Kontrolliertes Englisch für Anforderungsspezifikationen

Abhandlung
zur Erlangung der Doktorwürde
der Philosophischen Fakultät I
der Universität Zürich

vorgelegt von
Rolf Schwitter

von
Schneisingen/Aargau

Angenommen auf Antrag von den Herren
Prof. Dr. Michael Hess
Prof. Dr. Harald Burger
Dr. Norbert E. Fuchs

Studentendruckerei
Zürich 1998

Danksagung

Diese Dissertation ist im Rahmen des Projekts *Attempto – Controlled English for Requirements Specifications* am Institut für Informatik (IFI) der Universität Zürich entstanden. Das Projekt wird von Dr. Norbert E. Fuchs geleitet und durch den Schweizerischen Nationalfonds zur Förderung der wissenschaftlichen Forschung (Projekte: 20-43540.95 und 20-47151.96) finanziell unterstützt.

Ich möchte an erster Stelle Norbert Fuchs für die Supervision und die vielen intensiven und nachhaltigen Diskussionen sowie für seinen kritischen Lupen-Blick und seine sanfte Hartnäckigkeit während der Ausarbeitung dieser Dissertation danken. Gleichzeitig gilt mein Dank Prof. Dr. Michael Hess und Prof. Dr. Harald Burger von der Philosophischen Fakultät I der Universität Zürich, dass Sie sich bereit erklärt haben, diese interdisziplinäre Arbeit - zwischen Informatik und Computerlinguistik - zu unterstützen und zu begutachten. Prof. Dr. Martin Glinz danke ich für die inspirierende Arbeitsumgebung in der Gruppe *Requirements Engineering* und die dafür nötigen Freiräume, die erst ein optimales wissenschaftliches Arbeiten erlauben. Prof. Dr. Kurt Bauknecht bin ich für die ausgezeichnete Infrastruktur am IFI, für seinen unermüdlichen Einsatz für's IFI und für seine sichere Hand, mit der er das Institut lenkt, zu Dank verpflichtet.

Vielfältige fachliche, technische, organisatorische und persönliche Unterstützung habe ich während der Arbeit erfahren. Für fachlich interessante Diskussionen und Ratschläge danke ich Martin Arnold, Stefan Berner, Norbert Fuchs, Michael Hess, Stefan Joos, Ina Kraan, Julian Richardson, Johannes Ryser, Uta Schwertel und Martin Volk. Für technische Unterstützung und besondere Hilfsbereitschaft möchte ich speziell Stefan Berner, Stefan Joos und Rico Solca danken. Für organisatorische und persönliche Unterstützung danke ich allen Freunden und Kollegen am IFI, die mir während meiner Arbeit manche Last abgenommen haben und mich immer wieder dann aufgemuntert haben, wenn der "Karren im Dreck" stecken blieb.

Rolf Schwitter
Zürich, Januar 1998

Inhaltsverzeichnis

1. Einleitung

> Somewhere between ridiculous pedantry and erroneous formulation there presumably exists a reasonably precise way of specifying a problem in English [Dodd 90:17].

1.1 Problembeschreibung

Der Werkstoff für die Entwicklung von Softwareprodukten sind Sprachen. In der Regel werden Sprachen mit ganz unterschiedlichen Eigenschaften dazu verwendet, Anforderungen zu klären, Anforderungen zu spezifizieren und Programme zu schreiben. Die natürliche Sprache bildet den Ausgangspunkt für die Softwareentwicklung. Sie dient einerseits im gesamten Entwicklungsprozess als Kommunikationsmittel zwischen den Beteiligten (= Systembenutzern, Anwendungsspezialisten und Softwareentwicklern) und andererseits als Werkstoff, durch den die wahrgenommenen Phänomene aus einem Anwendungsbereich in den ganz frühen Phasen der Entwicklung ausgedrückt, beurteilt und zueinander in Beziehung gesetzt werden. Natürliche Sprache ist ein praktischer Werkstoff, um unsichere Sachverhalte versuchsweise zu skizzieren. Aufgrund ihrer inhärenten Mehrdeutigkeit und Vagheit erweist sich die natürliche Sprache aber als ungeeignet, wenn es darum geht, gesicherte Sachverhalte eindeutig und präzis in einer Anforderungsspezifikation niederzuschreiben. Da das Endprodukt der Entwicklung immer ein formales Computerprogramm ist, das eine Maschine mit genau definiertem Verhalten und spezifischen Eigenschaften beschreibt, sind für den nahtlosen Übergang von der Spezifikation zum Programm bereits für das Schreiben von Anforderungsspezifikationen formale Sprachen vorgeschlagen worden. Formale Sprachen haben eine eindeutige Syntax, eine präzise Semantik und sind oft zusammen mit einer Beweistheorie definiert. Diese Eigenschaften bilden die Grundlage für ausführbare Spezifikationen und für die automatische Verifikation von Programmen. Formale Sprachen werden von den Softwareentwicklern als Werkstoff bevorzugt. Für die meisten Systembenutzer und Anwendungsspezialisten sind formale Sprachen jedoch nicht verständlich. Die Lesbarkeit einer Anforderungsspezifikation ist aber ein ganz entscheidendes Kriterium für die Adäquatheitsprüfung (= Validierung) der einzelnen Anforderungen. Sobald ein Softwareentwickler die informellen Anforderungen interpretiert und in eine formale Anforderungsspezifikation überführt hat, ist das Dokument für die Systembenutzer und Anwendungsspezialisten normalerweise nicht mehr zugänglich. Es ist für diese Beteiligten nicht mehr überprüfbar, ob die Anforderungsspezifikation die relevanten Sachverhalte korrekt, eindeutig und vollständig beschreibt.

1.2 Aufgabenstellung

Ausgehend von einer Untersuchung der Begriffe *Anforderung* und *Spezifikation* sowie einem Vergleich zwischen natürlichen und formalen Spezifikationssprachen, soll anhand eines Spezifikationsbeispiels aus der Literatur (Bibliotheksproblem) illustriert werden, welche Probleme sich für den Anwendungsspezialisten ergeben, wenn eine informelle Problembeschreibung in eine formale Anforderungsspezifikation überführt wird. Um die konzeptionelle Lücke, die bei der Verwendung von formalen Spezifikationssprachen zwischen den Anwendungsspezialisten und den Softwareentwicklern entsteht, zu überbrücken, soll nach einer geeigneten Spezifikationssprache gesucht werden. Zu diesem Zweck soll abgeklärt werden, inwieweit kontrollierte natürliche Sprachen, die vor allem in der Industrie für die bessere Verständlichkeit und Verarbeitbarkeit von technischen Dokumenten entwickelt worden sind, als Modell für eine anwendungsspezifische Spezifikationssprache dienen können. Für die Softwareentwicklung stellt sich dann die Frage, ob eine kontrollierte natürliche Sprache definiert werden kann, die erstens die gleichen Eigenschaften wie eine formale Spezifikationssprache hat, die zweitens nach computerlinguistischen Gesichtspunkten effizient verarbeitet werden kann und die drittens für den Anwendungsspezialisten leicht erlernbar ist. Abschliessend soll die zu entwickelnde Spezifikationssprache am Bibliotheksproblem erprobt werden, so dass die Ergebnisse der Arbeit überprüfbar werden. Die englische Sprache ist als Ausgangspunkt gewählt worden, weil die Problembeschreibung für das Spezifikationsbeispiel bereits englischsprachig vorliegt und weil die relativ feste Wort- und Satzgliedstellung des Englischen unsere Zielsetzung unterstützen.

1.3 Resultate

Mit Attempto Controlled English (ACE) wird eine kontrollierte natürliche Sprache als textuelle Sicht auf eine formale Spezifikation in Logik vorgeschlagen. Die Sprache ACE erlaubt es dem Anwendungsspezialisten, wahre und bedingte Sachverhalte in einer begriffsnahen Notation direkt und präzis zu beschreiben. ACE besteht aus einer wohldefinierten Teilmenge der englischen Sprache und wirkt deshalb informell und vertraut, obwohl die Sprache in ihrem Kern formal ist. ACE kann eindeutig in Diskursrepräsentationstheorie (d.i. eine strukturierte Form der Prädikatenlogik) übersetzt werden. Um den nötigen Grad an Präzision zu erreichen, verwendet der Anwendungsspezialist beim Schreiben der Spezifikation einen einfachen abstrakten Satzbauplan als mnemotechnisches Hilfsmittel und eine Reihe von Konnektoren (d.h. Koordinatoren und Subordinatoren), mit denen er aus einfachen ACE Sätzen zusammengesetzte ACE Sätze bilden kann. Die Sprache ACE basiert auf einer leicht merkbaren

Menge von Schreibprinzipien, die bestimmte Formen von Mehrdeutigkeit bei der Sprachverarbeitung unter Kontrolle bringen. Das führt einerseits zu einer besseren Lesbarkeit der Anforderungsspezifikation und andererseits zu einer Vereinfachung bei der maschinellen Verarbeitung. Um die Sprache erfolgreich verwenden zu können, muss der Anwendungsspezialist nur diese Schreibprinzipien erlernen.

Die Sprache ACE ist in das Attempto Spezifikationssystem eingebettet. Der Spezifikationstext wird mit bekannten computerlinguistischen Methoden automatisch verarbeitet und eindeutig in Diskursrepräsentationsstrukturen übersetzt. Die anschliessende Weiterverwendung der übersetzten Spezifikation ist ähnlich, wie das aus der Verarbeitung von formalen Spezifikationssprachen bekannt ist. Ausser dem Spezifikationstext und dem anwendungsspezifischen Lexikon wird kein zusätzliches Weltwissen im Attempto System modelliert.

Parallel zur Übersetzung in Diskursrepräsentationsstrukturen erzeugt das Attempto System für jeden Satz eine eindeutige Paraphrase in kontrollierter natürlicher Sprache und macht so sichtbar, wie der Satz interpretiert worden ist. Wenn ein Satz nicht analysiert werden kann oder ein Wort nicht bekannt ist, wird dies dem Anwendungsspezialisten mitgeteilt. Er kann das fachspezifische Vokabular des Spezifikationssystems zu jedem Zeitpunkt interaktiv erweitern und braucht dazu bloss schulgrammatisches Wissen. Um eine konsistente Verwendungsweise der lexikalischen Elemente in einer Spezifikation für unterschiedliche Benutzer zu garantieren, können informelle Erkennungsregeln als Kommentare den einzelnen Lexikoneinträgen beigefügt werden. Eine informelle Erkennungsregel hilft, eine Entität, auf die das sprachliche Zeichen Bezug nimmt, im Anwendungsbereich zu identifizieren.

Die gesamte Anforderungsspezifikation wird vom Anwendungsspezialisten auf der Ebene der kontrollierten natürlichen Sprache entwickelt. Der Anwendungsspezialist kann sich über die spezifizierten Sachverhalte orientieren, indem er Fragen in ACE an die in Logik übersetzte Spezifikation richtet. Die Antwort kommt durch logische Inferenz zustande und besteht wiederum aus einem ACE Satz oder einer ACE Konstruktion. Darüber hinaus sind ACE Spezifikationen interaktiv ausführbar, so dass der Anwendungsspezialist die Ausführung in ACE beobachten und seine intendierte Interpretation unmittelbar überprüfen kann.

1.4 Struktur der Arbeit

Die Arbeit ist folgendermassen strukturiert: Das ***zweite Kapitel*** untersucht, welche Rollen Anforderungen und Spezifikationen bei der Softwareentwicklung spielen. Das führt zu einer begrifflichen Unterscheidung von modalen und faktischen Anforderungen und zu einer Aufstellung von Qualitätsattributen für gut geschriebene Spezifikationen. Im Anschluss daran werden die Eigenschaften von natürlichen Sprachen und formalen Sprachen als Spezifikationssprachen untersucht und ihre Stärken und Schwächen diskutiert. Am Bibliotheksproblem - einem bekannten Spezifikationsbeispiel - wird anhand von verschiedenen Lösungsvorschlägen gezeigt, welche Schwierigkeiten entstehen, wenn man eine informelle Problembeschreibung in eine formale Spezifikation überführt. Das ***dritte Kapitel*** geht diesen Schwierigkeiten auf den Grund und skizziert einen Vorschlag, wie durch die Verwendung von textuellen Sichten auf formale Spezifikationen informelle und formale Aspekte in Einklang gebracht werden können. Textuelle Sichten orientieren sich an der Benutzersprache und werden in einer präzis definierten Teilmenge der natürlichen Sprache erstellt. Das ***vierte Kapitel*** untersucht, inwieweit sich bekannte kontrollierte natürliche Sprachen oder allenfalls Subsprachen als Modell für das Schreiben von Spezifikationen auf der textuellen Sichtebene eignen. Es wird gezeigt, mit welchem Erfolg kontrollierte natürliche Sprachen - in verwandten Gebieten - für die Dokumentation und maschinelle Übersetzung eingesetzt werden und wie Subsprachen für die Spezifikation genutzt werden. Diese Ansätze geben Anhaltspunkte, eignen sich aber nur bedingt, um eindeutige Spezifikationen zu schreiben, da die Sprachen zu wenig genau definiert sind. Das ***fünfte Kapitel*** stellt die Sprache *Attempto Controlled English (ACE)* vor, die informell und vertraut erscheint, aber in ihrem Kern formal und eindeutig ist und die Schwachstellen von bisherigen Spezifikationssprachen behebt. Es wird für die Philosophie von ACE geworben und mit einer Einführung in die linguistischen Grundlagen von ACE der Weg für eine weitergehende Diskussion der Sprache geebnet. Die anschliessende Darlegung erörtert, welche Wörter und Konstruktionen in ACE zulässig sind und wieso andere Elemente ausgeschlossen bleiben müssen. Diese Entscheidungen werden durch einen Satzbauplan und 30 Schreibprinzipien greifbar gemacht. Das ***sechste Kapitel*** zeigt eine Evaluation der Sprache ACE am Bibliotheksproblem und macht die Zweckmässigkeit und Handhabbarkeit der Sprache überprüfbar. Das ***siebte Kapitel*** verdeutlicht an einem kleinen Spezifikationsausschnitt, welche Funktionalität das Attempto Spezifikationssystem den Benutzern bei der Erstellung einer ACE Spezifikation bietet und wie die linguistische Verarbeitung erfolgt. Das ***achte Kapitel*** fasst die wichtigsten Resultate der Arbeit zusammen, spricht offene Probleme an, ermuntert zur Weiterarbeit und zur konstruktiven Kritik.

2. Anforderungen und Spezifikationen

Die Entwicklung eines Softwareproduktes beginnt gewöhnlich mit *Requirements Engineering*. Diese Phase umfasst zwei Aktivitäten: zum einen die Anforderungsanalyse und zum anderen die Anforderungsspezifikation. Während der Anforderungsanalyse wird ein zu lösendes Problem gedanklich und sprachlich durchdrungen, so dass ein Problembewusstsein entstehen kann, aufgrund dessen die Beteiligten die Anforderungen an ein zu entwickelndes Softwaresystem erkennen und zueinander in Beziehung setzen können. Sobald ein hinreichendes Verständnis der Anforderungen erreicht ist und die Beteiligten davon überzeugt sind, die relevanten Sachverhalte aus dem Anwendungsgebiet zu erkennen, ist es Zeit für die Anforderungsspezifikation. Während der Anforderungsspezifikation wird das funktionale Verhalten des geplanten Softwareproduktes in einem Dokument - der sogenannten Software-Anforderungsspezifikation - vollständig spezifiziert. Die funktionalen Anforderungen beschreiben, was das Produkt aus der Sicht des Anwendungsspezialisten tun soll, ohne darauf einzugehen, wie diese Leistungen zu erbringen sind [vgl. Davis 90:17, Partsch 91:34]. Von den funktionalen Anforderungen sind die nichtfunktionalen Anforderungen zu unterscheiden. Nichtfunktionale Anforderungen machen quantifizierbare und nichtquantifizierbare Aussagen darüber, wie das geplante Softwareprodukt die gestellten funktionalen Anforderungen erfüllen soll und schränken die Anzahl der möglichen Implementationen ein [vgl. Partsch 91:35, Stokes 91:16/3]. Oft wird empfohlen, nichtfunktionale Anforderungen wie Schnittstellenattribute, Ausführungsverhalten, Zuverlässigkeit und Produktestandards in der Software-Anforderungsspezifikation textuell unabhängig zu spezifizieren [Sommerville 96:69].

2.1 Anforderungen

Während der Anforderungsanalyse untersuchen die Beteiligten das Anwendungsgebiet, um alle Phänomene und Beziehungen zu identifizieren, die im Problemkontext eine massgebende Rolle spielen. Der Problemkontext ist der Ort, für den eine Softwarelösung gesucht wird. Ein Problem ist nicht durch die Art der zu bauenden Maschine charakterisiert, sondern einerseits durch die Struktur und Eigenschaften des Anwendungsgebiets und andererseits durch die Anforderungen der Systembenutzer und Anwendungsspezialisten in diesem Anwendungsgebiet [Jackson 95:10 ff.].

Anforderungen sind schwierig zu entdecken, da sie aus unterschiedlichen, heterogenen Quellen stammen können, die nicht allen Beteiligten zugänglich sind. Die Benutzer und Anwendungsspezialisten haben normalerweise verschiedenartige Vorstellungen darüber, welche Funktionalität das zukünftige Softwareprodukt zur Verfügung stellen soll und unter welchen Bedingungen es eingesetzt werden soll. Systematisch genutzt, können aber die divergenten Sichtweisen der Benutzer und Anwendungsspezialisten dazu dienen, Anforderungen zu erheben, zu überprüfen, zu ergänzen und allfällige Konflikte explizit zu machen [Pohl 93:282 ff.]. Zu diesem Zweck sind verschiedene Interview- und Fragetechniken entwickelt worden, um die Anforderungen an ein Softwareprodukt ausfindig zu machen. Eine weitere beliebte Technik der Informationsgewinnung sind Szenarien, die Verhaltensaspekte des zu entwickelnden Produktes vorab veranschaulichen können [Schach 96:198 ff.]. Diese unterschiedlichen Arten der Konzeptexploration sind notwendig und zeigen, dass keiner der Benutzer und Anwendungsspezialisten in dieser frühen Phase der Softwareentwicklung mit einer kompletten Liste aller Anforderungen aufwarten kann. Anforderungen wachsen kontinuierlich, während das Problem analysiert wird und ein Problembewusstsein bei den Beteiligten entsteht.

Die Praxis zeigt, dass Anwendungsspezialisten ihre Anforderungen fast immer in fachspezifischen Begriffen von unterschiedlichem Abstraktionsgrad ausdrücken und dazu auf bekannte Konventionen zurückgreifen. Vorzugsweise werden zur Darstellung des sachlichen Wissens informelle Notationen verwendet, die einen möglichst freien Detaillierungsgrad zulassen. Beispielsweise stellen die folgenden natürlichsprachlichen Anforderungen einen Sachverhalt in einer Bibliothek mit zunehmender Genauigkeit dar, und zwar ausgehend von einer abstrakten Beschreibung in (1) hin zu einer detaillierteren Beschreibung in (4):

(1) *The borrower returns a book.*

(2) *The ordinary borrower returns a book.*

(3) *The ordinary borrower returns a copy of a book.*

(4) *The ordinary borrower returns a copy of a book which he currently has.*

Normalerweise verwenden die Anwendungsspezialisten zur Beschreibung der Sachverhalte sprachliche Ausdrücke, mit denen sie vertraut sind, die aber oft implizites Wissen aus einem Anwendungsgebiet kondensieren und deshalb für den Softwareentwickler nicht unmittelbar verständlich sind. Neben der natürlichen Sprache als primärem Kommunikationsmittel werden unterstützend oft Diagramme, Tabellen, Videos und andere formalere Zeichensysteme benutzt, um Anforderungen zu klären. Die Schwierigkeit dabei ist, die entstehenden Informationen so zu ordnen, dass die

verschiedenartigen Sichtweisen der Beteiligten zum Ausdruck kommen und die Konflikte sichtbar werden. Die Erwägungen müssen lösungsneutral sein und dürfen keine Entwurfsentscheidungen vorwegnehmen.

