzeigt. Sie werden durch eine lexical view und den Slot eines Nachrichtenelementes des Nachrichtentyps beschrieben, der das Schlüsselwort aufnimmt. Die ausgezeichnete Rolle einer Phrase der lexical view `LV-order` als Aktivierungsinformation des Nachrichtentyps `MT-order` wird durch

```
(explicit-word-reference '( ((action LV-order) ME-order)...))
```

beschrieben. Der von der Aktivierung betroffene Eintrag ist der `action`-Slot des Nachrichtenelementes `me-order`.

primary-list, secondary-list, tertiary-list

In diesen drei Slots werden die zu einem Nachrichtentyp bekannten Listen der statistisch wichtigen Worte eines Nachrichtentyps als lexical views gespeichert. Die Primärliste enthält die Menge der wichtigsten Worte, die Sekundär- und Tertiärliste die zweit- bzw. drittwichtigsten Worte. Diese Listen werden für die Klassifizierung von Briefdokumenten durch das InfoClas-System ([Dittrich 92]) benötigt.

explicit-logical-object-reference, explicit-layout-reference, logical-object-induced-activation, layout-induced-activation

Diese Slots sind für die Angabe von logischen Objekten bzw. Layoutobjekten reserviert, deren Auftreten direkt auf den Nachrichtentyp verweist. Sie stellen auf einer anderen Ebene das Pendant zur expliziten und elementinduzierten Aktivierung durch Worte dar. Auf diese Slots wird bisher verzichtet.

implicit-reference

Mit der impliziten Referenz wird in einem Nachrichtentyp auf ihm verwandte verwiesen. Eine Verwandschaft zweier Nachrichtentypen besteht z.B. zwischen einer Bestellung und einer Bestelländerung: wenn ein Bestellbrief bearbeitet wird, ist es wahrscheinlich, daß darauf eine Bestelländerung folgt, der sich eventuell eine Stornierung anschließt.

In einer Liste werden die Namen der Nachrichtentypen genannt, deren Aktivierung wahrscheinlich ist. Für eine Bestellung sind dies zum Beispiel Bestelländerung (`MT-change-order`) und Stornierung (`MT-cancel`):

```
(implicit-reference '(MT-change-order MT-cancel))
```

Die Angaben zur impliziten Referenz legen damit auch die Struktur der Multi-Nachrichtentypen fest: Gibt es einen `implicit-reference`-Eintrag zwischen zwei Nachrichtentypen, so muß es auch einen Multi-Nachrichtentyp geben, der beide enthält.

event-induced-activation

Ähnlich wie der Slot `explicit-reference` dienen die hier spezifizierten Werte zur elementinduzierten Aktivierung als Verweis auf einen Nachrichtentyp. Sie sind aber nicht von einzelnen Worten abhängig, sondern von partiellen Konzeptualisierungen, also fragmentarischen Nachrichtenelementen. Die Angabe der expliziten Aktivierung durch eine lexical view kann zu Fehlaktivierungen führen, da z.B. viele Verben zur lexical view `LV-order` gehören. Erst die Zusatzbedingung, daß das Objekt des Verbs der lexical view `LV-product` angehören muß, macht eine Aktivierung sicherer. Durch die elementinduzierte Aktivierungsinformation

```
(event-induced-activation
   '( (  (action LV-order)
         (object LV-product)
         me-order
      )
      ...
   )
)
```

wird dies ausgedrückt. Eine elementinduzierte Aktivierung besteht aus einer Liste von Slots mit den geforderten lexical views und dem Nachrichtenelement, das von der Aktivierung betroffen ist. Die in den Slots zur elementinduzierten Aktivierung enthaltene Information wird in die Diskriminierungsbäume ([Gores 92]) kodiert.