Naturgemäss sind die Anforderungen zu diesem Zeitpunkt umstritten und unsicher: Sie werden postuliert, modifiziert und wieder verworfen. Bei Anforderungen, die in natürlicher Sprache geäussert werden, widerspiegelt sich diese Unsicherheit häufig auf der sprachlichen Ebene, wo viele modale Konstruktionen (z.B. *can, must, should, probably, possibly*) zu finden sind. Diese Konstruktionen bringen die subjektive Einstellung der Anwendungsspezialisten zu den Anforderungen zum Ausdruck. An ihnen sind die zweifelnden oder unsicheren Einschätzungen der Satzinhalte (= Propositionen) ablesbar. Modale Konstruktionen geben die Einstellung des Sprechers oder Schreibers hinsichtlich der Wahrheit an, die einer Proposition, welche einen Sachverhalt durch einen Satz ausdrückt, zugesprochen werden kann. Beispielsweise liegt den beiden folgenden Sätzen

(5) *The borrower returns a copy of a book.*

(6) *The borrower should return a copy of a book.*

dieselbe Proposition zugrunde. Die Aussage in (5) ist wahr, wenn der durch den Satz ausgedrückte Sachverhalt besteht. Zusätzlich zur propositionalen Basisstruktur weist der Satz (6) eine modale Komponente auf. Das modale Hilfsverb *should* zeigt an, dass der Sachverhalt entweder mit einer bestimmten Wahrscheinlichkeit gilt (epistemische Lesart), oder dass Faktoren existieren, die die Verbindlichkeit des im Satz ausgedrückten Sachverhaltes stützen (deontische Lesart) [Huddleston 88:78, Lyons 95:254]. Die beiden unterschiedlichen Lesarten können mit Hilfe von Paraphrasen (= sinngebundene Umschreibungen) verdeutlicht werden:

(7) *Relative to what the speaker knows, it is probable that the borrower returns a copy of a book.*

(8) *Certain factors oblige the speaker to believe or (even) to cause that the borrower returns a copy of a book.*

Die Paraphrase (7) zeigt, dass der Sprecher den Satz (6) möglicherweise deshalb äussert, weil er nicht die unmittelbaren Sachkenntnisse hat, die die Behauptung von (5) rechtfertigen würde. Andererseits wird durch die Paraphrase (8) deutlich, dass es dem Sprecher in (6) allenfalls darum gehen kann, den Sachverhalt, der in (5) behauptet wird, herbeiführen zu wollen.

Was genau sind denn nun Anforderungen?
Ein Blick in das IEEE Standard Glossary of Software Engineering Terminology [IEEE Std 610.12 90:62] macht schnell klar, dass der Ausdruck *Anforderung* (engl. *requirement*) für die Bezeichnung verschiedener Inhalte gebraucht wird:

Anforderung

(1) A condition or capability needed by a user to solve a problem or achieve an objective.

(2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.

(3) A documented representation of a condition or capability as in (1) or (2).

Eine Anforderung wird hier als *Bedingung* (*condition*) oder *Fähigkeit* (*capability*) definiert, die entweder von einem Benutzer benötigt wird, um ein Ziel zu erreichen, oder von einem System eingehalten werden muss, um eine bestimmte Leistung erbringen zu können. Ferner wird unter einer Anforderung auch die Beschreibung einer solchen Bedingung oder Fähigkeit verstanden. In den drei Definitionen wird nichts darüber ausgesagt, in welcher (sprachlichen) Form die Anforderungen zu verfassen sind. Eine Antwort darauf finden wir, wenn wir uns fragen, unter welchen Voraussetzungen Anforderungen zu Stande kommen und was sie schlussendlich leisten müssen.

Während der Anforderungsanalyse geht es darum, die Bedürfnisse der Benutzer zu klären und ein kollektives Verständnis für das Problem unter den Beteiligten zu erzielen, um die geforderte Funktionalität für das zu realisierende Softwareprodukt aushandeln zu können. Da das Wissen über das Anwendungsgebiet zu diesem Zeitpunkt oft unzusammenhängend und unvollständig ist, erstaunt es wenig, dass die natürlichsprachlich geäusserten Anforderungen eine Fülle von modalen Konstruktionen aufweisen, die die subjektive Einstellung der Sprecher offenbaren. Die natürliche Sprache ist ein ausgesprochen nützlicher Werkstoff, wenn es darum geht, erste Anforderungen an ein System, das bloss in den Köpfen der Beteiligten existiert, versuchsweise zu skizzieren. Um herauszustreichen, dass solche Anforderungen Eigenschaften beschreiben, die nicht notwendigerweise durch Beobachtung oder Experiment falsifizierbar sind [vgl. Jackson 95:109], bezeichnen wir diese Art von Anforderungen als ***modale Anforderungen***.

Während der Anforderungsspezifikation wird ein Dokument erstellt, das das externe (beobachtbare) Systemverhalten und dessen Einschränkungen präzis spezifiziert. Bei dieser Aktivität fokussieren wir auf gesicherte Anforderungen, die diejenigen Bedingungen oder Fähigkeiten beschreiben und normieren, die das Softwareprodukt erfüllen muss und denen alle Beteiligten zustimmen können. Wenn natürliche Sprache als Spezifikationssprache verwendet wird, ist es vorteilhaft, die gesicherten Anforderungen als deklarative Sätze oder als konditionale Sätze niederzuschreiben, um sie so von den modalen Anforderungen zu unterscheiden [vgl. Jackson 95:127]. Anforderungen, die wahre oder bedingte Sachverhalte beschreiben und im Prinzip durch Beobachtung oder Experiment falsifiziert werden können, bezeichnen wir als ***faktische Anforderungen***.

Die dritte Definition wollen wir nicht weiter untersuchen, da modale Anforderungen durchaus dokumentiert sein können und faktische Anforderungen notwendigerweise dokumentiert sein müssen, da sie in die Software-Anforderungsspezifikation eingehen.

Gegenstand der nachfolgenden Diskussion bilden ausschliesslich faktische Anforderungen, die durch Validierung falsifizierbar sind. Wir untersuchen zuerst, welchen Qualitätsattributen eine Anforderungsspezifikation idealerweise zu genügen hat, und stellen vor diesem Hintergrund anschliessend die Frage, wo natürliche und formale Spezifikationssprachen ihre Stärken und Schwächen haben.

2.2 Spezifikationen

Sobald ein relativ umfassendes Problemverständnis erreicht ist und die Bedürfnisse der Anwendungsspezialisten geklärt sind, müssen die faktischen Anforderungen während der Anforderungsspezifikation in einem Dokument durch eine geeignete Spezifikationssprache beschrieben werden. Unabhängig vom verfolgten Software-Entwicklungsparadigma bildet die Software-Anforderungsspezifikation (SAS) den Ausgangspunkt für alle späteren Phasen im Software-Entwicklungsprozess [vgl. Fromherz 93:9 ff.]. Die SAS ist ein konzeptionelles Modell des zu entwickelnden Softwareproduktes. Der Zweck der SAS besteht darin, die notwendige Information vollständig und widerspruchsfrei zur Verfügung zu stellen, um den nachfolgenden Systementwurf und die Systemimplementation möglichst erfolgreich durchführen zu können. Dabei spielt die Qualität der SAS eine entscheidende Rolle für die kostengünstige Entwicklung eines zuverlässigen Softwareproduktes.

> Without a well-written requirements specification, developers do not know what to build, customers do not know what to expect, and there is no way to validate that the system as built satisfies the requirements [Hsia et al. 93:75].

Die SAS ist nicht bloss eine Zusammenstellung aller Anforderungen an das Softwareprodukt, sondern auch ein Kommunikationsmedium, das den Softwareentwicklern die Intentionen der Anwendungsspezialisten zugänglich machen sollte. Neben der Beschreibung des beobachtbaren Verhaltens des zukünftigen Softwareproduktes bildet die SAS den Ausgangspunkt für die Validierung der Anforderungen durch die Anwendungsspezialisten. Zudem ist es möglich, aus einer gut geschriebenen SAS Testfälle zu extrahieren oder, falls die SAS in einer formalen Sprache geschrieben ist, die davon abgeleiteten Systembeschreibungen automatisch zu verifizieren. Im Idealfall liegt einer formalen SAS ein operationales Modell zugrunde, so dass die Spezifikation ausgeführt werden kann. Das hat den grossen Vorteil, dass das generierte Verhalten vom Anwendungsspezialisten in einer sehr frühen Phase im Software-Entwicklungsprozess überprüft werden kann [vgl. Fuchs 92:323 ff., Gravell & Henderson 96:104 ff.]. Im Gegensatz zu Prototypen beschreiben ausführbare Spezifikationen die gesamte Funktionalität des geplanten Softwareproduktes und machen eine klare Trennung zwischen problem- und implementationsorientierten Aspekten [Fromherz 93:18].

Welches sind die Qualitätsattribute, denen eine gut geschriebene SAS zu genügen hat? Meistens werden die folgenden sechs inhaltlichen Attribute unterschieden [Davis 90:184 ff., Davis et al. 93:141 ff., Davis 94:1050 ff., IEEE Std 830 93:5 ff.]:

- korrekt
- eindeutig
- vollständig
- verifizierbar
- konsistent
- verständlich

Eine SAS ist dann *korrekt*, wenn jede aufgeführte Anforderung einen funktionalen Aspekt beschreibt, der vom zukünftigen Softwareprodukt erfüllt wird. Es gibt weder Werkzeuge noch Verfahrensweisen, die Korrektheit zusichern könnten. Die Korrektheit einer Anforderung hängt vollständig von der intendierten Anwendung ab. Beispielsweise ist die Anforderung

(9) *The user searches for the name of the borrower who currently has the copy.*

an ein Bibliothekssystem inkorrekt, wenn in (9) mit dem Oberbegriff *user* sowohl jemand aus der Personengruppe der Mitarbeiter (*staff user*) als auch aus der Personengruppe der Ausleiher (*borrower*) bezeichnet wird, aber beim Schreiben der Spezifikation eigentlich intendiert worden ist, dass nur jemand aus der Belegschaft nachschauen darf, wer gegenwärtig eine (Buch-)Kopie besitzt. Die Validierung der Anforderungen muss von den Anwendungsspezialisten durchgeführt werden, die als einzige profunde Kenntnisse des Anwendungsgebiets haben. Das Auffinden von Inkorrektheit ist schwierig. Manuell kann die Suche nur durch sorgfältiges Lesen der Spezifikation (= Inspektion) verrichtet werden. Automatische Unterstützung ist dann möglich, wenn die SAS ausführbar ist oder wenn ein Prototyp die relevanten Aspekte abdeckt.

Eine SAS ist dann *eindeutig*, wenn jede aufgeführte Anforderung für alle Beteiligten nur eine einzige Interpretation hat. Natürlich-sprachliche Anforderungen sind inhärent mehrdeutig. Dies wird im normalen Sprachgebrauch oft nicht bemerkt, da der Kontext determinierend wirkt. Ein klassisches Beispiel sind logische Mehrdeutigkeiten, die durch den Skopus (= Bezugsbereich) von quantifizierten Nominalphrasen entstehen. Der Satz

(10) *Every borrower has a book limit.*

hat aufgrund des Skopus von *every borrower* und *a book limit* zwei implizite Lesarten. Die Bestimmung der korrekten Lesart von (10) hängt von kontextuellen Faktoren ab und wird in gesprochener Sprache häufig durch die Intonation fixiert. Im Gegensatz zu natürlicher Sprache weisen formale Spezifikationssprachen keine inhärenten Mehrdeutigkeiten auf. In einer formalen Sprache wie der Prädikatenlogik werden die beiden Lesarten von (10) durch explizite Darstellungen unterschieden:

(11) $\forall X: (borrower(X) \rightarrow \exists Y: (have(X,Y) \wedge book\_limit(Y)))$

(12) $\exists Y: (book\_limit(Y) \wedge \forall X: (borrower(X) \rightarrow have(X,Y)))$

Die Verwendung von formalen Spezifikationssprachen schliesst solche Mehrdeutigkeiten in der SAS aus. Meistens hat das jedoch einen unerwünschten Nebeneffekt, denn der höherer Grad an Formalität reduziert die gute Verständlichkeit der SAS für die Anwendungsspezialisten.

Eine SAS ist dann *vollständig*, wenn die relevante Funktionalität des Softwareproduktes durch die einzelnen Anforderungen vollumfänglich beschrieben wird. Insbe-

sondere sind die Systemantworten für alle Eingabedaten in sämtlichen Situationen komplett zu spezifizieren. Auch Vollständigkeit ist ein subjektives Kriterium, das nur durch den Anwendungsspezialisten entschieden werden kann. Falls die SAS unvollständig ist, sind die Softwareentwickler womöglich gezwungen, Annahmen über das vom Anwendungsspezialisten intendierte Systemverhalten zu treffen. Diese Annahmen können falsch sein und zu risikoreichen Folgefehlern führen. Das Ziel muss sein, dass alle Beteiligten zu einem frühen Zeitpunkt im Software-Entwicklungsprozess ein möglichst hohes Mass an Vertrauen in die Vollständigkeit der Anforderungen erhalten. Zu den wichtigsten Techniken zur Überprüfung der Vollständigkeit gehören das kritische Lesen der SAS und der Einsatz von Prototypen:

> To achieve completeness by any definition, reviews of the SRS [software requirements specification] by customer or user are essential. Prototypes also help raise awareness of new requirements and help us better understand poorly or abstractly defined requirements [Davis et al. 93:145].

Eine SAS ist dann *verifizierbar*, wenn jede einzelne (formalisierte) Anforderung verifizierbar ist. Eine Anforderung ist verifizierbar, wenn ein Verfahren existiert, wodurch nachgewiesen werden kann, dass das gebaute Softwareprodukt die Anforderung erfüllt. Das heisst, dass unabhängig vom Software-Entwicklungsparadigma für jede einzelne Abstraktionsebene der Systembeschreibung nachgewiesen werden muss, dass sie mit der vorausgehenden Beschreibungsebene konsistent ist. Die Verifikation eines Programms ist ein rein analytischer Prozess, der im Prinzip automatisierbar ist, wenn die SAS in einer formalen Sprache vorliegt und die einzelnen Anforderungen eindeutig, entscheidbar und messbar sind. Zum Beispiel ist ein Programm hinsichtlich der (modalen) Anforderung

(13) *The borrower usually returns the book within a week.*

nicht verifizierbar, da das Adverb *usually* die Häufigkeit relativiert, mit der das Ereignis eintritt.

Eine SAS ist dann *konsistent*, wenn die einzelnen Anforderungen nicht miteinander in Konflikt stehen. Inkonsistenz kann zum Beispiel dann entstehen, wenn spezifizierte Eigenschaften von Entitäten einander in die Quere kommen:

(14) *Every user is either a staff user or a borrower.*

(15) *No borrower is a user.*

Die Inkonsistenz der beiden Anforderungen ist hier offensichtlich. Doch in umfangreichen und komplexen SASen, in denen die Information verteilt ist, wird es sehr schwierig, Inkonsistenz durch manuelle Techniken (Reviews: Inspektionen oder Walkthroughs) zu entdecken. Der Einsatz automatischer Konsistenzprüfung durch mathematische Beweisführung ist davon abhängig, ob die Spezifikationssprache formal oder formalisierbar ist und ob die automatische Technik nicht zu kombinatorischer Explosion führt.

> Checking consistency, like verification, is a purely analytical process. Its successfulness depends on the language that is used to express the requirements, and while the representation and description of the requirements remain a problem, checking consistency will also be problematical [Stokes 91:16/7].

Ein weiteres Beispiel für Inkonsistenz sind temporale Konflikte, die zwischen Anforderungen auftreten können:

(16) *The staff user enters the password and the user identification.*

(17) *LibDB asks for the user identification and reads the password.*

In (16) wird spezifiziert, dass der Mitarbeiter (*staff user*) zuerst das Passwort (*password*) und dann die Benutzeridentifikation (*user identification*) eingibt. Zusammen mit (17) gelesen, führt das jedoch zu Inkonsistenz, da die Bibliotheksdatenbank (*LibDB*) das Passwort erst liest, nachdem es die Benutzeridentifikation verlangt hat. Ein weiterer Konflikt kann durch die inkonsistente Verwendungsweise von Ausdrücken in der SAS entstehen, und zwar dann, wenn in zwei oder mehr Anforderungen dieselbe Entität durch unterschiedliche Ausdrücke beschrieben wird (ohne dass die Ausdrücke zuvor in einem Glossar oder Lexikon als Synonyme gekennzeichnet worden sind):

(18) *Every copy of a book has an alphanumeric serial number.*

(19) *If the digital serial number of the copy is correct then ...*

Beispielsweise wird dieselbe Laufnummer (*serial number*) in (18) einmal als *alphanumeric* und in (19) einmal als *digital* bezeichnet, obwohl der Laufnummer nur eine der beiden Eigenschaften zukommen kann.

Eine SAS ist dann *verständlich*, wenn alle Beteiligten die Bedeutung der Anforderungen ohne zusätzliche Erklärungen verstehen können. Verständlichkeit und gute Lesbarkeit sind die Grundvoraussetzungen für die Validierbarkeit der SAS durch die Anwendungsspezialisten, so dass diese feststellen können, ob die Beschreibung ihre Intentionen wiedergibt. Obwohl die volle natürliche Sprache scheinbar ein ideales Kommunikationsmittel für alle Beteiligten ist, führt ihre potentielle Mehrdeutigkeit zu grossen Schwierigkeiten, wenn die Softwareentwickler eine formale Spezifikation aus einer informellen SAS ableiten müssen. Einerseits ist den Softwareentwicklern am besten mit formalen Spezifikationssprachen gedient, die die Eleganz und die Präzision der Mathematik haben. Andererseits hat für die Anwendungsspezialisten die Verständlichkeit und Lesbarkeit der SAS die höchste Priorität [vgl. van Vliet 93:156 ff.].

> In an attempt to make a SRS less ambiguous, more verifiable, complete, and consistent, we might be tempted to resort to extremely formal notations. Unfortunately such notations often make it impossible for non-computer specialists to understand the SRS [Davis 90:191].

Eine Spezifikationssprache zu entwickeln, die den Bedürfnissen aller Beteiligten entspricht, also sowohl gut lesbar als auch formal und maschinell verarbeitbar ist, so dass sämtliche Qualitätsattribute einer SAS erfüllbar sind, scheint der Quadratur des Zirkels gleich zu kommen.

2.3 Eigenschaften von natürlichen Sprachen als Spezifikationssprachen

2.3.1 Übersicht

Eine in allen natürlichen Sprachen beobachtbare Eigenschaft ist die Ambiguität (= Mehrdeutigkeit) von Ausdrücken, die auf der lexikalischen, syntaktischen, semantischen oder pragmatischen Ebene der jeweiligen Sprache nachweisbar ist. Ein Ausdruck ist dann ambig, wenn ihm mehrere Interpretationen zugeordnet werden können. Ambiguität ist durch Paraphrasenbildung (= Umschreiben) auflösbar. Paraphrasen können die einzelnen diskreten Lesarten präzisieren. Den ambigen Satz

(20) *Every borrower has a book limit.*

kann man beispielsweise dadurch präzisieren, indem man das Adjektive *personal* einführt, das die korrekte Lesart verdeutlicht:

(21) *Every borrower has a personal book limit.*

Wenn ein natürlich-sprachlicher Ausdruck keine diskrete Präzisierung erlaubt, da genauere Kriterien fehlen, um zu entscheiden, ob gewisse Eigenschaften einer Entität oder einem Sachverhalt zukommen oder nicht, dann spricht man von Vagheit [vgl. Pinkal 96:72 ff.]. Der Satz

(22) *The library database is small.*

ist nicht durch eine Paraphrase präzisierbar, da das prädikative Adjektiv *small* kontextabhängig ist. Vage Ausdrücke haben ihre Ursache in der Unschärfe der aussersprachlichen Wirklichkeit und in der mangelhaften Vertrautheit der Sprecher und Schreiber mit den durch die Äusserungen bezeichneten Entitäten und Sachverhalten. Vagheit ist nicht nur etwas Subjektives. Sie liegt auch in der Tatsache begründet, dass die natürliche Sprache nicht die ganze Wirklichkeit abbildet, sondern bloss ein Bild der Wirklichkeit durch einen klassifizierenden Rahmen festlegt. Die einzige Methode, die Vagheit von Ausdrücken zu verringern und einer idealen Exaktheit näher zu kommen, besteht darin, die Ausdrücke zu definieren. Eine Begriffsdefinition setzt die explizite Vereinbarung von anderen Grundbegriffen voraus. Streng genommen kann das nur durch den terminologischen Aufbau im Rahmen einer Theorie für ein Anwendungsgebiet geleistet werden [vgl. Lewandowski 94:795 ff.].

Ambiguität bleibt im alltäglichen Sprachgebrauch und beim Schreiben von natürlich-sprachlichen SASen oft unbemerkt, da das Weltwissen, das Situationswissen und das sprachliche Wissen [Grice 75:41 ff.] dafür sorgen, dass schlussendlich für eine ambige

Struktur nur eine Interpretation pro Rezipierenden übrigbleibt. Speziell dort, wo die Qualität des Textes ein ausdrückliches Ziel ist, helfen die Sprecher und Schreiber vielfach durch die explizite Verwendung von Disambiguierungssignalen (*each, both, a different, the same, as a group*) auf die intendierte Interpretation hinzulenken.

Der Mensch hat ausgeprägte perzeptive Fähigkeiten, ambige Sätze sinnvoll in den Äusserungskontext einzubetten und die notwendige Information zur Disambiguierung aus dem aktuellen Kontext zu extrahieren, wobei seine Hypothesen (inneres Modell vom Anwendungsbereich) sowie seine Überzeugungen (auch falsche) die Parsingstrategie beeinflussen.

> In particular, human parsing seems closer to a **deterministic** process - that is, a process that doesn't extensively search through alternatives but rather uses the information it has at the time to choose the correct interpretation [Allen 95:159].

Evidenz, dass die Parsingstrategie beim Menschen nicht auf blinder Suche beruht, kommt auch aus der kognitiven Psychologie [vgl. Collins & Loftus 75:407 ff.]. Ein gut untersuchtes und experimentell belegtes Phänomen ist der Priming-Effekt. Netzwerkmodelle des semantischen Gedächtnisses zeigen, dass die mentale Verarbeitung eines bestimmten Begriffs, die Verarbeitung von semantisch benachbarten Begriffen aktiviert und unterstützt. Der Grad der Aktivierung eines Begriffs ist dann eine Funktion der semantischen Nähe zum ursprünglichen Begriff. Ist beispielsweise ein Wort ambig, dann kann aufgrund der semantischen Distanz eine Lesart bevorzugt werden [Hirst 87:86 ff.].

Jedes Verfahren, das eine natürlich-sprachliche SAS in eine formalere Darstellung überführt, muss dem Anwendungsspezialisten klar machen können, für welche Lesart sich der Softwareentwickler bei der Analyse entschieden hat. Dazu bieten sich Paraphrasen an, die das Gemeinte durch Umschreibung und Explikation klarstellen können. Das setzt aber voraus, dass die Paraphrasen selber nicht ambig sind und dass der Softwareentwickler linguistische Techniken erlernt, um die natürliche Sprache in einer kontrollierten Art und Weise zu verwenden.

Es lassen sich vier Formen von Ambiguitäten unterscheiden, die gleichzeitig in natürlich-sprachlichen Strukturen einer SAS auftauchen können: lexikalische Ambiguität, syntaktische Ambiguität, semantische Ambiguität und pragmatische Ambiguität.

Lexikalische Ambiguität

Schlägt man irgendein Lexem (= lexikalische Einheit) an einer Stelle in einem Wörterbuch nach, sieht man schnell, dass es mehr als einer Wortart angehören kann (kategoriale Ambiguität). Unterschiedliche grammatische Realisierungen eines Lexems können zu identischen Wortformenbildungen führen (morphologische Ambiguität). Darüber hinaus ist es möglich, dass ein Lexem mehr als eine Bedeutung hat, wobei man traditionellerweise zwischen Polysemie und Homonymie unterscheidet. Ein polysemes Lexem hat mehrere untereinander zusammenhängende Bedeutungen (*hand* als Zeiger einer Uhr oder Teil des menschlichen Arms), die alle in etymologischer Verwandtschaft stehen. Im Gegensatz dazu hat ein homonymes Lexem mehrere untereinander unverbundene Bedeutungen (*kitty* als (gemeinsame) Kasse und *kitty* als Katze), die sich auf etymologisch verschiedene Wurzeln zurückführen lassen. Homonyme Lexeme werden oft weiter unterteilt in solche mit absoluter Homonymie (*bank*) und solche mit partieller Homonymie (*find* - finden, *found* - gründen) [Lyons 95:54 ff., Greenbaum 96:428 ff.]. Bei der Grenzziehung zwischen Polysemie und Homonymie ist das etymologische Kriterium nicht immer schlüssig, das zeigt die lexikographische Praxis. Entscheidender in unserem Zusammenhang ist die Tatsache, dass polyseme und homonyme Lexeme eine geringe Vorkommenshäufigkeit in gleichen oder ähnlichen Kontexten haben. In natürlich-sprachlichen SASen wirkt in der Regel der fachsprachliche Kontext als Filter, der zu einer Monosemierung der potentiell mehrdeutigen Lexeme führt.

Syntaktische Ambiguität

Syntaktische oder strukturelle Ambiguität resultiert daraus, dass es für eine Wortkette mehr als eine mögliche sturkturelle Analyse geben kann. Strukturelle Ambiguität ist immer von einem bestimmten Grammatikmodell des Sprachsystems abhängig. Die drei wichtigsten Quellen von struktureller Ambiguität in der englischen Sprache sind Komposita (besonders Nominalkomposita), Anbindung (besonders Präpositionalphrasen-Anbindung) und Koordination [vgl. Hirst 87:131 ff., Gazdar & Mellish 89:172].

Komposita

Komposita sind in fachlichen und technischen Texten sehr verbreitet. In natürlich-sprachlichen SASen findet man eine intensive Nutzung von nominalen Wortbildungsmodellen, die den Wortschatz erweitern und spezialisieren, um dadurch Entitäten des jeweiligen Anwendungsgebiets zu erfassen. Die Normierung und begriffliche Systematisierung führt zu einer grossen Anzahl von Nominalkomposita (*user identification*, *staff user*, *bar code reader*). Es ist schwierig, Komposita vollständig zu analysieren, da die syntaktisch-semantische Beziehung zwischen Bestimmungs- und Grundwort häufig unterspezifiziert ist (vgl. *library copy* mit *library science*). Probleme entstehen nicht nur bei der Analyse von Komposita, sondern auch aus dem Zusammenspiel von Komposita und Wörtern aus anderen Kategorien. Beispielsweise ist *foreign bank manager* eine Wortsequenz, die auf zwei Arten gruppiert werden kann:

(23) *(foreign (bank manager))*

(24) *((foreign bank) manager)*

In (23) wird zuerst das Wort *foreign* als Adjektiv analysiert und anschliessend *bank manager* als Nominalkompositum bestimmt - eine richtige Entscheidung, wenn man sich einen Kontext vorstellt, indem der Bankmanager ein Ausländer ist. Die Analyse in (24) zeigt, dass eine weitere strukturelle Interpretation möglich ist, wenn man zuerst *foreign bank* als Kompositum und erst anschliessend *manager* als einfaches Nomen analysiert. Hier wird davon ausgegangen, dass es sich um eine Auslandsbank handelt und der Manager nicht weiter beschrieben wird.

Die häufigsten Strukturtypen von Komposita in der englischen Sprache sind in der nachfolgenden Tabelle in absteigender Ordnung zusammengefasst [Rackow 92:13].

Strukturtypen von Komposita

noun-noun:	brain death
adjective-noun:	abdominal layer
noun-of-noun:	letter of resignation
gerund-noun:	sleeping sickness
noun-preposition-noun:	thirst for adventure
participle-noun:	stewed apple
Saxon Genitive:	worker's revolt
verb-noun:	goggle-box
preposition-noun:	afterheat
noun-infinitive:	permission to work
adverb-noun:	now generation
verb-preposition:	dropout

Anbindung
Für gewisse syntaktische Konstituenten (z.B. Präpositionalphrasen, Relativsätze, Adverbien, Partizipien) ist es schwierig zu entscheiden, an welche dominierende Konstituente sie korrekterweise angebunden werden müssen. Beispielsweise kann die Präpositionalphrase *with the code* im Satz

(25) *The staff user enters the library card with the code.*

entweder an das vorausgehende Nomen *library card* oder an das Verb *enters* angebunden werden. Oftmals sind solche Mehrdeutigkeiten ausserhalb des Gebrauchskontextes nicht zu entscheiden. Um die passende Anbindung automatisch zu finden, benötigt man Selektionsbeschränkungen und Inferenz über Weltwissen [Allen 91:3] oder einen statistischen Ansatz [Hindle & Rooth 93:103 ff.]. Sequenzen von Präpositionalphrasen wie in

(26) *The staff user enters the library card with the code into the slot.*

vervielfältigen die Anzahl der möglichen Kombinationen, da die strukturelle Ambiguität multiplikativ ist. Ein noch grösserer Grad an Komplexität ergibt sich, wenn strukturelle Ambiguität und lexikalische Ambiguität ineinandergreifen [Gazdar & Mellish 89:173, Allen 95:177].

Koordination
Ähnliche strukturelle Probleme treten bei Koordination auf, wenn die Konstituenten durch Konjunktionen verschiedenartig gruppiert werden können. Beispielsweise ist in

(27) *The staff user selects the buttons A and B or C.*

nicht klar, welche Konjunktion dominiert. Es sind die folgenden zwei Lesarten denkbar:

(28) *The staff user selects the buttons ((A and B) or C).*

(29) *The staff user selects the buttons (A and (B or C)).*

Formale Spezifikationssprachen lösen diese Art von struktureller Ambiguität durch explizite Klammerung der Ausdrücke oder durch Operatordeklarationen.

Semantische Ambiguität
Wenn in einem natürlich-sprachlichen Satz zwei oder mehr quantifizierte Nominalphrasen auftreten, dann lassen sich bei identischer Syntaxstruktur aufgrund der

Skopusalternativen verschiedene Interpretationen ableiten (vgl. (10)). Die korrekte Interpretation der quantifizierten Nominalphrasen hängt von unterschiedlichen Informationsquellen ab, zum einen vom Erfahrungswissen über die beschriebenen Sachverhalte und zum anderen von der Bindungsstärke der einzelnen Determinatoren [vgl. Chierchia & McConnell-Ginet 90:128 ff., Covington 94a:213]. Die Bindungsstärke schafft eine Präzedenzhierarchie unter den Determinatoren und macht eine Aussage darüber, welche quantifizierten Nominalphrasen weiten Skopus haben. In der Literatur wird oft die folgende Präzedenzhierarchie als Arbeitshypothese vorgeschlagen [Allen 95:355]:

each > every > all, some, several, a

Skopusentscheidungen können sehr komplex sein. Im Beispielsatz (30) muss die universell quantifizierte Nominalphrase *every book* für die korrekte Interpretation aus ihrem lokalen Bereich angehoben werden, so dass sie weiten Skopus über die dominierende definite Nominalphrase nehmen kann [vgl. Allen 95:358].

(30) *The ISBN number in every book consists of a ten-digit number.*

Geschieht das nicht, ergibt sich eine Lesart, bei der nur eine ISBN Nummer existiert, die in jedem Buch abgedruckt ist. Die Erfahrung sagt uns, dass die Wirklichkeit nicht so aussieht, sondern dass jeder Buch(-titel) seine eigene ISBN Nummer hat. Determinatoren führen nicht nur zu Skopusambiguitäten, sondern sie unterscheiden sich auch hinsichtlich ihrer Präzision. Der Determinator *every* ist sehr präzis, da Nominalphrasen, die ihn enthalten, alle Elemente einer Menge denotiert. Der Determinator *most* hingegen ist vage, denn Nominalphrasen mit diesem Determinator denotieren eine nicht genau präzisierbare Menge:

(31) *Most staff users have a password.*

Weitere diffizile Skopusbeziehungen entstehen aus dem Zusammenspiel von Determinatoren und logischen Operatoren. Beispielsweise hat der Satz

(32) *A staff user does not have a password.*

zwei Lesarten. Die erste Lesart kann durch den Satz

(33) *It is not the case that there exists some staff user who has a password.*

paraphrasiert werden und die zweite Lesart durch den Satz

(34) *It is the case that there exists some staff user who does not have a password.*

Ausser Determinatoren und Operatoren führen auch Adverbien zu Skopusproblemen. Der Satz

(35) *A staff user always loses the password.*

hat zwei Lesarten: Entweder ist es immer irgend jemand aus der Belegschaft, der das Passwort verliert, oder es ist ein ganz bestimmter Mitarbeiter, der das Passwort ständig verliert.

Pragmatische Ambiguität

Pragmatische Ambiguität tritt bei anaphorischen Verweisen auf. Bei Anaphern handelt es sich um Ausdrücke, die innerhalb eines Satzes oder Textes auf Antezedenzien (= vorausgehende sprachliche Ausdrücke) Bezug nehmen. Anaphern als Verweisformen mit indirekter Referenz sind unter anderem Pronomen und definite Nominalphrasen. Notorisch schwierig ist die eindeutige Lokalisierung der passenden Antezedenzien für die automatische Sprachverarbeitung [Smith 91:407 ff.]. Die folgenden Beispielsätze verdeutlichen das Problem:

(36) *A staff user enters the user identification and the password.*

(37) *If it is correct then LibDB is ready for the check-out.*

(38) *If the user interrupts the input then LibDB displays the message 'Stop'.*

(39) *If not then LibDB displays the message 'Continue'.*

Im Satz (37) ist nicht eindeutig, worauf sich das Personalpronomen *it* bezieht. Es stehen zwei Antezedenzien im vorausgehenden Satz (36) zur Auswahl: *the user identification* oder *the password*. Nicht klar ist, ob eine Referenzidentität zwischen der definiten Nominalphrase *the user* in (38) und der indefiniten Nominalphrase *a staff user* in (36) besteht. Ausserdem kann für Satz (38) nur mit Hilfe von Weltwissen entschieden werden, dass sich die definite Nominalphrase *the input* auf das Ereignis in (36) bezieht. Auch für den elliptischen Ausdruck *not* in (39) ist der Bezug unklar, denn entweder negiert *not* die Information im Vorderglied von Satz (37) oder von Satz (38).

2.3.2 Vorteile von natürlichen Sprachen als Spezifikationssprachen

Natürliche Sprachen als Spezifikationssprachen sind als primäres Kommunikationsmittel für alle Beteiligten verständlich. Die Anwendungsspezialisten können ihre Begriffe direkt in der vertrauten fachsprachlichen Terminologie niederschreiben und brauchen sie nicht in eine formale Sprache zu kodieren [vgl. Sommerville 96:122 ff.].

SASen in natürlicher Sprache sind gut lesbar und verständlich. Wenn die Bedeutung der Terminologie in einem Glossar gesichert ist, sind die Fachausdrücke im Prinzip auch für die Softwareentwickler zugänglich. Allerdings ist das eine verkürzte Sichtweise, denn eine Fachsprache lässt sich nicht auf den Sprachgebrauch reduzieren, sondern muss durch kognitive Leistungen erworben werden. Ein fachsprachlicher Begriff ist nicht eine Vokabel, die man kennt oder nicht kennt, sondern es ist ein Gebilde, das man sich durch Vergleichen, Folgern und Problemlösen aneignen muss. Eine Fachsprache erlernen und verstehen, heisst immer auch an einer Einführung in ein Fach teilnehmen [Lewandowski 94:295].

Wenn eine SAS in natürlicher Sprache geschrieben ist, dann ist das Dokument unmittelbar für die Review zugänglich und kann auf Schwachstellen und Mängel hin untersucht werden. Die Review ist ein formaler und strukturierter Analyse- und Bewertungsprozess, in dem die SAS durch eine Gruppe von Anwendungsspezialisten und Softwareentwicklern nach vorgegebenen Prüfkriterien untersucht wird [vgl. Schach 96:114 ff.]. Regelmässige Reviews garantieren den Wissenstransfer zwischen den Beteiligten. Inspektion ist das einzige Mittel, das den Gutachtern erlaubt, die Anforderungen aktiv zu validieren, wenn die SAS in natürlicher Sprache geschrieben ist. Bei einer Inspektion wird die SAS Zeile für Zeile kritisch gelesen, um so festzustellen, ob die einzelnen Anforderungen den Qualitätsattributen genügen. Für die Durchführung der Inspektion leisten Checklisten wertvolle Hilfe:

> As you read a piece of software description, you should be constantly questioning what you read. What kind of description is it? What is it about? What is it for? Does it assert that something is *true*? Then how might it be proved wrong? Or does it state that something is *required*? By whom? How could I check that it really is required? What does the description assume? What does it leave out? Is there something else I must know to understand it? Could it have more than one meaning? How can I check my understanding? Does it define new terms to be used elsewhere? How does it fit with the other description in the same development? [Jackson 95:39 ff.].

2.3.3 Nachteile von natürlichen Sprachen als Spezifikationssprachen

Ambiguität und Vagheit sind ein unvermeidbarer Grundzug von SASen, die in natürlicher Sprache geschrieben sind. Es ist hoffnungslos, sich auf die volle natürliche Sprache als Spezifikationssprache zu verlassen, da die SAS eindeutig und konsistent sein muss. Was Not tut, sind Prinzipien und Werkzeuge, die die Autoren anleiten und unterstützen, die natürlich-sprachlichen Anforderungen in der SAS eindeutig zu verfassen.

> More research must be done to provide natural-language writers with insight into how they can improve their documents without formal models. Although user needs truly are difficult to determine, we still need guiding principles to reduce the ambiguity inherent in a natural-language document when it is written [Hsia et al. 93:77].

Die volle natürliche Sprache ist als Spezifikationssprache zu flexibel: Sie verleitet zu Redundanz, Unter- und Überspezifikation beim Schreiben der SAS [vgl. Meyer 85:7 ff.]. Gefährlich wird Redundanz dann, wenn derselbe Sachverhalt mehrmals durch unterschiedliche sprachliche Mittel dargestellt wird und das dem Leser verborgen bleibt. Bei Unterspezifikation verlassen sich die Autoren auf gemeinsame aussersprachliche Wissensbestände und überlassen dann die Herstellung von Kohärenz dem Leser der SAS, der die erforderliche Textarbeit aufgrund seines Fach- und Erfahrungswissens leisten muss. Bei Überspezifikation unterlassen es die Autoren, aufgrund ihrer Vertrautheit mit der natürlichen Sprache die deklarative Problembeschreibung und die prozedurale Problemlösung sauber auseinanderzuhalten.

Um eine SAS in voller natürlicher Sprache automatisch weiter zu verarbeiten, müsste das Fach- und Erfahrungswissens erst in einer formalen Sprache modelliert werden können, bevor es für ein natürlich-sprachliches System zugänglich ist [vgl. Lenat & Guha 90]. Die Modellierung von Fach- und Erfahrungswissen ist ein ausgesprochen komplexes Problem, da einerseits die Festlegung einer adäquaten Repräsentationsebene schwierig ist und andererseits die Auswahl der darzustellenden Information bereits zu einer ersten Interpretation des Anwendungsgebiets führt.

2.4 Eigenschaften von formalen Sprachen als Spezifikationssprachen

2.4.1 Übersicht

Um Mehrdeutigkeit, Inkonsistenz und Unvollständigkeit bei der Anforderungsspezifikation besser in den Griff zu bekommen, sind anstelle der Verwendung von natürlicher Sprache formale Spezifikationssprachen vorgeschlagen worden, die in formale Methoden eingebunden sind [Meyer 85, Hall 90, Pohl 93, Wing 94, Bowen & Hinchey 95, Larsen et al. 96, Clarke & Wing 96]. Formale Methoden basieren auf mathematischen Sprachen und umfassen Techniken und Werkzeuge, die für das Schreiben von formalen Spezifikationen, das Beweisen von Eigenschaften von formalen Spezifikationen sowie für die Programmkonstruktion und Verifikation eingesetzt werden [Hall 90:13]. Eine Methode ist formal, wenn sie eine gültige mathematische Grundlage hat, die bei der Definition der Spezifikationssprache zum Ausdruck kommt.