5.2 Zusammenfassung

In diesem Kapitel wurde nun die Definition des Nachrichtenmodells angegeben. Zusammen mit der Kenntnis des Conceptual Dependency-Formalismus ist es nun möglich, die verschiedenen Klassen von Geschäftsbriefen, die in **ALV** analysiert werden sollen, zu modellieren. Dazu werden die Briefe in die sie bildenden Bestandteile zerlegt, die als Nachrichtenelemente definiert werden. Ein Nachrichtentyp, der den Inhalt einer bestimmten Dokumentklasse enthalten soll, wird durch Zusammensetzen der zugehörigen Nachrichtenelemente aufgebaut. Durch die Angabe von Constraints und Steuerinformationen wird so eine Struktur erzeugt, der die Semantik der modellierten Klasse zugewiesen werden soll. Mit dem Konstrukt der Multi-Nachrichtentypen ist zum einen die Modellierung von Briefen möglich, die mehr als einen Nachrichtentyp enthalten. Zum anderen kann damit eine Korrespondenz über mehrere Briefe als Erwartungshaltung aufgestellt, und damit vom Analysesystem verstanden werden.

6 Implementierung

In diesem Kapitel werden abschliessend einige Bestandteile des Nachrichtenmodells kommentiert vorgestellt, um eine Vorstellung von der Realisierung des Modells in dem Objektsystem der Programmiersprache Common Lisp zu geben.

Hilfsdefinitionen

Mit der Konstante `*all-control-slots*` werden die Slots in einer Liste angegeben, die keine Informationseinheiten enthalten. Diese Slotnamen sind ausgezeichnete Elemente der Syntax des Nachrichtenmodells, wohingegen die Slotnamen von Informationseinheiten frei benannt werden können.

```
(defconstant *all-control-slots*
   '( need-substantiator expected-locations
      phrase-found semantic-constraints syntactic-constraints
      importance order rule-default rule-can rule-must
      list-all-slots implicit-reference
      explicit-word-reference explicit-logical-object-reference
      explicit-layout-reference element-induced-activation
      logical-object-induced-activation layout-induced-activation)
)
```

Die Klasse `Contents-class` ist Oberklasse jeder Klasse eines Nachrichtenelementes, Nachrichtentyps und Multi-Nachrichtentyps. Sie vererbt somit den ausgezeichneten Slot `list-all-slots`, der, bis auf die Elemente von `*all-control-slots*`, die Namen aller Slots einer Klasse enthält.

```
(defclass Contents-class ()
   ((list-all-slots  :accessor list-all-slots))
)
```

Mit der Methode `initialize-instance` dieser Klasse wird somit erreicht, daß immer, wenn eine Instanz erzeugt wird, der Slot `list-all-slots` ebenfalls gefüllt wird.

```
(defmethod initialize-instance
   :after ((obj contents-class) &key)
   (setf
      (slot-value obj 'list-all-slots)
      (list (nset-difference
                (mapcar #'(lambda (one-slot)
                          (slot-value one-slot 'clos::name))
                          (slot-value (class-of obj) 'clos::slots)
                )
                *all-control-slots*)
      )
   )
)
```

Die Klasse `infoclas-class` ist Oberklasse jedes Nachrichtentyps. In ihr werden die drei Slots zur Aufnahme der Primär, Sekundär und Tertiärlisten definiert.

```
(defclass infoclas-class ()
   ((primary-list    :accessor primary-list
                     :initform '())
    (secondary-list  :accessor secondary-list
                     :initform '())
    (tertiary-list   :accessor tertiary-list
                     :initform '())
   )
)
```

Mit der Klasse `cd-constraints` werden die für Nachrichtenelemente und Slots von Nachrichtenelemehten wichtigen Aspekte definiert, die Substantiiererempfehlungen bzw. Layoutangaben enthalten.

```
(defclass cd-constraints (Contents-class)
   ((need-substantiator    :initform '(*General-Substantiator*)
                           :initarg :need-substantiator
                           :reader need-substantiator)
    (expected-locations    :initform nil
                           :initarg :expected-locations
                           :reader expected-locations)
   )
)
```

Die Klasse `cd-slot` stellt die Verfeinerung von `cd-constraints` dar, die für die Slots von Nachrichtenelementen die zusätzlichen Aspekte `phrase-found`, `semantic-constraints` und `syntactic-constraints` definiert

```
(defclass CD-slot (CD-constraints)
   ((phrase-found       :initform nil
                        :accessor phrase-found)
    (semantic-constraints  :initform nil
                           :initarg :semantic-constraints
                           :reader semantic-constraints)
    (syntactic-constraints :initform nil
                           :initarg :syntactic-constraints
                           :reader syntactic-constraints)
   )
)
```