Eine formale Spezifikationssprache kann als eine Relation (= Interpretation) zwischen einer syntaktischen Domäne (= Notation) und einer semantischen Domäne (= Diskursbereich) definiert werden.

> Definition: A formal specification language is a triple, <*Syn*,*Sem*,*Sat*>, where *Syn* and *Sem* are sets and $Sat \subseteq Syn \times Sem$ is a relation between them. *Syn* is called the language's syntactic domain; *Sem*, its semantic domain; and *Sat*, its satisfies relation.
> Definition: Given a specification language, <*Syn*,*Sem*,*Sat*>, if *Sat(syn,sem)* then *syn* is a specification of *sem*, and *sem* is a specificand of *syn*.
> Definition: Given a specification language, <*Syn*,*Sem*,*Sat*>, the specificand set of a specification *syn* in *Syn* is the set of all specificands *sem* in *Sem* such that *Sat(syn,sem)* [Wing 94:504].

Diese Definitionen entsprechen der klassischen modelltheoretischen Methode für die Interpretation von formallogischen Sprachen. Eine Spezifikation wird als eine wohlgeformte Formel verstanden, die aus den Elementen der syntaktischen Domäne gebildet wird. Die Formel denotiert eine Teilmenge von Entitäten aus der semantischen Domäne. Für die Interpretation der syntaktischen Elemente ist die Relation *Sat(syn, sem)* verantwortlich, die die Elemente der Sprache zu den Entitäten im Diskursbereich in Beziehung setzt. Wenn eine syntaktische Formel in einer Relation (= Interpretation) wahr ist, so ist diese Interpretation ein Modell dieser Formel. Ein Modell existiert genau dann, wenn es zu einer Formel - die einen Sachverhalt beschreibt - eine Interpretation gibt, so dass der Wahrheitswert dieser Formel *wahr* ist.

Eine Formel kann mehrere Modelle haben, doch gewöhnlich hat der Softwareent-

wickler beim Schreiben einer formalen Spezifikation eine intendierte Interpretation im Kopf und assoziiert die syntaktische Formel mit einem ganz bestimmten Sachverhalt im Diskursbereich [vgl. Hogger 90:27].

Die einzelnen Elemente aus der syntaktischen Domäne einer formalen Sprache brauchen nicht unbedingt textuell zu sein. Solange die Interpretation eindeutig ist, kann graphischen Elementen genausogut ein formale Semantik gegeben werden. Eine formale Spezifikation ist dann eindeutig, wenn jedes syntaktische Element auf genau eine Entität aus dem Diskursbereich abgebildet werden kann:

> Definition: Given a specification language, <*Syn,Sem,Sat*>, a specification *syn* in *Syn* is unambiguous if and only if *Sat* maps *syn* to exactly one specificand set [Wing 94:506].

Neben der Eindeutigkeit spielt die Konsistenz eine zentrale Rolle bei der Definition von formalen Spezifikationssprachen. Denn die Konsistenz einer formalen Spezifikation ist Voraussetzung dafür, dass nichts Widersprüchliches aus ihr abgeleitet werden kann.

> Definition: Given a specification language, <*Syn,Sem,Sat*>, a specification *syn* in *Syn* is consistent (or satisfiable) if and only if *Sat* maps *syn* to a non-empty specificand set [Wing 94:506].

Die meisten formalen Spezifikationssprachen sind zusammen mit einer Modelltheorie und einer Beweistheorie definiert. Die Modelltheorie untersucht die Relationen zwischen syntaktischen Formeln und dem sprachexternen Diskursbereich und führt den Begriff der logischen Konsequenz ein. Wenn jede Relation (= Interpretation), die Modell von M ist, auch Modell von F ist, so sagt man, dass F eine logische Konsequenz aus der Formelmenge M ist, geschrieben: $M \models F$. Im Gegensatz zur Modelltheorie interessiert sich die Beweistheorie für die syntaktische Ableitbarkeit von neuen wohlgeformten Formeln (= Theoremen) aus vorgegebenen Formeln (= Axiomen) unter Verwendung von Schlussregeln (= Inferenzregeln). Ein Theorem F ist ableitbar, wenn es unter Anwendung der Inferenzregeln aus einer Axiomenmenge M in endlich vielen Schritten bewiesen werden kann, geschrieben: $M \vdash F$. Eine logische Beweismethode sollte korrekt und vollständig sein. Korrekt ist sie dann, wenn für jede Axiomenmenge M jedes ableitbare Theorem F auch eine logische Konsequenz der Axiome ist. Umgekehrt ist eine Beweismethode vollständig, wenn für jede Axiomenmenge M jedes Theorem F, das aus den Axiomen folgt, auch ableitbar ist. Sind beide Forderungen erfüllt, dann schreibt man: $M \vdash F \equiv M \models F$ [vgl. Fuchs 90:116].

Die Gültigkeit eines Beweises hängt ausschliesslich von der syntaktischen Struktur der Formeln ab und kommt vollständig mechanisch - ohne die Verwendung von spezifischem Weltwissen - zustande.

> Any attempt to encode real-world requirements in the computer must ultimately entail their translation into symbols and hence only contribute further to the computer´s internal syntactic world [Hogger 90:55].

Schlussendlich sind für einen Computer immer nur syntaktische Symbole verfügbar, deren Bezug zur Welt erst durch eine intendierte Interpretation hergestellt wird.

Die Axiome zusammen mit allen durch die Inferenzregeln abgeleiteten Theoremen bilden eine Theorie des Anwendungsbereichs.

Mit der Hilfe einer Beweismethode, die auf einer Menge von Inferenzregeln und einer Suchstrategie beruht, kann ein Programm auf Konsistenz und Korrektheit (hinsichtlich einer formalen Spezifikation) geprüft werden. Werkzeuge wie Theorembeweiser oder Modellchecker unterstützen den Softwareentwickler beim Nachweisen, ob bestimmte Eigenschaften aus der Spezifikation ableitbar sind, oder helfen beim Überprüfen, ob beispielsweise ein vorliegendes System mit den Eigenschaften eines endlichen Automaten ein Modell für die Spezifikation ist [Clarke & Wing 96:5 ff.]. Darüber hinaus wird es möglich, bestimmte Klassen von Fehlern leichter zu finden oder durch formalisierte Fragen intendierte Interpretationen zu überprüfen.

Insbesondere eignen sich Spezifikationen, die in einer formalen Sprache geschrieben sind, für die automatische Weiterverarbeitung. Liegt der formalen Sprache ein operationales Modell zugrunde, dann kann die Vollständigkeit der Spezifikation durch den Anwendungsspezialisten im Experiment überprüft werden. Falls ein transformationsbasiertes Software-Entwicklungsparadigma verfolgt wird, kann durch die Anwendung einer Reihe von korrektheitsbewahrenden Transformationen aus einer formalen Spezifikation sogar ein lauffähiges Programm erzeugt werden [Balzer 85:1257 ff., Fromherz 93:16].

Logikprogrammierung - Prolog ist die bekannteste Vertreterin - bietet dem Softwareentwickler ein Werkzeug an, Logik als formale Sprache zur Beschreibung von Sachverhalten zu verwenden. Eine Logik-Programmierumgebung besteht aus einer Logikkomponente, in der das Wissen über das zu lösende Problem spezifiziert wird, und einer Kontrollkomponente, die ein Lösungsverfahren für das Problem aufgrund einer

vorgegebenen Suchstrategie (z.B. Tiefensuche) und einer Inferenzmethode (z.B. SLD-Resolution) liefert [Lloyd 87:40]. Die Kontrollkomponente bestimmt die prozedurale Semantik eines Logikprogramms.

Logikprogrammierung ist Programmierung durch Beschreibung. Der Softwareentwickler kodiert Fakten und Regeln, die idealerweise in Form von deklarativen oder konditionalen Sätzen in einer SAS vorliegen, in Hornklauseln, einer Teilmenge der Prädikatenlogik. Die Klauseln sind eine Menge von Axiomen, die Relationen zwischen Entitäten aus dem Anwendungsgebiet festlegen. Anfragen an ein logisches Programm (≈ Spezifikation) können als Theoreme formuliert werden und dann mittels der applikationsunabhängigen Inferenzprozedur durch deduktive Folgerungen aus den Axiomen bewiesen werden. Während des Beweises werden Variablen, die in den Anfragen auftauchen, durch Substitutionen an Werte gebunden, die als Rechenergebnisse betrachtet werden können.

> A logic program is a set of axioms, or rules, defining relations between objects. A computation of a logic program is a deduction of consequences of the program. A program defines a set of consequences, which is its meaning. The art of logic programming is constructing concise and elegant programs that have the desired meaning [Sterling & Shapiro 94:9].

Im Idealfall sollte sich der Softwareentwickler beim Schreiben eines logischen Programms beziehungsweise einer formalen Spezifikation keine Gedanken über die Kontrollkomponente (prozedurale Semantik) machen müssen, sondern sich ausschliesslich auf die Darstellung von wahren und bedingten Sachverhalten (modelltheoretische Semantik) konzentrieren können. Diese ideale Trennung ist jedoch bei den existierenden Logik-Programmierumgebungen nicht gegeben, da die prozedurale und modelltheoretische Semantik nicht in jedem Fall identisch sind. Es kann Lösungen geben, die aufgrund der Suchstrategie nicht gefunden werden, obwohl sie logische Konsequenz des Programms sind. Man spricht dann davon, dass die prozedurale Semantik unvollständig ist. Der Softwareentwickler muss deshalb die Reihenfolge der Klauseln sorgfältig planen und die Suchstrategie in Betracht ziehen.

Es ist interessant zu sehen, dass die Logikprogrammierung und die ihr zugrundeliegende mathematische Logik die Unterscheidung zwischen Programm und formaler Spezifikation verwischt, so dass viele logische Programme als ausführbare Spezifikationen aufgefasst werden können. Der einzige Unterschied zwischen Programm und Spezifikation liegt darin, dass das Programm gewöhnlich effizienter ist als die Spezifikation [Kowalski 85:11].

Spezifikationen, die in einer logischen Programmiersprache geschrieben sind, stellen nicht nur ein konzeptionelles Modell eines geplanten Softwaresystems dar, sondern auch ein Verhaltensmodell. Die Verfügbarkeit von ausführbaren Spezifikationen macht es für die Anwendungsspezialisten möglich, die Anforderungen in einer frühen Phase im Software-Entwicklungsprozess zu validieren und nicht erst nachdem bereits Entwurfsentscheidungen gefällt worden sind. Diese formale ausführungsorientierte Vorgehensweise kann die Herstellungskosten für ein Softwareprodukt erheblich senken, verlangt aber einen zeitlichen Mehraufwand bei der Anforderungsspezifikation. Der Aufwand ist sicher gerechtfertigt, wenn man sich vor Augen führt, dass die Behebung eines Fehlers, der sich während der Anforderungsspezifikation einschleicht und erst nach der Auslieferung des Softwareproduktes bemerkt wird, in der klassischen Softwareentwicklung bis zu 100 Mal teurer zu stehen kommt als die Beseitigung eines Fehlers im Programmcode [Sommerville 96:71].

2.4.2 Vorteile von formalen Sprachen als Spezifikationssprachen

Auf den ersten Blick scheinen formale Sprachen - und insbesondere das Paradigma der Logikprogrammierung - eine Anwort auf eine Vielzahl von Problemen bei der Anforderungsspezifikation zu geben. Formale Spezifikationen sind präzis und eindeutig. Das sind wichtige Voraussetzungen, die erst die formale Verifikation von Darstellungen unterschiedlicher Abstraktionsebenen durch Beweise ermöglichen. Beispielsweise kann man ein Programm mit seiner formalen Spezifikation vergleichen und beweisen, dass das Programm eine gültige Implementation der Spezifikation ist. Obwohl die vollständig formale Verifikation von komplexen Softwaresystemen gegenwärtig technisch nicht machbar ist, kann die Verifikation von sicherheitskritischen Programmteilen das Vertrauen in die Zuverlässigkeit des Softwareproduktes erhöhen [Hall 90:16, Sommerville 96:163]. Zusätzlich zur Verifikation erlauben formale Methoden die Konsistenz von formalen Anforderungsspezifikationen zu überprüfen, um Konflikte zwischen den einzelnen Anforderungen auszuschliessen. Beides, Verifikation und Konsistenzprüfung, sind rein mechanische Aufgaben, die automatisierbar sind. Die Lösung der Aufgabe ist ausschliesslich von den Eigenschaften der formalen Sprache und von der Grösse der formalen Anforderungsspezifikation abhängig.

2.4.3 Nachteile von formalen Sprachen als Spezifikationssprachen

Auf den zweiten Blick haben formale Methoden einen schwerwiegenden Nachteil: Sie sind für Anwendungsspezialisten kaum verständlich. Obwohl die Verwendung von formalen Sprachen wie Z [Spivey 89], VDM [Jones 86], Larch [Guttag & Horning 93],

Petri-Netze [Peterson 81], Statecharts [Harel 87], logische Programmiersprachen [Lloyd 94] anstelle von voller natürlicher Sprache die Präzision der Anforderungsspezifikation erhöht und Mehrdeutigkeit ausschliesst, geht das fast immer zu Lasten der guten Lesbarkeit. Das Schreiben und Lesen von formalen Spezifikationen erfordert Kenntnisse der mathematischen Grundlagen (diskrete Mathematik, Logik), mit denen viele Anwendungsspezialisten nicht vertraut sind. Die Anwendungsspezialisten können deshalb eine formale SAS, nicht unmittelbar durch Reviews validieren, um festzustellen, inwieweit die formalisierten Anforderungen ihr Verständnis der Sachverhalte wiedergibt. Hinzu kommt, dass formale SASen oftmals durch bereichsfremde Softwareentwickler geschrieben werden, die nicht das gleiche Sachwissen wie die Anwendungsspezialisten haben. Falls eine SAS unvollständig ist, sind die Softwareentwickler gezwungen, Annahmen über das intendierte Systemverhalten zu treffen. Das kann zu Missverständnissen und fehlerhaften Spezifikationen führen. Da formale Spezifikationen bloss Abbildungen der wirklichen Welt sind, können sie natürlich auch Sachverhalte falsch beschreiben, was zu unzulässigen Schlussfolgerungen führt. Selbst wenn die formale Spezifikation korrekt ist, kann die verwendete Beweisprozedur fehlerhaft sein.

> The real world is not a formal system. A proof, therefore, does not show that, in the real world, things will happen as you expect. So you can never be sure that your specifications are "correct", however much you prove about them [Hall 90:12].

Ausserdem ist es sehr schwierig, eine formale Spezifikation von informellen Anforderungen abzuleiten, weil der Ableitungsprozess selber weder formalisierbar noch formal validierbar ist [Hoare 87:372]. Unverhofft kommt die natürliche Sprache wieder zur Hintertür herein, wenn es darum geht, die formale Spezifikation in natürlicher Sprache zu paraphrasieren, um dem Anwendungsspezialisten verständlich zu machen, *what the specification means in real-world terms and why the specification says what it does* [Hall 90:18]. Um die nötige Akzeptanz bei den Anwendungsspezialisten zu erreichen und den Zugang zu formalen Spezifikationen zu erleichtern, empfiehlt die Z Schule, formale Spezifikationen massiv mit natürlich-sprachlichen Kommentaren anzureichern [Clarke & Wing 96:14]. Dabei machen sich die Autoren aber keinerlei Gedanken über die Interdependenzen zwischen informellen Kommentaren und formalem Spezifikationstext.

Die Einführung von formalen Sprachen führt zu enormen Schwierigkeiten, wenn es nicht gelingt, adäquate Darstellungen zu finden, die auch für den Anwendungsspezialisten in eindeutiger Weise verständlich sind.

2.5 Ein Spezifikationsbeispiel

2.5.1 Das Bibliotheksproblem

Das Bibliotheksproblem (engl. *library problem*) ist eine von vier Problembeschreibungen, die die Organisatoren des *Fourth International Workshop on Software Specification and Design* im Publikationsaufruf als Ausgangspunkt für die Beiträge vorgelegt hatten [Marca 87, vgl. Kemmerer 85].

Library Problem

Consider a small library database with the following transactions:

1. Check out a copy of a book / Return a copy of a book.
2. Add a copy of a book to / Remove a copy of a book from the library.
3. Get the list of books by a particular author or in a particular subject area.
4. Find out the list of books currently checked out by a particular borrower.
5. Find out what borrower last checked out a particular copy of a book.

There are two types of users: staff users and ordinary borrowers. Transactions 1, 2, 4 and 5 are restricted to staff users, except that ordinary borrowers can perform transaction 4 to find out the list of books currently borrowed by themselves. The data base must also satisfy the following constraints:

- All copies in the library must be available for checkout or be checked out.
- No copy of the book may be both available and checked out at the same time.
- A borrower may not have more than a predefined number of books checked out at one time.

Im Workshop-Band sind zwölf Beiträge veröffentlicht worden, die das Bibliotheksproblem aus unterschiedlichen Blickwinkeln angegangen sind. Die Beiträge decken einen breiten Themenbereich ab und fokussieren verschiedene Aspekte im Software-Entwicklungsprozess. Hauptsächlich ging es den Autoren darum, die in ihrer Forschungsarbeit verfolgten Spezifikationsmethoden und entwickelten Spezifikationssprachen (fragmentarisch) auf das Bibliotheksproblem anzuwenden. Zusammengenommen liefern die Beiträge eine Fundgrube von Ungenauigkeiten, auf die die Autoren beim Versuch gestossen sind, die informelle Problemspezifikation mit ihren eigenen Methoden zu verfeinern und in eine detailliertere informelle Spezifikation

oder in eine formale Spezifikation zu überführen. Einen Überblick über die verschiedenen Ansätze gibt [Wing 88:70]. Wir interessieren uns hier speziell für drei Lösungsvorschläge, die zum Ziel haben, die informelle Problembeschreibung in eine formale Spezifikation zu überführen und diskutieren die drei Beiträge von [Rich et al. 87, Lee & Sluizer 87, Wing 87] eingehender.

Die Problembeschreibung besteht aus drei Teilen:

- Fünf Transaktionen, die durch die Datenbank unterstützt werden.
- Zwei Restriktionen, die auf die Transaktionen angewendet werden.
- Drei Restriktionen, denen die Datenbank genügen muss.

Die fünf Transaktionen beschreiben grundlegende Tätigkeiten in einer Bibliothek. Die zwei Transaktionsrestriktionen betreffen die beiden Benutzerklassen: Mitarbeiter *(staff user*) und Ausleiher (*ordinary borrower*). Die Mitarbeiter dürfen sämtliche Transaktionen ausführen. Für die Ausleiher sind nur zwei Transaktionen möglich: Die 3. Transaktion ist für sie unbeschränkt zulässig und die 4. Transaktion ist dann zulässig, wenn sie individuelle Informationen beziehen. Die drei Datenbankrestriktionen machen Aussagen über die Verfügbarkeit der Buchkopien in der Bibliothek und über die maximale Anzahl der Bücher, die von einem Ausleiher zu einem Zeitpunkt ausgeliehen werden können.

Auf den ersten Blick erscheint das Bibliotheksproblem täuschend einfach. Die harten Probleme tauchen dann auf, wenn die natürlich-sprachliche Spezifikation auf Mehrdeutigkeit und Vagheit abgeklopft wird. Was bedeutet *small* im nominalen Ausdruck *small library database* oder was versteht man unter *last checked out*?

Bevor die einzelnen Lösungsvorschläge diskutiert werden, soll darauf hingewiesen werden, dass Kemmerers Originalbeschreibung des Bibliotheksproblems in drei Punkten vom vorliegenden Publikationsaufruf abweicht [Kemmerer 85:32 ff.]. Einige der Autoren waren mit der Originalbeschreibung vertraut und berücksichtigten den einen oder anderen der drei untenstehenden Punkte für ihre eigenen Lösungsvorschläge. Beispielsweise sprechen [Reubenstein et al. 91:231] von einer Datenbank für eine Universitätsbibliothek (*university library database*) und [Wing 87:37] behandelt eine vierte Datenbankrestriktion, die ausschliesst, dass ein Ausleiher eine zweite Kopie eines Buches bezieht.