Für die Slots von Nachrichtenelementen und Nachrichtenelemente von Nachrichtentypen wird in den Slots `importance` und `order` eine Wichtigkeits- bzw. Reihenfolgeinformation angegeben. Die Klasse `valuation` ist daher Oberklasse jedes Nachrichtenelementes und Nachrichtentyps.

```
(defclass valuation ()
   ((importance   :initform nil
                  :reader importance)
    (order        :initform nil
                  :reader order)
   )
)
```

Die Klasse `rules` als Oberklasse jedes Nachrichtenelementes, Nachrichtentyps und Multi-Nachrichtentyps enthält die drei Slots zur Angabe der Regeln, die die Analyse unterstützen sollen.

```
(defclass rules ()
   ((rule-default :initform nil
                  :reader rule-default)
    (rule-can     :initform nil
                  :reader rule-can)
    (rule-must       :initform nil
                  :reader rule-must)
   )
)
```

Durch die Klasse `connections`, die Oberklasse jedes Nachrichtentyps ist, werden die Slots zur Aktivierung und Festlegung der Beziehungen zwischen den Nachrichtentypen definiert.

```
(defclass connections ()
   ((implicite-ref      :initform nil
                        :reader implicite-ref)
    (explicit-word-ref      :initform nil
                            :reader explicit-word-ref)
    (explicit-logical-object-ref
                        :initform nil
                        :reader explicit-logical-object-ref)
    (explicit-layout-ref    :initform nil
                            :reader explicit-layout-ref)
    (element-ind-act    :initform nil
                        :reader element-ind-act)
    (logical-object-ind-act    :initform nil
                               :reader logical-object-ind-act)
    (layout-ind-act     :initform nil
                        :reader layout-ind-act)
   )
)
```

Das Nachrichtenelement `ME-sender`

Durch das Nachrichtenelement `ME-sender` werden die Erwartungen einer Senderadresse ausgedrückt. Die einzelnen Slots, die Spezialisierungen der Klasse `cd-slot` sind, werden hier nicht weiter beschrieben. Analog erfolgt die Definition des Nachrichtenelementes `ME-recipient`:

```
(defclass ME-sender
   (address CD-constraints valuation rules)
   ((importance :initform  '((first-name . 2) (last-name . 2)
                             (company . 2) (po-box-nr . 2)
                             (street-name . 2) (street-nr . 2)
                             (country . 2) (zip-code . 2)
                             (city-name . 1) (post-office . 2)))
    (order  :initform   '((first-name . 1) (last-name . 2)
                          (company . 3) (po-box-nr . 4)
                          (street-name . 5) (street-nr . 6)
                          (country . 7) (zip-code . 8)
                          (city-name . 9) (post-office . 10)))
    (expected-locations    :initform '((logical-object sender)))
    (company
      :initform (make-instance 'company-slot
                  :expected-locations '((logical-object sender))))
    (first-name
      :initform (make-instance 'first-name-slot
                  :expected-locations '((logical-object sender))))
    (last-name
      :initform (make-instance 'last-name-slot
                  :expected-locations '((logical-object sender))))
    (po-box-nr
      :initform (make-instance 'po-box-nr-slot
                  :expected-locations '((logical-object sender))))
    (street-name
```

```
      :initform (make-instance 'street-name-slot
                  :expected-locations '((logical-object sender))))
    (street-nr
      :initform (make-instance 'street-nr-slot
                  :expected-locations '((logical-object sender))))
    (country
      :initform (make-instance 'country-slot
                  :expected-locations '((logical-object sender))))
    (zip-code
      :initform (make-instance 'zip-code-slot
                  :expected-locations '((logical-object sender))))
    (city-name
      :initform (make-instance 'city-name-slot
                  :expected-locations '((logical-object sender))))
    (post-office
      :initform (make-instance 'post-office-slot
                  :expected-locations '((logical-object sender))))
  )
  (:documentation "CD-Form fuer den Absender")
)
```