Ausschnitt aus der Originalbeschreibung [Kemmerer 85]

- The example system is a university library database.
- Check out a copy of a book / Return a copy of a book are not restricted to only staff users.
- A borrower may not have more than one copy of the same book checked out at one time.

2.5.2 Lösungsvorschläge

Toward a Requirements Apprentice [Rich et al. 87]

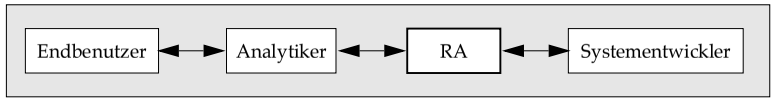
Die Autoren stellen mit dem Requirements Apprentice (RA) ein Werkzeug vor, das den Übergang von einer informellen Problembeschreibung (z.B. Bibliotheksproblem) zu einer formalen Spezifikation unterstützt. Der Endbenutzer verwendet das Werkzeug nicht direkt. Die Vorgehensweise ist so, dass ein Analytiker zusammen mit den Endbenutzern die Problembeschreibung während der Anforderungsanalyse festlegt. Ausgehend von dieser noch unpräzisen Problembeschreibung überführt der Analytiker die einzelnen Anforderungen mit Hilfe des RA schrittweise in eine formale Repräsentation. Zu diesem Zweck verwendet der Analytiker eine formale Kommandosprache. Grundlegend ist, dass dem RA bereits zu Beginn des Spezifikationsprozesses eine umfassende Wissensbasis über fach- und prozessspezifische Informationen in einer Cliché-Bibliothek vorliegt.

> The *cliché library* is a declarative repository of information relevant to requirements in general and to domains of particular interest. When creating a requirement, the information unique to the particular problem comes from the analyst. However the bulk of general information about the domain comes from the cliché library [Reubenstein & Waters 91:228].

Konzeptionell besteht ein Cliché aus einer Menge von Rollen und Constraints. Die Rollen sind diejenigen Teile eines Clichés, die sich je nach Verwendungszweck ändern. Die Constraints bestimmen, wie die Rollen zusammenspielen und schränken diejenigen Entitäten ein, die verwendet werden können, um die Rollen zu füllen. Der RA nutzt Schlüsselkonzepte aus der KI wie abhängigkeitsbezogenes Schliessen, hybride Wissensrepräsentation und Wiederverwendung für die Verarbeitung, Darstellung und Organisation der Clichés.

Während des Spezifikationsprozesses werden die Anforderungen laufend verfeinert. Das Endprodukt ist eine maschinell weiterverarbeitbare Repräsentation der Anforderungen. Aus dieser formalen Repräsentation kann entweder eine natürlich-sprachliche SAS für Validierungszwecke generiert oder eine formale Spezifikation abgeleitet werden, die dem Systementwickler als Ausgangspunkt für den Systementwurf dient. Zwischen dem Analytiker und dem RA besteht eine strikte Arbeitsteilung: Der Analytiker leitet den Spezifikationsprozess, und der RA überprüft die Anforderungen fortlaufend.

Arbeitsteilung



Die entstehende Repräsentation wird einerseits auf Konsistenz geprüft, indem der RA nach Widersprüchen sucht, und andererseits auf Vollständigkeit hinsichtlich der vorliegenden Wissensbasis. Der RA disambiguiert und verfeinert die Anforderungen durch konservative und heuristische Klassifikation [Reubenstein & Waters 91:234].

Der Analytiker gibt seine Anforderungen dem RA in Form von Lisp Ausdrücken ein. Für das Bibliotheksbeispiel untersucht der Analytiker mit dem Ausdruck *Find-Requirement* in einem ersten Schritt, ob der RA bereits eine Anforderung *Library-System* kennt.

```
1: (Find-Requirement Library-System)
Beginning-A-New-Requirement-Called-The Library-System.
Library-System Is-An-Instance-Of Requirement.
```

Da das nicht der Fall ist, wird eine neue Wissensbasis mit dem Namen *Library-System* bereitgestellt, in der alle entstehenden Informationen abgespeichert werden [vgl. Reubenstein & Waters 91:231 ff.]. Anschliessend definiert der Analytiker den Namen des Systems, das spezifiziert wird:

```
2: (Define University-Library-Database : System : Synonym Uldb)
```

Der zu definierende Systemname *University-Library-Database* folgt dem *Define* Befehl als erstes Argument. Abgetrennt durch Doppelpunkte schliessen sich die

beiden Schlüsselwörter *System* und *Synonym* an. Das erste Schlüsselwort gibt den Typ des Wortes an und das zweite Schlüsselwort zeigt an, dass es sich bei *Uldb*, um ein Synonym handelt. Ähnlich wird das Wort *University-Library* definiert:

```
3: (Define University-Library : Environment : Library : Synonym Ul)
Ul Is-An-Instance-Of "Library"
```

Da der RA noch keine Information darüber verfügt, was eine Bibliothek ist, erscheint das Wort *"Library"* für den Analytiker in der Ausgabe in Anführungszeichen. Daraufhin definiert er dieses Wort genauer:

```
6: (Define Library : Ako Repository : Defaults (: Collection-Type Book))
Ul.Collection-Type Has-Value "Book"
```

Library ist eine Art von Repository (*Ako Repository*), welches eine Sammlung von Büchern (*Collection-Type Book*) enthält. Der RA kennt das Konzept *Repository* aufgrund eines abstrakten Repository Clichés. Offen bleibt die Definition für das Wort *"Book"*. Der Analytiker definiert *Book* als eine Art von physikalischem Objekt (*Ako Physical-Object*), dem die drei Rollen *Title*, *Author* und *Isbn* zukommen:

```
7: (Define Book : Ako Physical-Object : Member-Roles (Title Author Isbn))
```

Nachdem der Analytiker die grundlegenden Objekte definiert hat, beginnt er die funktionalen Anforderungen zu spezifizieren. Zum Beispiel definiert er die untenstehende Transaktion *Check-Out* aufgrund der abstrakten Rolle *Remove*, die eine Standardoperation darstellt. Da *Uldb* im Verlauf des Spezifikationsprozesses als ein spezieller Typ von Informationssystem (*Tracking-Information-System*) definiert wurde, für das ein eigenes Cliché (*Action-Tracking-Operation* Cliché) zur Verfügung steht, kann die Transaktion *Check-Out* spezialisiert werden. Zusammen mit der aktuellen Wissensbasis des RA wird es möglich, neue Information abzuleiten:

```
9: (Define Check-Out : Roles (: Records Remove))
Uldb Is-An-Instance-Of Tracking-Information-System.
Uldb.Manner-Of-Observation Is-An-Instance-Of Indirect-Observation.
Check-Out Is-An-Instance-Of Action-Tracking-Operation.
Check-Out.Object-Type Has-Value Book.
Check-Out.Objects Has-Value (!The (?0) Such-That (= (Isbn ?0) $Input)).
Check-Out.Records Has-Value Remove-Repository. ...
```

Der Analytiker kann sich vom RA zu jedem Zeitpunkt einen natürlich-sprachlichen Report generieren lassen, der Auskunft über den aktuellen Zustand der SAS gibt. Nach einigen weiteren Spezifikationsschritten ist die folgende Zusammenfassung verfügbar:

> The environment of the library-system requirement is the University-Library (UL). The UL is a library. A library is a repository for books.
> The system being specified is the University-Library-Database (ULDB). The ULDB is a tracking-information system, which tracks the state of the (UL). Three transactions and one report are specified.
> The library-system requirement is inconsistent and incomplete. There are five pending issues and one unresolved conflict.

Ausser diesem natürlich-sprachlichen Report, der aus einem abstrakten Cliché generiert wird, ist die gesamte Interaktion zwischen Analytiker und RA formal-sprachlich. Die Autoren [Rich et al. 87:80, Reubenstein & Waters 91:227] erachten das als Vorteil, da sie so die Probleme, die bei der natürlich-sprachlichen Analyse entstehen würden, umgehen können. Sie anerkennen jedoch, dass nur natürliche Sprache als Interaktionssprache in Frage kommen kann, wenn man den RA in die Hände von Endbenutzern geben möchte.

> The main benefit of excluding direct interaction with the end-user is that it avoids having to deal with the syntactic complexity of natural language input. Free-form natural language input would be essential for interaction with a naive end-user [Reubenstein & Waters 91:227].

SXL: An Executable Specification Language [Lee & Sluizer 87]

Die beiden Autoren vertreten in ihrem Beitrag die Auffassung, dass der Modellbildung bei der Softwareentwicklung die gleiche Aufmerksamkeit zukommen muss, wie das in den klassischen Ingenieurswissenschaften der Fall ist. Sie präsentieren mit SXL eine logikbasierte Spezifikationssprache, die entwickelt wurde, um ausführbare Modelle (= Prototypen) zu bauen, an denen das funktionale Systemverhalten überprüft werden kann. Mit der Sprache SXL können sowohl endliche als auch nichtendliche Automaten spezifiziert werden. SXL ist in Prolog implementiert; kennt aber selber kein Backtracking, sondern verwendet für die Ausführung vorwärtsverkettende Regeln.

Ein SXL Modell besteht aus einer Menge von Objekten mit Attributen und Relationen zwischen den Objekten. Transaktionsregeln beschreiben das Verhalten während der

Ausführung des Modells. Eine Reihe von Restriktionen sorgen dafür, dass bestimmte Attribute der Objekte und Relationen zwischen den Objekten bei der Ausführung in allen Zuständen gelten. Für das Bibliotheksproblem sind bei der Konzeptualisierung drei Arten von Objekten bestimmt worden: Bibliotheksbenutzer (*normal*, *staff*), Katalogeinträge und Buchkopien. Diese Objekte werden Entitätsklassen (***entity***) zugeordnet. Beispielsweise definiert die Entitätsklasse *book* Objekte mit den folgenden drei Attributen (***attribute***):

```
entity book
   attribute author_of : author
             title_of  : title
             status    : {available, out}
```

Die einzelnen Buchkopien werden erst bei der Ausführung der Spezifikation als Fakten zum Modell hinzugefügt und erhalten eine eindeutige Nummer (*), die vom System vergeben wird. Das führt zu folgender Repräsentation für eine Buchkopie:

```
book(*,sedgewick,algorithms,computer_science,available).
```

Die zulässigen Tätigkeiten sind in der Spezifikation durch Transaktionsregeln (***tran***) beschrieben, die mit Hilfe von deklarativen Vor- und Nachbedingungen angegeben werden. Die Vorbedingung (***pre***) sagt, wann eine Transaktion zulässig ist. Die Nachbedingung (***post***) beschreibt, was sich als Resultat einer Transaktion verändert. Die Transaktion *check_out* hat die folgende Form:

```
tran check_out(User, Book, Borrower)
   pre   status(Book, available)
            and is_staff(User)
            and user(Borrower)
   post  status(Book, out)
            and last_borrower(Book, Borrower)
```

User, *Book* und *Borrower* sind die Eingabeparameter der Transaktion. Die Vorbedingungen verlangen erstens, dass das Buch für die Ausleihe verfügbar ist (*status(Book, available)*); zweitens, dass der Systembenutzer ein Mitarbeiter ist (*is_staff(User)*); und drittens, dass der Ausleiher aus der Klasse der Benutzer (*user(Borrower)*) stammt. Die Nachbedingungen halten folgende Sachverhalte fest: Das Buch ist ausgeliehen (*status(Book, out)*), und der Ausleiher des Buches wird als letzter Ausleiher vermerkt (*last_borrower(Book, Borrower)*). Alle Werte, die

nicht explizit erwähnt werden, bleiben unverändert. Zur Transaktion *check_out* gehören die folgenden beiden Restriktionen:

```
relation last_borrower(book, user) 1-1 auto
maintain user(U) => bounds(B, checked_out_to(B, U), 0, 5)
```

Die erste implizite Restriktion (***1-1 auto***) setzt fest, dass für ein Buch genau ein Ausleiher als letzter Ausleiher vermerkt sein darf. Dadurch wird bei der Ausführung garantiert, dass jeder vorgängige Ausleiher eines Buches, der als letzter Ausleiher vermerkt war, gelöscht wird. Die zweite explizite Restriktion (***maintain***) überprüft bei der Ausführung, ob der Ausleiher nicht bereits mehr als 5 Bücher ausgeliehen hat.

Bestimmte Transaktionen erzeugen Seiteneffekte. Das Schlüsselwort ***output***, das in der folgenden Transaktion anstelle einer Nachbedingung steht, zeigt an, dass es sich hier um eine Transaktion handelt, die zu keiner Zustandsänderung führt, sondern den Systembenutzer bloss informiert:

```
tran list_last_borrower(User, Book)
   pre     is_staff(User) and book(Book)
   output  all Borrower where
           last_borrower(Book, Borrower)
```

Vor der Ausführung der SXL Spezifikation muss ein Anfangszustand definiert werden. Durch Initialisierung wird erreicht, dass es mindestens einen Bibliotheksbenutzer *user* vom Typ *normal* und *staff* gibt und ein Katalogeintrag *cc_entry* sowie eine Buchkopie *book* existiert:

```
initial
   new     user(*, mary, normal)             and
   new     user(*, jane, staff)              and
   new     cc_entry(haley & roots, history)  and
   new     book(*, haley, roots, available)
```

Das Interessante an der Sprache SXL ist, dass sie eine sachverhaltsorientierte Modellierung unterstützt, bei der das Vokabular für die Objekte und Relationen (fast) unmittelbar aus der Problembeschreibung übernommen werden kann:

> Our prototypes are very-high-level models whose "vocabulary" is similar to that used in informal, English-language requirements [Lee & Sluizer 87:231].

A Larch Specification of the Library Problem [Wing 87]
Die Autorin der Spezifikation argumentiert, dass erhöhte Präzision in den frühen Phasen der Softwareentwicklung das Verständnis des Softwareentwicklers für die Anforderungen der Kunden klären kann. Sie schlägt vor, das Bibliotheksproblem mit einer formalen Methode (Larch) anzugehen, die prädikative und algebraische Spezifikationstechniken kombiniert.

Larch ist eine formale Spezifikationsmethode, die besonders die Modularisierung der Spezifikation und die Datenabstraktion unterstützt. Eine Larch Spezifikation besteht aus Komponenten, die in zwei unterschiedlichen Sprachen geschrieben sind: zum einen in einer prädikativen Schnittstellensprache und zum anderen in einer algebraischen Spezifikationssprache (Larch Shared Language). Mit der Schnittstellensprache wird das beobachtbare Verhalten zwischen den Schnittstellen von Programmkomponenten beschrieben. Um die Probleme bei der Kommunikation zwischen den Schnittstellen klein zu halten, sind spezielle Larch Schnittstellensprachen für verschiedene Programmiersprachen (z.B. Ada, C++, Smalltalk, CLU) entwickelt worden. Für das Bibliotheksproblem wurde Larch/CLU als Schnittstellensprache benutzt. Die Programmiersprache CLU ist eine objektorientierte Sprache aus der Pascal Familie [Liskov et al. 81]. Mit Larch/CLU werden in der Beispielspezifikation im wesentlichen die Transaktionen durch Prädikate (Vor- und Nachbedingungen) festgelegt. Für die Definition dieser Prädikate stützt sich die Schnittstellensprache auf eine algebraische Hilfsspezifikation ab, die in der Larch Shared Language (LSL) geschrieben ist. Die LSL erlaubt es, die charakteristischen Abstraktionen programmunabhängig in Form von algebraischen Gleichungen anzugeben und diese zusammenzufassen. Diese algebraischen Konstrukte (*Traits*) sind Minitheorien und definieren die Bedeutung der Prädikate in der Larch/CLU Spezifikation [vgl. Wing 87:35]. Der Softwareentwickler ist mit LSL nicht auf eine beschränkte Menge von Bezeichnungen festgelegt, sondern er kann ein spezialisiertes Vokabular aufbauen, das für eine einzige Spezifikation oder eine ganze Klasse von Spezifikationen verwendbar ist.

Der allgemeine Sachverhalt, dass zu einem Buch ein Titel, ein Autor und ein Sachgebiet gehört, wird in LSL durch einen Trait *book* dargestellt. Die Beziehung zwischen einem Buch *B* und den einzelnen Komponenten wird durch den Datentyp ***record of*** modelliert.

```
Book: trait
  B record of [title:T, author:A, subject:S]
```

Diese Information ist Teil von einem komplexeren Trait *User*. Im untenstehenden Trait *User* zeigt das Schlüsselwort ***includes*** an, dass neben dem Trait *Book* ein Trait *Status* und ein Trait *Set* verwendet werden. Dem Bibliotheksbenutzer *U* werden durch den Datentyp ***record of*** ein Name *N*, ein Status *St* und eine Menge von ausgeliehenen Büchern *BS* zugeordnet. Der Name des Benutzer wird an dieser Stelle nicht explizit angegeben, sondern es wird vorausgesetzt (***assumes***), dass der Name *N* den Eigenschaften einer Äquivalenzrelation genügt. Das Schlüsselwort ***introduces*** führt zwei Operatoren ein, die durch ihren Namen (z.B. *notStaff*) und ihre Signatur (Sorte *U* und Bereich *Bool*) definiert sind. Die Gleichungen, die sich an die ***constraints*** Klausel anschliessen, definieren die Bedeutung der Prädikate *notStaff* und *responsible*.

```
User: trait
  assumes Equality with [N for T]
  includes Status, Book, Set with [BS for S, B for E]
  U record of [name:N, status:St, books:BS]
  introduces
      notStaff: U → Bool
      responsible: B, U → Bool
  constraints U so that for all [n:N, bs:BS, b:B, u:U]
      notStaff(<n, regular, bs>) = true
      notStaff(<n, staff, bs>) = false
      responsible(b, u) = b ∈ u.books
```

Dieser Trait ist wiederum Bestandteil eines komplexeren Trait *Library*, der die Semantik für die Prädikate in der Larch/CLU Spezifikation liefert [vgl. Wing 87:37].

Die Schnittstellensprache Larch/CLU verwendet die in der Hilfsspezifikation definierten Prädikate. Die untenstehende *check_out* Prozedur (***proc***) beschreibt, dass ein Benutzer *u* ein Buch *b* aus der Bibliothek *l* im Normalfall (***ensures normally***) ausleihen kann; ausser (***except***), wenn das Buch nicht verfügbar ist (*notAvail*), die Buchlimite überschritten ist (*overLimit*) oder der Benutzer bereits eine Kopie des Buches ausgeliehen hat (*hasCopy*). Mit dem Buchausleih wird der Benutzer für das Buch verantwortlich (*responsible(b, u_{post})*). Zusätzlich wird vermerkt, dass das Buch ausgeliehen ist (*checkedOut($l_{post'}$, b)*). Das Schlüsselwort ***modifies*** setzt fest, dass während der Transaktion nur die Objekte *l* und *u* verändert werden dürfen.


```
check_out = proc(l:library, u:user, b:book)
            signals(notAvail, overLimit, hasCopy)
   modifies  [l, u]
   ensures normally responsible(b, u_post) ∧
                    checkedOut(l_post', b) except
            signals notAvail  when b ∉ allBooks(l) ∨ checkedOut(l,b)
            signals overLimit when size(u.books) = limit(l)
            signals hasCopy   when responsible(b, u)
```

Interessant ist, dass die Schnittstellensprache Larch/CLU einen Mechanismus zur Verfügung stellt, der signalisiert (***signals***), aus welchem Grund (***when***) eine Transaktion nicht ausführbar ist. Dadurch wird eine explizite Fehlerbehandlung für verschiedene Situationen möglich.

[Wing 87:38] behauptet, dass sich aufgrund der Hilfs- und Schnittstellenspezifikation die Erfüllbarkeit der Anforderungen in der Problembeschreibung nachweisen lasse. Für die Constraints gilt, dass sie vollständig aus der Hilfsspezifikation ableitbar sind.

2.5.3 Nachprüfung

In allen drei Arbeiten wird die informelle Problembeschreibung in eine formale SAS überführt. Die korrekte Ableitung von formalen Spezifikationen aus informellen Anforderungen ist äusserst schwierig. Während dem Spezifikationsprozess werden vom Softwareentwickler eine ganze Reihe von konzeptionellen Entscheidungen getroffen, um die informelle Problembeschreibung zu präzisieren. Genaugenommen wird die Problembeschreibung vom Softwareentwickler interpretiert [vgl. Wing 88:71]. Diese Interpretationen können verschiedenartig ausfallen und sind vom Anwendungsspezialisten in der formalen SAS nur schwer zu validieren.

Hierzu ein Beispiel, das die 4. und die 5. Transaktion der Problembeschreibung betrifft. Das Beispiel soll zeigen, wie sich die Interpretationen von einzelnen Ausdrücken gegenseitig beeinflussen und zu welchen unterschiedlichen Lösungen das führen kann.

Ausschnitt aus dem Library Problem

> 4. Find out the list of books currently checked out by a particular borrower.
> 5. Find out what borrower last checked out a particular copy of a book.
>
> .

[Lee & Sluizer 87] und [Wing 87] interpretieren den verbalen Ausdruck *last checked out* aus der 5. Transaktion unterschiedlich. (Nebenbei: [Rich et al. 87] sprechen dieses Problem nicht an). [Lee & Sluizer 87] unterscheiden bei ihrer Interpretation den Ausdruck *currently checked out* aus der 4. Transaktion vom Ausdruck *last checked out*. Die 5. Transaktion der SXL Spezifikation gibt entweder den gegenwärtigen Ausleiher zurück, wenn eine Buchkopie ausgeliehen ist, oder den letzten Ausleiher, wenn die Buchkopie nicht ausgeliehen ist. Bei dieser Interpretation ist die Menge der gegenwärtig ausgeliehenen Buchkopien eine Teilmenge der zuletzt ausgeliehenen Buchkopien. Im Gegensatz dazu ist für [Wing 87] der Ausdruck *currently checked out* gleichbedeutend mit dem Ausdruck *last checked out*, obwohl die beiden temporalen Adverbien *currently* und *last* in der Problembeschreibung eine Ungleichheit anzuzeigen scheinen. Diese zweite Interpretation hat zur Konsequenz, dass die 5. Transaktion in der Larch Spezifikation den gegenwärtigen Ausleiher einer Buchkopie als Resultat zurückgibt. Ausser diesen beiden Interpretationen ist noch eine dritte Interpretation denkbar. Dabei wird davon ausgegangen, dass die beiden Sätze über zwei disjunkte Mengen von Buchkopien Aussagen machen. Bei dieser Interpretation gibt die 5. Transaktion den letzten Ausleiher nur dann als Resultat zurück, wenn die Buchkopie (gegenwärtig) nicht ausgeliehen ist [vgl. Rueher 87:131].

Wir überprüfen nun zum einen, wie solche unterschiedlichen Interpretationen für den Anwendungsspezialisten in den drei Beispielspezifikationen sichtbar werden, und zum anderen, wie gut die Spezifikationen im allgemeinen den Qualitätsattributen (korrekt, eindeutig, vollständig, verifizierbar, konsistent und verständlich) genügen. Leider erlauben es die veröffentlichten Darstellungen nicht, für jedes einzelne qualitative Attribut eine abschliessende Beurteilung zu machen.

[Rich et al. 87] betonen in ihrer Arbeit, dass sie mit dem Requirements Apprentice (RA) ein Werkzeug anbieten, das die konzeptionelle Lücke zwischen einer informellen Problembeschreibung und einer formalen SAS schliessen soll. Dieser Anspruch wird aber nur zum Teil erfüllt, da die Anwendungsspezialisten vom eigentlichen Spezifikationsprozess weitgehend ausgeschlossen sind. Während des Spezifikationsprozesses greift der RA auf grosse Mengen von Weltwissen zurück, das in einer Cliché-Bibliothek verfügbar ist. Unklar ist, von wem dieses formalisierte Wissen stammt und inwieweit dieses Wissen den Anwendungsbereich überhaupt adäquat widerspiegelt. Der RA überprüft die Konsistenz und die Vollständigkeit der entstehenden SAS aufgrund seines Hintergrundwissens und verlangt bei Bedarf Entscheidungshilfen vom Analytiker. Es ist schwierig, die Korrektheit der SAS zu überprüfen, da allein die Anwendungsspezialisten entscheiden können, ob die intendierten Sachverhalte in der formalen Sprache korrekt beschrieben sind. Das Überprüfen der Korrekheit erfordert eine gute Lesbarkeit und Verständlichkeit der SAS für alle Beteiligten. Um diesem Anspruch gerecht zu werden, erzeugt der RA für die Validierung der SAS eine Zusammenfassung in voller natürlicher Sprache, so dass die Anwendungsspezialisten die Qualität der SAS durch manuelle Techniken überprüfen können. Es gibt jedoch keine Garantie dafür, dass die natürlich-sprachliche Zusammenfassung nicht wiederum mehrdeutig ist und der Intention der Anwendungsspezialisten und Softwareentwickler entspricht.

[Lee & Sluizer 87] verfolgen in ihrer Arbeit einen sachverhaltsorientierten Modellierungsansatz, in dem die Namen der einzelnen Objekte und Relationen für die formale Sprache direkt aus dem Anwendungsbereich übernommen werden. Die entstehende SXL Spezifikation ist für den Softwareentwickler gut lesbar und verständlich. Dagegen dürfte die formale Sprache für die Anwendungsspezialisten zu anspruchsvoll sein, wenn sie die SAS validieren müssen, da sich bestimmte Relationen auf Operationen abstützen, die Kenntnisse der zugrundeliegenden Maschinerie verlangen. Die SXL Spezifikation ist ausführbar, so dass das Systemverhalten im Prinzip durchgespielt werden kann. Der Nachteil bei der Sprache SXL ist, dass die Zwischenergebnisse der symbolischen Ausführung formal-sprachlich sind, was die Überprüfbarkeit der Korrektheit, Vollständigkeit und Konsistenz der SAS durch die Anwendungsspezialisten erschwert.

[Wing 87] spezifiziert das Bibliotheksproblem mit einer formalen Spezifikationssprache, die es erlaubt, modulare und wiederverwendbare SASen zu schreiben. Die formale Präzision von Spezifikationssprachen wie Larch ermöglichen erst die automatische Konsistenzprüfung und die Programmverifikation durch Beweisführung. [Wing 87] zeigt, dass die mathematische Präzision einer formalen Spezifikationssprache dem Softwareentwickler hilft, Fehler, Mehrdeutigkeiten und Fälle von Überspezifikation in der Problembeschreibung zu entdecken. Das Entdecken von Mängeln und die korrekte Behebung dieser Mängel sind jedoch zwei unterschiedliche Dinge. Die Behebung der Mängel verlangt Feedback von den Anwendungsspezialisten. Larch Spezifikationen sind komplexe mathematische Objekte, die nur von Softwareentwicklern mit entsprechender Ausbildung geschrieben und verstanden werden können. [Wing 87] verwendet zwar für die Visualisierung des Bibliotheksmodells eine Statechart, die sie - nebenbei bemerkt - wiederum in natürlicher Sprache erklärt. Leider ist die Beziehung zwischen der Statechart und der Larch Spezifikation nur informell, so

dass es für die Anwendungsspezialisten keine Möglichkeit gibt, nachzuprüfen, ob die Softwareentwickler die in der Problembeschreibung entdeckten Mängel korrekt behoben haben.

2.5.4 Anforderungen an eine Spezifikationssprache

Die drei Spezifikationsbeispiele zeigen, dass die Verwendung von formalen Sprachen dazu führt, dass es für die Benutzer und Anwendungsspezialisten äussert schwierig wird, eine SAS gegenüber den (faktischen) Anforderungen zu validieren. Auf der einen Seite möchte man informelle Spezifikationssprachen verwenden, um

- alle Beteiligten in den Spezifikationsprozess einzubeziehen
- die Korrektheit der SAS intersubjektiv überprüfbar zu machen
- die Vollständigkeit der SAS sicherzustellen
- die Verständlichkeit der SAS zu garantieren

und auf der anderen Seite werden formale Sprachen benötigt, um

- eine eindeutige SAS zu schreiben
- allenfalls von einer ausführbaren SAS zu profitieren
- eine SAS automatisch auf Konsistenz zu überprüfen
- die anschliessende Verifikation zu unterstützen.

In diesem Spannungsfeld lassen sich die Anforderungen an eine optimale Spezifikationssprache formulieren. Eine optimale Spezifikationssprache sollte es den Anwendungsspezialisten erlauben, die in einem Anwendungsbereich wahrgenommenen Phänomene direkt und eindeutig in anwendungsspezifischen Begriffen zu beschreiben, ohne dass dabei eine formal-sprachliche Kodierung notwendig wird [vgl. Balzer et al. 78, Börger & Rosenzweig 95]. Die verwendete Sprache sollte einerseits so vertraut wirken, wie man das von der natürlichen Sprache her kennt, und andererseits dieselbe Genauigkeit und Präzision einfordern, die eine formale Sprache aufweist. Ausserdem muss die Sprache für Spezifikationszwecke genügend ausdrucksstark sein, effizient maschinell verarbeitbar sein und leicht von einem Menschen erlernt werden können.

3. Informalität und Formalität im Einklang

Die Praxis zeigt, dass die Benutzer und Anwendungsspezialisten die faktischen Anforderungen an ein zu entwickelndes Softwareprodukt vorzugsweise informell beschreiben. Sie verwenden dazu die natürliche Sprache als Spezifikationssprache mit der aus dem Anwendungsbereich vertrauten fachsprachlichen Terminologie. Das führt zu zwei Nachteilen: Erstens ist die natürliche Sprache mehrdeutig und vage und zweitens ist die Fachterminologie für die Softwareentwickler oft nur schwer zugänglich. Das hat zur Konsequenz, dass die Konzepte und die akzeptierten Verfahrensweisen aus dem Anwendungsbereich für die Softwareentwickler nicht immer klar erkennbar sind.

Die Softwareentwickler haben die schwierige Aufgabe die informelle Problembeschreibung in eine formale SAS zu überführen. Bei diesem Übergang werden die Ausdrücke der natürlichen Sprache in eine formale (logikbasierte) Sprache übersetzt und modelltheoretisch interpretiert oder allenfalls beweistheoretisch hergeleitet [vgl. Kowalski 94]. Das Resultat ist eine eindeutige formale SAS, die die Systemfunktionalität in erster Linie aufgrund der intendierten Interpretation der Softwareentwickler festlegt und nicht aufgrund der Vorstellungen der Anwendungsspezialisten.

SASen sind Schnittstellen zwischen den faktischen Anforderungen aus dem Anwendungsbereich und dem prozeduralen Verhalten eines Computers. Wenn an dieser Schnittstelle bei der Formalisierung Fehler passieren oder bereits vorschnell Entwurfsentscheidungen getroffen werden und die Anwendungsspezialisten keine Möglichkeit haben, das Dokument zu überprüfen und die Sachverhalte richtig zu stellen, weil sie mit der formalen Spezifikationsmethode nicht vertraut sind, dann wird der Computer die geforderte Funktionalität im Anwendungsbereich nicht erbringen können.

Unter welchen Rahmenbedingungen kann man eine formale Spezifikationssprache den Anwendungsspezialisten zugänglich machen? Auf der einen Seite möchte man von den Eigenschaften einer formalen Spezifikationssprache profitieren, um die Validierung und die Verifikation maschinell unterstützen zu können. Auf der anderen Seite muss man aber feststellen, dass formale Spezifikationssprachen für Leute aus dem Anwendungsgebiet, die nicht über die nötigen mathematische Grundlagen verfügen, kaum zugänglich sind. Wir schlagen deshalb vor, dass man eine anwendungsspezifische Spezifikationssprache einführt, mit der die faktischen Anforderungen auf eine natürliche Weise eindeutig dargestellt werden können. Die Sprache soll informell erscheinen, aber die gleichen formalen und operationalen Eigenschaften wie eine logikbasierte Spezifikationssprache haben [vgl. Fuchs & Robertson 96].

3.1 Textuelle Sichten auf formale Spezifikationen

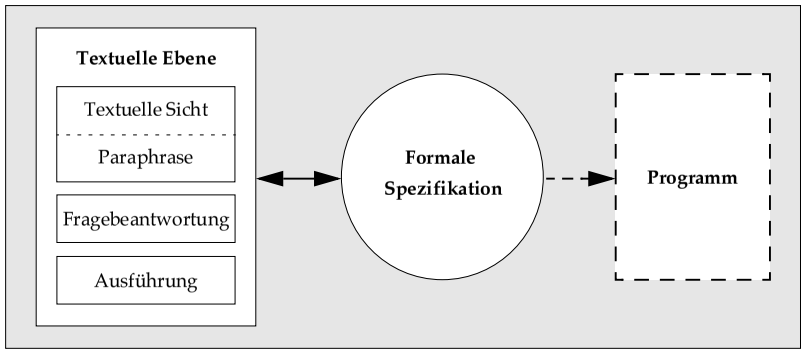
Um die informellen und die formalen Aspekte beim Schreiben einer SAS in deklarativen Einklang zu bringen, führen wir textuelle Sichten (Views) auf formale Spezifikationen in Logik ein. Textuelle Sichten erscheinen informell und werden direkt auf dem Computer in einer anwendungsspezifischen Benutzersprache B erstellt. Zwischen einer textuellen Sicht V und der entsprechenden formalen Spezifikation $Spez$ besteht eine Korrespondenz. Die Korrespondenz kommt durch eine Abbildung der textuellen Sicht V auf die formale Spezifikation $Spez$ mit Hilfe einer eindeutigen Übersetzungsfunktion $f: V \rightarrow Spez$ zustande. Diese Übersetzungsfunktion weist der Sicht V die formale Semantik $[[\alpha]]^M$ der aus der logischen Sprache L konstruierten Spezifikation $Spez$ zu, und zwar relativ zu einem vorgegebenen Modell M. Die inverse Abbildung der formalen Spezifikation $Spez$ auf die textuelle Sicht V entsteht durch die Umkehrfunktion f^{-1}, die die formale Spezifikation $Spez$ in der anwendungsspezifischen Benutzersprache B paraphrasiert.

Die Autoren der SAS schreiben ihre faktischen Anforderungen direkt auf der Sichtebene mit Hilfe eines Editors, der den Schreibprozess unterstützt. Sie verwenden dazu eine wohldefinierte Teilmenge der natürlichen Sprache als Benutzersprache, die aus einem anwendungsspezifischen Vokabular, einer eingeschränkten Grammatik und einer Menge von Schreibprinzipien besteht. Die Autoren entscheiden, welche Ausdrücke der eingeschränkten Sprache die Sachverhalte hinreichend exakt beschreiben und bauen das inhaltsbezogene Vokabular für die SAS selber auf. Durch diese Vorgehensweise kann die Vagheit von Ausdrücken mit Bezug auf die Situation und den Kontext abgefedert werden. Die Schreibprinzipien haben die Funktion von Kontrollstrukturen, denn sie reduzieren die Menge der lexikalischen, syntaktischen, semantischen und pragmatischen Ambiguitäten der vollen natürlichen Sprache bereits vor der maschinellen Analyse, so dass die auf der Sichtebene erstellte SAS eindeutig in eine (ausführbare) logische Spezifikation übersetzt werden kann. Die Autoren brauchen die zugrundeliegende formale Sprache nicht zu kennen, sondern sie müssen bloss mit den Schreibprinzipien vertraut sein. Die automatische Übersetzung macht die scheinbar informelle textuelle Sicht formal, ohne dass die interne formale Repräsentation für die Autoren sichtbar wird. Die Interpretation der textuellen Sicht wird durch eine Paraphrase auf der textuellen Ebene verdeutlicht. Zwei textuelle Sichten sind Paraphrasen voneinander, wenn sie in den verwendeten Ausdrücken oder den syntaktischen Strukturen voneinander abweichen, aber dieselbe modelltheoretische Interpretationen haben. Paraphrasierung wird hier als ein syntaktisches Verfahren verstanden, bei dem das Ergebnis durch kontrollierte Ersetzung und Einsetzung von

Wörtern und Konstituenten zustande kommt. Die Paraphrase gehört zu der gleichen Teilmenge der natürlichen Sprache, die im Prinzip dem Autor auch bei der Textproduktion zur Verfügung steht. Die Paraphrase soll anaphorische Bezüge, elliptische Konstruktionen und Lesarten - die aus den Schreibprinzipien folgen - durch syntaktische Substitutionen und Klammerungen erläutern.

Die Validierung der übersetzten SAS muss anschliessend vollständig auf der textuellen Ebene in der Benutzersprache durchführbar sein. Die Autoren sollen Fragen an die formale Spezifikation stellen können und dazu Konstruktionen der eingeschränkten natürlichen Sprache als Abfragesprache verwenden. Die Fragebeantwortung soll durch logische Inferenz zustande kommen, wobei die Systemantworten wiederum in der Benutzersprache auf der textuellen Ebene verfügbar sein müssen. Neben der Befragung als Validierungstechnik soll die formale Spezifikation symbolisch ausführbar sein, so dass das funktionale Verhalten der Spezifikation auf der textuellen Ebene beobachtet und die Zwischenergebnisse in den vertrauten Begriffen aus dem Anwendungsbereich überprüft werden können.

Textuelle Sicht auf eine formale Spezifikation



Durch die hier beschriebene Vorgehensweise wird erreicht, dass einerseits die Lesbarkeit der SAS für alle Beteiligten garantiert ist und dass die maschinelle Verarbeitbarkeit der SAS gewährleistet ist. Für die Benutzer und Anwendungsspezialisten wird es möglich, die SAS in der Benutzersprache zu validieren. Um die formale SAS ausführbar zu machen, muss sie vom Anwendungsspezialisten oder auch vom Softwareentwickler um situations- und simulationsabhängige Informationen erweitert werden [vgl. Schwitter & Fuchs 96]. Die Softwareentwickler können von den Vorteilen einer

formalen SAS für den weiteren Softwareentwicklungsprozess profitieren. Der Unterschied zwischen einer ausführbaren SAS und einem Anwendungsprogramm liegt in der Effizienz. Da die ausführbare Spezifikation möglicherweise noch nicht genügend effizient ist, muss sie durch Transformationstechniken und anschliessende Kompilation in ein effizientes Programm überführt werden [vgl. Kowalski 85, Vasconcelos & Fuchs 95]. Der Übergang von einer formalen Spezifikation in Logik in ein effizientes Programm wird in dieser Arbeit nicht weiter untersucht. Das Hauptinteresse richtet sich hier auf den Entwurf einer eingeschränkten natürlichen Sprache, die auf der textuellen Ebene die Formalität vor den Anwendungsspezialisten verbirgt, um sie dann durch Übersetzung für die Softwareentwickler wieder zugänglich zu machen.

3.2 Skizze einer eingeschränkten natürlichen Sprache

Wie könnte eine eingeschränkte natürliche Sprache aussehen, die für alle Beteiligten leicht lesbar ist, zudem genügend mächtig ist, um faktische Anforderungen zu beschreiben, und eindeutig in eine logische Sprache übersetzt werden kann? Verlassen wir hier kurz das Bibliotheksbeispiel und schauen uns noch einen weiteren Spezifikationsausschnitt aus einem anderen Anwendungsbereich an. Der untenstehende Textausschnitt *Access* stammt aus einer SAS, die die Funktionalität eines Geldautomaten in voller natürlicher Spache beschreibt. Der Ausschnitt beschreibt die Zugangsbedingungen zum Geldautomaten (*SimpleMat*) aus der Perspektive eines Systembenutzers.

> *Access*
> The customer introduces his card which is read by the SimpleMat. The card number is checked for validity and whether it is currently active. If the card cannot be read, has an invalid card number, or is not currently active it is rejected with the message 'Invalid Card'. Otherwise the access door of the SimpleMat is opened and the message 'Enter Your Personal Code' is displayed. The customer enters the personal code. The code is checked for validity by a trap door algorithm which should result in the check code [being] stored on the card. If the personal code entered is not correct the SimpleMat displays the message 'Incorrect Code. Transaction Terminated. Card Retained.' and the door of the SimpleMat is closed. The SimpleMat retains the card [Fuchs 89].