Das Nachrichtenelement ME-recipient

```
(defclass ME-recipient
  (address CD-constraints valuation rules)
  ((importance
     :initform '((first-name . 2) (last-name . 2) (company . 2)
                 (po-box-nr . 2) (street-name . 2) (street-nr . 2)
                 (country . 2) (zip-code . 2) (city-name . 1)
                 (post-office . 2)))
   (order
     :initform '((first-name . 1) (last-name . 2) (company . 3)
                 (po-box-nr . 4) (street-name . 5) (street-nr . 6)
                 (country . 7) (zip-code . 8) (city-name . 9)
                 (post-office . 10)))
   (expected-locations    :initform '((logical-object recipient)))
   (company
     :initform (make-instance 'company-slot
                 :expected-locations '((logical-object
                                        recipient))))
   (first-name
     :initform (make-instance 'first-name-slot
                 :expected-locations '((logical-object
                                        recipient))))
   (last-name
     :initform (make-instance 'last-name-slot
                 :expected-locations '((logical-object
                                        recipient))))
   (po-box-nr
     :initform (make-instance 'po-box-nr-slot
                 :expected-locations '((logical-object
                                        recipient))))
```

DFKI Documents

D-92-06
Hans Werner Höper: Systematik zur Beschreibung von Werkstücken in der Terminologie der Featuresprache
392 Seiten

D-92-07
Susanne Biundo, Franz Schmalhofer (Eds.): Proceedings of the DFKI Workshop on Planning
65 pages

D-92-08
Jochen Heinsohn, Bernhard Hollunder (Eds.): DFKI Workshop on Taxonomic Reasoning Proceedings
56 pages

D-92-09
Gernod P. Laufkötter: Implementierungsmöglichkeiten der integrativen Wissensakquisitionsmethode des ARC-TEC-Projektes
86 Seiten

D-92-10
Jakob Mauss: Ein heuristisch gesteuerter Chart-Parser für attributierte Graph-Grammatiken
87 Seiten

D-92-11
Kerstin Becker: Möglichkeiten der Wissensmodel-lierung für technische Diagnose-Expertensysteme
92 Seiten

D-92-12
Otto Kühn, Franz Schmalhofer, Gabriele Schmidt: Integrated Knowledge Acquisition for Lathe Production Planning: a Picture Gallery (Integrierte Wissensakquisition zur Fertigungsplanung für Drehteile: eine Bildergalerie)
27 pages

D-92-13
Holger Peine: An Investigation of the Applicability of Terminological Reasoning to Application-Independent Software-Analysis
55 pages

D-92-14
Johannes Schwagereit: Integration von Graph-Grammatiken und Taxonomien zur Repräsentation von Features in CIM
98 Seiten

D-92-15
DFKI Wissenschaftlich-Technischer Jahresbericht 1991
130 Seiten

D-92-16
Judith Engelkamp (Hrsg.): Verzeichnis von Softwarekomponenten für natürlichsprachliche Systeme
189 Seiten

D-92-17
Elisabeth André, Robin Cohen, Winfried Graf, Bob Kass, Cécile Paris, Wolfgang Wahlster (Eds.): UM92: Third International Workshop on User Modeling, Proceedings
254 pages
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

D-92-18
Klaus Becker: Verfahren der automatisierten Diagnose technischer Systeme
109 Seiten

D-92-19
Stefan Dittrich, Rainer Hoch: Automatische, Deskriptor-basierte Unterstützung der Dokumentanalyse zur Fokussierung und Klassifizierung von Geschäftsbriefen
107 Seiten

D-92-21
Anne Schauder: Incremental Syntactic Generation of Natural Language with Tree Adjoining Grammars
57 pages

D-92-23
Michael Herfert: Parsen und Generieren der Prolog-artigen Syntax von RELFUN
51 Seiten

D-92-24
Jürgen Müller, Donald Steiner (Hrsg.): Kooperierende Agenten
78 Seiten

D-92-25
Martin Buchheit: Klassische Kommunikations- und Koordinationsmodelle
31 Seiten

D-92-26
Enno Tolzmann:
Realisierung eines Werkzeugauswahlmoduls mit Hilfe des Constraint-Systems CONTAX
28 Seiten

D-92-27
Martin Harm, Knut Hinkelmann, Thomas Labisch: Integrating Top-down and Bottom-up Reasoning in COLAB
40 pages

D-92-28
Klaus-Peter Gores, Rainer Bleisinger: Ein Modell zur Repräsentation von Nachrichtentypen
56 Seiten

Ein Modell zur Repräsentation von Nachrichtentypen

Klaus-Peter Gores, Rainer Bleisinger

D-92-28
Document