Der Textausschnitt enthält einfache und zusammengesetzte Sätze. Die zusammengesetzten Sätze bestehen aus einfachen Teilsätzen, die durch Konnektoren entweder fest miteinander verfugt oder lose verbunden sind. Im ersten Fall handelt es sich um untergeordnete durch Subordinatoren (*which*, *if*) eingeleitete Nebensätze, insbesondere um Relativsätze wie in (40) und Konditionalsätze wie in (41), die zusammen mit einem Hauptsatz ein Satzgefüge bilden.

(40) *The customer introduces his card which is read by the SimpleMat.*

(41) *If the personal code entered is not correct the SimpleMat displays the message 'Incorrect Code. Transaction Terminated. Card Retained.'.*

Im zweiten Fall stehen Hauptsätze (42) gleichrangig - durch Koordinatoren (*and*) verbunden - nebeneinander und bilden eine Satzverbindung:

(42) *SimpleMat is opened and the message 'Enter Your Personal Code' is displayed.*

Die Verben in den Teilsätzen stehen in der dritten Person Singular, im Präsens, meistens im Indikativ und in der aktiven oder passiven Handlungsform. Im Gegensatz zur aktiven Handlungsform tritt beim Passiv die Beziehung zwischen dem Subjekt und dem Verb in den Hintergrund, wodurch der Agent, der ein Ereignis verursacht, oft aus dem Kontext erschlossen werden muss.

Im Textausschnitt tauchen anaphorische Verweisformen (z.B. *it, the card*) von unterschiedlicher Komplexität auf, für die der Leser eine korrekte Bezugstelle finden muss. Ellipsen kommen in den einfachen Teilsätzen vor, wenn gleichwertige Satzteile aus einem vorausgehenden Teilsatz ausgespart werden. Ellipsen haben sprachökonomische Ursachen und verlangen von den Lesern eine struktur- und sinngerechte Ergänzung. Beispielsweise kommt im konditionalen Satzgefüge

(43) *If the card cannot be read, has an invalid card number, or is not currently active it is rejected with the message 'Invalid Card'.*

das Subjekt *the card* nur im ersten Teilsatz vor. Die syntaktischen Leerstellen in den beiden nachfolgenden Teilsätzen können aber ohne Schwierigkeiten durch den Leser ergänzt werden. Beim Personalpronomen *it*, das den Hauptsatz in (43) einleitet, handelt es sich um eine anaphorische Verweisform, die dem Leser eine Suchanweisung für einen vorausgehenden syntaktisch passenden sprachlichen Ausdruck (*the card*) gibt. Schwieriger ist die Frage zu beantworten, welcher (implizite) Koordinator den ersten Teilsatz mit dem zweiten Teilsatz verknüpft:

(44) *If the card cannot be read, [and/or the card] has an invalid card number, or [the card] is not currently active [the card] is rejected with the message 'Invalid Card'.*

Handelt es sich hier um eine Konjunktion (*and*) oder um eine Disjunktion (*or*), die ergänzt werden muss? Diese Ellipse kann nicht mehr auf der syntaktischen Ebene aufgelöst werden, sondern die Resolution ist davon abhängig, was der Autor mit der Äusserung intendiert und ob der Leser diese Intention teilt. Das Verknüpfungsmuster

ist in diesem Fall textuell unterspezifiziert.

Der Textausschnitt zeigt, dass man sich sehr genau überlegen muss, welche Konnektoren, welche Gebrauchsweisen von Verben, welche anaphorischen Verweisformen und welche Formen von Ellipsen man in der eingeschränkten natürlichen Sprache zulassen will, so dass die entstehende SAS - ohne zusätzliches Weltwissen - immer noch eindeutig in Logik übersetzt werden kann.

Hier ist eine erste Skizze, wie der Textausschnitt in einer eingeschränkten natürlichen Sprache aussehen könnte [vgl. Fuchs & Schwitter 96]:

```
% Access
The customer introduces his card.
SimpleMat reads the card and checks the card number.

If the card is not readable
or the card number is not valid
or the card number is not active
then SimpleMat rejects the card and displays the message 'Invalid Card'.

If the card is readable
and the card number is valid
and the card number is active
then SimpleMat opens the access door.

SimpleMat displays the message 'Enter Your Personal Code'.

The customer enters the personal code.

SimpleMat checks the correctness of the personal code.
If the trap door algorithm results in the check code
then the personal code is correct.

If the personal code is not correct
then SimpleMat displays the message
'Incorrect Code. Transaction Terminated. Card Retained.'
and SimpleMat closes the door and retains the card.
```

Der Text besteht wie das Original wiederum aus einfachen und zusammengesetzten Sätzen. Im Unterschied zum Original werden die einfachen Sätze "konstruktiv" zu komplexen Sätzen zusammengebaut. Alle Funktionswörter (*and*, *or*, *not*, *if-then*) wer-

den explizit genannt und erscheinen an der Textoberfläche. Gewisse anaphorische Verweisformen mit eindeutiger Reichweite und syntaktische Ellipsen sind zulässig, so dass die Sprache natürlich und vertraut wirkt. Diese Einschränkungen der Sprache erlauben es, Mehrdeutigkeiten auszuschliessen und die maschinelle Verarbeitungszeit in vernünftigen Grenzen zu halten.

Die Gebrauchsweise der eingeschränkten Sprache muss für die Autoren leicht erlernbar und erinnerbar sein. Dies ist ein wichtiger Punkt, denn die Autoren sollen die SAS interaktiv auf dem Computer schreiben können. Die Frage stellt sich dann, wie man die eingeschränkte natürliche Sprache den Anwendungsspezialisten und Autoren vermittelt, so dass der Lernaufwand vertretbar bleibt. Ein interessanter Ansatz besteht darin, dass man die Sprache, die das System verarbeiten kann, "transparent" macht. Wenn ein Spezifikationssystem Rückmeldungen gibt und dabei die gleiche Teilmenge der natürlichen Sprache verwendet, die verarbeitet werden kann, dann lernen die Autoren leicht von den Erfahrungen, die sie mit dem System machen, und können die Systemantworten als Modell für ihre eigenen Formulierungen nehmen [vgl. Slator et al. 86, Capindale & Crawford 89, Zoltan-Ford 91].

Der Schreibprozess auf dem Computer muss durch Korrekturwerkzeuge unterstützt werden, denn die technischen Autoren machen Fehler beim Schreiben der Spezifikation. Es müssen zwei Arten von Fehlern unterschieden werden: Kompetenzfehler und Performanzfehler [vgl. Véronis 91]. Kompetenzfehler entstehen aufgrund mangelnder Kenntnisse der linguistischen Regeln (z.B. der zulässigen grammatischen Schreibprinzipien). Performanzfehler haben ihre Ursache in technischen Missgeschicken (z.B. Tippfehler). Die Fehlerkorrektur im Fall von Kompetenzfehlern erfordert besondere Sorgfalt, weil aufgrund dieser Fehler die Interaktion zwischen den Autoren und der Maschine zusammenbrechen kann. Im Gegensatz zu Kompetenzfehlern können Performanzfehler von den Autoren selber behoben werden. Wenn dafür eine automatische Korrektur zu aufwendig ist, reicht bereits eine einfache Fehlermeldung, die den Autor auf den Fehler hinweist. Vollautomatische Fehlerkorrektur ist eine schwierige Aufgabe, da ein Fehler aus unterschiedlichen Quellen stammen kann. Eine gute Strategie besteht darin, dass das System dem Autor einen Korrekturvorschlag unterbreitet und ihn dann selber entscheiden lässt, ob er den Vorschlag akzeptieren oder ablehnen will. Um die Autoren nicht zu entmutigen, darf das System diese nie über die Grenzen der maschinellen Verarbeitbarkeit der Sprache im Unklaren lassen.

4. Kontrollierte natürliche Sprachen

Nach der Einführung von textuellen Sichten auf formale Spezifikationen und einer ersten kurzen Skizze einer eingeschränkten natürlichen Sprache für die Anforderungsspezifikation, soll nun eingehender untersucht werden, wie eine Teilmenge der natürlichen Sprache beschaffen sein muss, damit sie als Spezifikationssprache eingesetzt werden kann. Zu diesem Zweck wird in einem ersten Schritt zwischen Subsprachen, die organisch gewachsen sind, und kontrollierten natürlichen Sprachen, die künstlich geschaffen worden sind, unterschieden. In einem zweiten Schritt wird untersucht, wie kontrollierte natürliche Sprachen für die Dokumentation und Übersetzung in der Industrie eingesetzt werden. In einem dritten Schritt werden zum einen eine Reihe von Ansätzen besprochen, die Subsprachen für die Anforderungsspezifikation nutzen, und zum anderen wird ein Ansatz vorgestellt, der eine kontrollierte natürliche Sprache für die Anforderungsspezifikation einsetzt. Die verschiedenen Ansätze werden auf Schwachstellen abgeklopft und bilden dann zusammen mit den Vorschlägen für kontrollierte natürliche Sprachen aus der Industrie den Ausgangspunkt für die Entwicklung einer eigenen kontrollierten natürlichen Sprache, die für die Spezifikation von funktionalen Anforderungen ausgelegt ist.

4.1 Subsprachen und kontrollierte natürliche Sprachen

Subsprachen entstehen auf natürliche Weise durch den Gebrauch der vollen natürlichen Sprache in Gemeinschaften, in denen Fachleute über spezialisiertes Wissen in einem begrenzten Diskursbereich kommunizieren. Im Gegensatz zu Subsprachen sind kontrollierte natürliche Sprachen künstlich geschaffen, um komplexe (technische) Informationen zwischen Fachleuten und Laien auf eine leicht verständliche Art zu vermitteln.

Der Begriff der Subsprache wurde von [Harris 68:152 ff.] in Anlehnung an das Konzept von Subsystemen in der Mathematik in die Linguistik eingeführt. In der Mathematik ist ein Subsystem relativ einfach durch die Beschränkung der Elemente und der zulässigen Operationen eines komplexeren Systems definierbar. Für die Definition einer Subsprache der natürlichen Sprache ist diese Vorgehensweise nicht gleichermassen praktikabel, da die Beschreibung der Subsprache möglicherweise Grammatikregeln (≈ Operationen) verlangt, die für eine natürliche Standardsprache (= normierte überregionale Verkehrssprache) nicht gelten.

Eine Subsprache ist eine echte Teilmenge einer natürlichen Sprache, aber nicht notwendigerweise eine Teilmenge der Standardsprache einer Sprache. Eine Subsprache wird aus einer speziellen Menge von Grammatikregeln erzeugt, wobei ein Teil der Grammatikregeln zum Inventar der Standardsprache gehört und es möglicherweise einen anderen Teil gibt, der ausschliesslich der Subsprache angehört.

> A sublanguage is a proper subset of the sentences of a language, closed under certain grammatical operations of the whole language. That is, the result of these operations (e.g. transformations or conjunctions) operating on a sentence of the sublanguage, or on a pair of them, is again a sentence of the sublanguage. The sublanguage is characterized by particular word classes and sentence classes (word-class sequences) which are not necessarily classes of the language, and possibly by grammatical operations that are not distinguished as such in the language [Harris et al. 89:2].

Diese Definition ist weitgehend syntaktisch. Sie macht nur indirekt eine inhaltsbezogene Aussage über den Diskursbereich einer Subsprache, insofern als die Art und Weise, wie Wörter und Sätze klassifiziert werden, die Organisation des Diskursbereichs widerspiegeln kann. In der Literatur sind Subsprachen hauptsächlich anhand von technischen und wissenschaftlichen Texten untersucht worden [Grishman & Kittredge 86, Bonzi 90, Sager 90, Losee & Haas 95, Haas 97]. Dabei taucht immer wieder eine Konstante auf: Subsprachen haben ein spezialisiertes Vokabular, das sich auf einen beschränkten Diskursbereich (z.B. Wetterprognosen, organische Chemie, Unterhaltshandbücher, Gesetzestexte) bezieht. Aus diesem Vokabular wird meistens nur eine Teilmenge der Wortformen einer natürlichen Sprache aktiviert. Zudem ist die Bildung von neuen Wörtern durch produktive Wortbildungsregeln oft subsprachenspezifisch (z.B. charakeristische Affigierung und Akronymbildung) [Kittredge 82:80]. Von aussen betrachtet unterscheiden sich Subsprachen nicht nur lexikalisch, syntaktisch oder semantisch von der Standardsprache, sondern sie weisen auch eigene markante Diskursstrukturen auf. Von innen betrachtet, sind Subsprachen konsistent, wenn auch unterschiedlich komplex: Sie haben ihre eigene Grammatik und können theoretisch als unabhängige natürlich-sprachliche Systeme aufgefasst werden. Allgemein verbindliche Eigenschaften von Subsprachen lassen sich nur schwer angeben:

> There are, however, certain tendencies such as the use of telegraphic style, the use of imperatives, the absence of or restriction to questions, the absence of certain tenses, absence of or frequent use of passives, and the use of certain types of pronominalization or coreference [Lehrberger 86:30].

Ein gutes Beispiel für eine einfach gebaute Subsprache, deren Grammatik aber erheblich vom natürlich-sprachlichen Standard abweicht, sind Wetterbulletins. Wetterbulle-

tins sind stark formatiert, verwenden ein relativ kleines Vokabular und eine einfache telegraphische Sprache. Sätze in Wetterbulletins haben ein verarmtes Tempussystem, wenige Artikel und brauchen nur bestimmte pronominale Verweisformen [Kittredge 87:61, Arnold et al. 95]. Folgender Beispieltext illustriert diese Eigenschaften:

> FORECASTS FOR YUKON AND NORTHWESTERN BC
> ISSUED BY ENVIRONMENT CANADA AT 5:30 AM PDT
> FRIDAY JULY 11 1980 FOR TODAY AND SATURDAY
> KLONDIKE
> BEAVER CREEK
> STEWART RIVER
> RAIN OCCASIONALLY MIXED WITH SLEET TODAY CHANGING TO SNOW THIS EVENING. HIGHS 2 TO 4. WINDS INCREASING TO STRONG NORTHWESTERLY THIS AFTERNOON. CLOUDY WITH A FEW SHOWERS SATURDAY. HIGHS NEAR 6 [Kittredge 87:61].

Erfahrungen mit TAUM-METEO, einem spezialisierten Übersetzungssystem [Chevalier et al. 78], das Wetterbulletins aus der englischen Sprache ins Französische übersetzt, haben gezeigt, dass der Komplexitätsgrad einer Subsprache der entscheidende Faktor für die erfolgreiche maschinelle Übersetzung bildet [vgl. Tucker 87:30 ff.]. Da Subsprachen natürlich gewachsene Gebilde sind, eignen sie sich nicht gleichermassen gut für die maschinelle Verarbeitung. Gewisse Subsprachen sind mit der gegenwärtig verfügbaren computerlinguistischen Technologie nicht vollautomatisch verarbeitbar [van der Eijk et al. 96:66]. Besonders problematisch sind stark elliptische Texte, bei denen eine Tilgung des sprachlichen Materials in extremer Form vorliegt. Ein Beispiel aus der Softwareentwicklung für stark elliptische Texte sind Programmkommentare [vgl. Fuchs 90:214]:

```
% chain_forward(Facts,NewFacts) :-
% derive NewFacts from Facts by forward-chaining

chain_forward(Facts,NewFacts) :-          % find new facts
    derive_facts(Facts,Facts1),           % one forward step
    chain_forward(Facts1,NewFacts).       % recurse
chain_forward(Facts,Facts).               % all facts found
```

Der Leser solcher Kommentartexte muss die syntaktischen Leerstellen sinn- und strukturgerecht ergänzen. Die Rekonstruktion der Texte verlangt Inferenz und Sachwissen; Leistungen, die von Fachleuten gut, aber von einer programmierbaren Maschine nur unter bestimmten Voraussetzungen erbracht werden können. Um die Komplexität

einer Subsprache zu bestimmen und die Möglichkeit für die automatische Verarbeitung abschätzen zu können, muss ein beträchtlicher Aufwand an Vorarbeit in Korpusanalysen investiert werden.

Korpusanalysen zeigen, dass realistische Texte, die in einer Subsprache geschrieben sind, oft sprachliches Material enthalten, das nicht zur eigentlichen Subsprache gehört. Diese Texte bestehen aus einer Mischung von fachspezifischen Diskursbereichen und Metadiskursen über diese Bereiche [Harris et al. 89:26 ff.]. Während Fachleute keine Probleme mit der Unterscheidung von diesen zwei Diskursebenen haben, ist es für die maschinelle Sprachverarbeitung schwierig, die Grenze zwischen dem Diskursbereich der Subsprache und dem Metadiskurs algorithmisch zu bestimmen. Das folgende Beispiel verdeutlicht das Problem an einem Textausschnitt aus dem Gebiet der Geometrie:

> Pythagoras proved the theorem that now bears his name in the sixth century B.C. He was able to show that the area of the square on the hypothenuse of a right triangle equals the sum of the areas of the squares on the other two sides [Lehrberger 86:21].

Der erste Satz stammt nicht aus der Subsprache der Geometrie, sondern macht eine Aussage über die Geschichte der Geometrie. Es handelt sich hier um einen Metadiskurs. Der zweite Satz besteht aus einem Hauptsatz und einem Relativsatz, wobei der Hauptsatz Teil des Metadiskurses ist und der Relativsatz Information über den Diskursbereich der Geometrie liefert. Der Wechsel zwischen den beiden Diskursebenen vollzieht sich im selben Satz und ist in diesem Fall an der syntaktischen Struktur ablesbar.

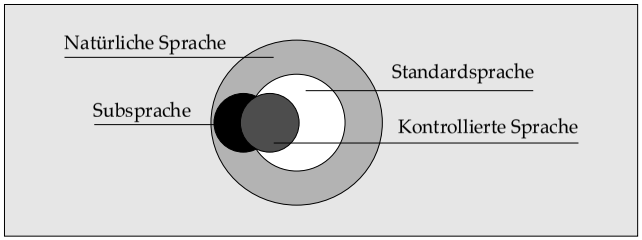
Die Beispiele zeigen, dass dort, wo hochspezialisiertes Wissen in engbegrenzten Diskursbereichen unter Fachleuten kommuniziert wird, sich innerhalb der natürlichen Sprache Subsprachen herausbilden, die den Fachleuten eine ökonomische und differenzierte Ausdrucksweise über bereichsspezifische Sachverhalte ermöglichen. Bei diesen Subsprachen handelt es sich um ein Kontinuum von sprachlichen Varietäten, für die sich - wie bereits betont - allgemeine Eigenschaften nur tendenziell angeben lassen.

Subsprachen sind für Laien opak und kaum zugänglich. Um die Sprachbarriere zwischen Fachleuten und Laien zu überwinden, sind vor allem in der Industrie aus ganz praktischen (wirtschaftlichen) Gründen kontrollierte natürliche Sprachen entwickelt worden. Der zunehmende Umfang und die steigende Komplexität von technischen Dokumenten einerseits und die Globalisierung der Marktwirtschaft andererseits ver-

langen, dass technische Dokumente möglichst eindeutig und leicht verständlich geschrieben sind, so dass sie auch von Personen verstanden werden können, deren Muttersprache nicht die Sprache ist, in der die Dokumente geschrieben sind. Hinzu kommt, dass Dokumente, die einem wohldefinierten Standard gehorchen, besser unterhalten, wiederverwendet und in andere Zielsprachen übersetzt werden können [vgl. Wojcik & Hoard 96]. Ähnlich wie Standardsprachen den interindividuellen Sprachgebrauch regeln, normieren kontrollierte natürliche Sprachen den Sprachgebrauch von spezialisierten Subsprachen.

Eine kontrollierte natürliche Sprache ist eine präzis definierte Teilmenge der natürlichen Sprache. Sie weist gegenüber der Standardsprache eine Reihe von lexikalischen, grammatischen und stilistischen Einschränkungen auf und kann durch eine bereichsspezifische Terminologie und spezielle grammatische Konstruktionen erweitert sein. Die Einschränkungen und die spezifischen Erweiterungen werden durch explizite (präskriptive) Regeln festgehalten, die den Autoren bei der Textproduktion zur Verfügung stehen und die im Idealfall maschinell überprüfbar sind.

Subsprache - Kontrollierte Sprache



Kontrollierte natürliche Sprachen haben zum Ziel, alle Aspekte der Textproduktion und der Textrezeption zu verbessern und die maschinelle Verarbeitung der produzierten Texte zu erleichtern. Das Konzept der kontrollierten natürlichen Sprache setzt voraus, dass die Autoren über das entsprechende Sachwissen aus dem Anwendungsbereich der Subsprache verfügen und das Sprachwissen der kontrollierten Sprache erworben haben. Das Sprachwissen muss in Form von Regeln bzw. Schreibprinzipien vorliegen, die leicht erlernbar und erinnerbar sind, so dass die kontrollierte natürliche Sprache den Schreibprozess nicht behindert.

Während heutzutage viele Firmen kontrollierte natürliche Sprachen einsetzen, um die Lesbarkeit und die Verarbeitbarkeit ihrer Dokumente zu verbessern [vgl. Adriaens

96a], sind nur wenige Versuche unternommen worden, kontrollierte natürliche Sprachen für den Spezifikationsprozess in der Softwareentwicklung nutzbar zu machen [vgl. Macias & Pulman 95, Fuchs & Schwitter 96, Schwitter & Fuchs 96]. Im Vergleich zu bekannten kontrollierten natürlichen Sprachen, die für die technische Dokumentation verwendet werden, erfordern kontrollierte natürliche Sprachen für die Softwareentwicklung spezielle syntaktische und semantische Einschränkungen, da man gewisse Eigenschaften einer Spezifikationen, zum Beispiel ihre Übersetzbarkeit in eine formale Sprache garantieren möchte.

4.2 Kontrollierte natürliche Sprachen für die Dokumentation und für die Übersetzung

Der bekannteste Vorläufer von kontrollierten natürlichen Sprachen ist *BASIC English*. *BASIC* ist ein Akronym und steht für "British American Scientific International Commercial". BASIC English wurde in den 30er Jahren mit dem Ziel entwickelt, die englische Sprache durch systematische Einschränkung des Vokabulars zu vereinfachen und leichter erlernbar zu machen. Zum einen sollte BASIC English als eine internationale Hilfssprache dienen, die weltweit als Verkehrssprache in Handel und Wissenschaft gebraucht werden kann. Zum anderen sollte BASIC English für Nichtmuttersprachler eine angemessene Einführung in die englische Sprache bieten und für Muttersprachler als Leitfaden für den klaren und korrekten Sprachgebrauch dienen [vgl. Ogden 32, Ogden 38]. BASIC English unterscheidet sich von früheren Versuchen, eine internationale Hilfssprache zu konstruieren, dadurch, dass die Sprache in sich komplett ist und - im Unterschied zu vollkommen künstlichen oder hybriden Sprachen - eine echte Teilmenge der englischen Sprache ist [Crystal 87:352 ff.].

Das Vokabular von BASIC English umfasst 850 Wörter und wurde so gewählt, dass die Bedürfnisse des täglichen Sprachgebrauchs, wofür gewöhnlich 25´000 Wörter nötig sind, abgedeckt werden können. Diese 850 Wörter sind unterteilt in 400 allgemeine Nomen (*account, year*), 200 konkrete Nomen (*angle, worm*), 100 allgemeine Eigenschaften (*angry, long, young*), 50 gegensätzliche Eigenschaften (*short, old*) und 100 Operationen (*and, or, do, if, not, that*). Dieses Grundvokabular kann bei Bedarf um bereichsspezifische technische und wissenschaftliche Wörter ergänzt werden. Zudem wurde eine kleine Menge von Regeln vorgeschlagen, um weitere Wörter aus den im Wörterbuch verzeichneten Einträgen abzuleiten (z.B. Negation durch Präfigierung *common -> uncommon*). Da es in BASIC English nur 18 Verben gibt, müssen weitere Prädikationen durch Paraphrasenbildung zum Ausdruck gebracht werden (z.B. *ask* ->

put a question).

Es hatte sich gezeigt, dass BASIC English relativ schnell - in ungefähr 60 Stunden - zu erlernen war. Mehr Schwierigkeiten machte es den Autoren, Texte in BASIC English so zu verfassen, dass die intendierte Bedeutung bewahrt blieb. Die Vereinfachung des Vokabulars führte zu komplexeren syntaktischen Strukturen und zu wachsendem Gebrauch von idiomatischen Wendungen. Die Autoren bekundeten Mühe, den passenden Ausdruck in BASIC English zu finden und waren gezwungen, ungeschickte Umschreibungen zu verwenden [vgl. Crystal 87:356, Arnold et al. 95]. Der Diskursbereich von BASIC English war im Verhältnis zur Grösse des Vokabulars zu umfangreich und das Grundvokabular lieferte nicht die notwendige Präzision. Heute ist BASIC English nur noch von historischem Interesse, doch gingen von dieser Sprache wichtige Impulse für spezialisiertere kontrollierte natürliche Sprachen aus.

Mitte der 60er Jahre nahm die Firma Caterpillar Tractor Company (USA) das Konzept von BASIC English auf und wandte es auf einen engeren Diskursbereich an. Konkret wurden technische Handbücher mit Hilfe der Sprache Caterpillar Fundamental English (CFE) - einer Variante von BASIC English - produziert. Die Absicht war es, die Kosten für die Produktion dieser technischen Dokumente zu senken, indem man die Texte nicht mehr in verschiedene Zielsprachen übersetzte, sondern direkt in CFE abfasste. Dadurch erhoffte man sich, eine relativ gute Lesbarkeit und Verständlichkeit der Texte für diejenigen Ingenieure und Mechaniker, die nur geringe Englischkenntnisse hatten. CFE bestand aus dem Grundvokabular von BASIC English, das um die notwendige technische Terminologie erweitert war. Es stellte sich aber auch hier heraus, dass das Grundvokabular von CFE zu limitiert war, was die Autoren dazu verleitete, komplexe Sachverhalte durch lange und umständliche Sätze zu beschreiben [Mitamura & Nyberg 95:12].

Caterpillar Inc. verwendet CFE heute nicht mehr. An die Stelle von CFE ist der firmeneigene Standard Caterpillar Technical English (CTE) getreten. CTE wird in einem kombinierten Autoren- und Übersetzungssystem erfolgreich für die Produktion von technischen Dokumenten und als Quellsprache für die maschinelle Übersetzung eingesetzt [Hayes et al. 96:84 ff.].

Historisch gesehen ist die Sprache CFE interessant. In der Industrie diente CFE als Modell für die Entwicklung von zwei kontrollierten natürlichen Sprachen: Smart's Plain English Program (PEP) und White's International Language for Serving and Maintenance (ILSAM). PEP führte zu kontrollierten Sprachen, die von Firmen wie Clark, Rockwell International und Hyster (Hyster Easy Language Program - Help) verwendet werden. ILSAM kann als Ausgangspunkt für anwendungsorientierte Weiterentwicklungen von kontrollierten natürlichen Sprachen gesehen werden, die heute von der Association Européenne des Constructeurs de Matériel Aérospatial (AECMA), Alcatel Bell, IBM, Rank Xerox, Perkins Engines und Ericsson Telecommunications benutzt werden [vgl. Adriaens & Schreurs 92:595 ff.].

Zur Zeit sind verschiedene Varianten von kontrollierten natürlichen Sprachen in Entwicklung, die auf die charakteristischen Bedürfnisse der einzelnen Firmen zugeschnitten sind. In Schweden entwickelt die Firma Scania CV AB in Zusammenarbeit mit dem Institut für Linguistik der Universität Uppsala die kontrollierte natürliche Sprache ScaniaSwedish. Diese Sprache dient als Standard für die Produktion von Lkw Handbüchern und als Quellsprache für die maschinelle Herstellung von mehrsprachigen Dokumenten [Almqvist & Sågvall Hein 96:159 ff.]. In Deutschland beschäftigt sich die Firma Siemens AG im Projekt Siemens-Dokumentationsdeutsch mit der Definition von kontrolliertem Deutsch, das den Anforderungen für die Dokumentation von verschiedenen industriellen Anwendungen genügen muss und als Grundlage für die maschinelle Übersetzung der Texte dient [Schachtl 96:143 ff.]. In Frankreich führt die Firma Aérospatiale Korpusstudien durch, um festzustellen, wie eine gut verarbeitbare kontrollierte natürliche Sprache definiert sein muss. Es wird untersucht, ob eine bestehende kontrollierte natürliche Sprache (GIFAS Rationalised French) modifiziert werden kann oder ob die kontrollierte Sprache besser von Grund auf neu zu entwerfen ist [Lux & Dauphin 96:193 ff.].

Das rege Interesse an der guten Verarbeitbarkeit von kontrollierten natürlichen Sprachen kommt daher, weil man den Schreibprozess durch den Computer unterstützt möchte, um dadurch die Grammatik und den Schreibstil konsistent halten zu können. Ausserdem möchte man von einer vollautomatischen Übersetzbarkeit und einer optimalen Wiederverwendbarkeit der textuellen Erzeugnisse profitieren. Es soll an dieser Stelle bloss erwähnt werden, dass auf dem Gebiet der Entwicklung von Prüf- und Korrekturwerkzeugen, die den Schreibprozess von Texten in kontrollierter natürlicher Sprache unterstützen, eine rege Forschungstätigkeit herrscht [vgl. Adriaens 94, Adriaens & Macken 95, Wojcik & Holmback 96, Adriaens 96b].

Ausser der Reduktion von Übersetzungskosten liessen sich durch die Verwendung von kontrollierten natürlichen Sprachen zwei weitere Vorteile nachweisen: Zum einen

kann die Lesbarkeit und die Verständlichkeit von technischen Dokumenten für die Leser aller Sprachgruppen (Muttersprachler und Nichtmuttersprachler) verbessert werden [vgl. Hayes et al. 96:87]. Zum anderen konnten bei der maschinellen Übersetzung bessere Resultate erzielt werden, wenn die Quelltexte in kontrollierter natürlicher Sprache vorlagen [vgl. Mitamura & Nyberg 95:12]. Kontrollierte natürliche Sprachen sind in der Regeln in kurzer Zeit erlernbar, doch kann der Schreibprozess in der Anfangsphase im Vergleich zu konventioneller Sprache bis zu 20% mehr Zeit in Anspruch nehmen [vgl. Farrington 96:21, Goyvaerts 96:139].

4.2.1 KANT Controlled English (KCE)

KANT Controlled English (KCE) wird für das wissensbasierte maschinelle Übersetzungssystem KANT am Zentrum für maschinelle Übersetzung (CMT) der Carnegie Mellon Universität entwickelt [Carbonell et al. 92, Mitamura & Nyberg 95, Nyberg & Mitamura 96]. KANT ist für die Übersetzung von technischen Dokumenten in KCE ausgelegt und basiert auf einem Interlingua-Ansatz [Lonsdale et al. 94]. Die Analysekomponente von KANT erstellt für jeden Satz der Quellsprache einen Interlingua-Ausdruck, der von der Generierungskomponente in einen Satz der Zielsprache überführt wird. Die Interlingua ist unabhängig von der Quell- und Zielsprache und beruht auf einer Menge von Konzepten (Objekte, Ereignisse und Eigenschaften), die aus dem jeweiligen Diskursbereich durch einen vorausgehenden Wissensakquisitionsprozess abgeleitet wird [Mitamura et al. 93]. KANT produziert Übersetzungen von hoher Qualität, ohne dass eine manuelle Nachbearbeitung nötig ist. Die Autoren werden beim Schreiben der KCE Texte durch das Werkzeug *ClearCheck* unterstützt, mit dem sich überprüfen lässt, ob die Sätze mit dem kontrollierten Vokabular und der kontrollierten Grammatik übereinstimmen. Stösst ClearCheck auf eine lexikalische oder strukturelle Mehrdeutigkeit, wird vom Autor eine Entscheidung für eine Lesart verlangt. Die Sprache KCE unterstützt die Verwendung von speziellen SGML (Standard Generalized Markup Language) Tags, um die gewünschte Lesart von mehrdeutigen Konstruktionen zu kennzeichnen. Der Autor legt beispielsweise für den Satz

(45) *Secure the gear with twelve rivets.*

fest, dass die Präpositionalphrase *with twelve rivets* an das Verb *secure* und nicht an das Nomen *gear* gebunden wird. ClearCheck markiert zu diesem Zweck die entsprechende Textstelle mit einem *attach* SGML Tag:

(46) *Secure the gear with <attach head='secure' modi='with'> twelve rivets.*

Die Sprache KCE verwendet ein bereichsspezifisches Lexikon, das aus vorhandenen

Online-Dokumenten automatisch aufgebaut und anschliessend von einem Lexikographen verfeinert wird [Mitamura et al. 93]. Jedem Wort wird eine Bedeutung zugeordnet. Wenn ein Wort in einem Anwendungsbereich mehr als eine Bedeutung hat, dann untersucht der Lexikograph, ob die überzähligen Bedeutungen des Wortes auf andere unbelegte Ausdrücke verteilt werden können. Ausdrücke, für die sich eine solche Spezialisierung finden lässt, werden im Lexikon als zusammengehörig gekennzeichnet, so dass die alternierenden Bedeutungen für ClearCheck zugänglich werden. Wenn eine Verteilung der verschiedenen Bedeutungen nicht möglich ist, dann führt das zu mehreren lexikalischen Einträgen für dasselbe Wort. Die Disambiguierung kann dann erst während der Quelltextanalyse vorgenommen werden.

Für die Definition des technischen Vokabulars unterscheidet die Sprache KCE drei Kategorien:

- Technische Phrasen (das sind oft kompositional nicht errechenbare Einheiten)
- Technische Wörter (Akronyme, Abkürzungen)
- Technische Symbole (Zahlen, Zahlwörter, Buchstaben und Masseinheiten)

Zusätzlich zur Beschränkung des Diskursbereichs gelten für den Gebrauch des Vokabulars und der Grammatik von KCE eine Reihe von Einschränkungen [Baker et al. 94, Mitamura & Nyberg 95, Nyberg & Mitamura 96]. Für das Vokabular gelten die folgenden Regeln:

- Nur konsistente Orthographie ist zulässig.
- Nur spezifizierter Gebrauch von Nominalkomposita ist erlaubt.
- Nur spezifizierter Gebrauch von Funktionswörtern ist gestattet.
- Nur spezifizierter Gebrauch von Modalverben ist erlaubt.

Die Sprache KCE unterscheidet zwischen grammatikalischen Einschränkungen auf der Phrasen- und der Satzebene. Die phrasalen Einschränkungen sind folgendermassen geregelt:

- Nominalphrasen sind mit Determinatoren zu verwenden.
  The primary power supply component will supply the necessary 240 Volts DC to the input lead.

- Quantoren und Partitive dürfen nicht allein stehen.
  Repeat these steps until none are left. -->
  Repeat these steps until no bolts are left.
- Phrasale Verben sind durch einfache Verben zu ersetzen.
  turn on --> start
- Koordinierte Verbalphrasen sind durch koordinierte Sätze zu ersetzen.
  Extend and retract the cylinders. -->
  Extend the cylinders and retract the cylinders.
- In koordinierten Präpositionalphrasen ist die Präpositionen zu wiederholen.
  5 cubic meters of concrete and sand -->
  5 cubic meters of concrete and of sand.

Auf der Satzebene gelten für die Sprache KCE unter anderen die folgenden Einschränkungen:

- Partizipialkonstruktionen in Nebensätzen sind nicht zulässig.
  After installing the gear ... -->
  After you install the gear ...
- Nur subjektsmodifizierende Relativsätze sind gestattet.
  The pumps that are mounted to the pump drive.
- Nur spezifizierte elliptische Konstruktionen sind erlaubt.
  if necessary ..., if equipped ...
- Nur gleiche Satztypen sind koordinierbar.
  Extend the cylinders and retract the cylinders.
- Nur Satzfragen, aber keine Wortfragen sind zulässig.
  Does ... ?, Is ... ?
- Standardisierte Satzzeichensetzung wird vorausgesetzt.

Ein wichtiger Punkt ist, dass die Sprache KCE - im Gegensatz zu früheren kontrollierten natürlichen Sprachen wie Caterpillar Fundamental English (CFE) - die Grösse des Vokabulars nicht einschränkt, sondern bloss diejenigen lexikalischen und grammatischen Konstruktionen reduziert, die zu unnötiger Komplexität führen. Das Resultat ist eine Sprache, die genügend expressiv ist, um technische Dokumente zu verfassen, und die von limitierter Komplexität ist, so dass Übersetzungen von hoher Qualität möglich werden.

Wie die Untersuchungsergebnisse einer Praktikabilitätsstudie zeigen, die aufgrund einer Testsatzsammlung von 750 Sätzen gemacht wurde, reduziert die Verwendung der Sprache KCE die Anzahl der möglichen syntaktischen Analysen pro Satz drastisch. Dafür ausschlaggebend ist erstens die Verwendung eines kontrollierten Vokabulars aus dem Diskursbereich, zweitens die kontrollierte Grammatik und drittens die Benutzerinteraktion während der Analysephase:

> When a constrained lexicon and grammar for the domain were utilized, along with disambiguation by the author, the average number of syntactic analyses dropped from 27.0 to 1.04. 95.6% of the sentences were assigned a single interlingua representation [Mitamura & Nyberg 95:12].

Der KANT Ansatz ist nicht als allgemeine Lösung zu werten, die auf alle maschinellen Übersetzungsprobleme passt. Der Ansatz ist speziell für technische Anwendungen ausgelegt, wo der Diskursbereich klar begrenzt und konkret ist und wo die Möglichkeit besteht, jedem Ausdruck eine eindeutige Bedeutung zuzuordnen.

4.2.2 AECMA Simplified English

In den späten 70er Jahren beauftragte die Association of European Airlines (AEA) die Association Européenne des Constructeurs de Matériel Aérospatial (AECMA), die Lesbarkeit ihrer Unterhaltsdokumente für Flugzeuge zu untersuchen. Die AECMA Documentation Working Group (DWG) gründete zu diesem Zweck die Projektgruppe Simplified English Working Group (SEWG), die mit der Untersuchung von existierenden Dokumenten begann. Anfangs der 80er Jahre stiess die Aerospace Industries Association (AIA) aus den USA zu der Projektgruppe. Aus der Zusammenarbeit entstand die kontrollierte natürliche Sprache AECMA Simplified English (SE), die heute den De-facto-Standard für Unterhaltsdokumente in der Flugzeugindustrie bildet [AECMA 95, Farrington 96].

Ausschlaggebend für die Entwicklung der Sprache AECMA SE waren zwei Gründe: Erstens sollte die Lesbarkeit der Unterhaltsdokumente für Flugzeuge verbessert werden und zweitens sollte die englische Sprache, in der diese Dokumente bis anhin geschrieben wurden, standardisiert werden. Die Verfechter von SE waren überzeugt, dass durch die kontrollierte Verwendungsweise der englischen Sprache die Unterhaltsdokumente leichter lesbar und besser verständlich werden, so dass die prozeduralen Anweisungen und die deklarativen Beschreibungen in den Texten an Klarheit gewinnen. Ausserdem erhoffte man sich Kostenreduktion und Zeitgewinn bei der

Übersetzung der Dokumente in andere Zielsprachen.

Die kontrollierte Sprache AECMA SE hat zum Ziel, den Technikern und Mechanikern zu einem eindeutigen Textverständnis zu verhelfen, damit Unterhalts- und Wartungsarbeiten an Flugzeugen möglichst optimal durchgeführt werden können. AECMA SE richtet sich sowohl an Personen mit englischer Muttersprache als auch an Personen, die Englisch als Zweitsprache erworben haben. Von beiden Personengruppen wird erwartet, dass sie mit dem Anwendungsgebiet vertraut sind.

Den Autoren steht mit dem AECMA SE Handbuch ein Instrument zur Verfügung, das den Gebrauch der Sprache für die Produktion von technischen Dokumenten regelt. Das zulässige Vokabular und die informellen Schreibregeln helfen den technischen Autoren, komplexe Instruktionen und Sachverhalte aus dem Anwendungsgebiet der Flugzeugwartung in einfacher Form darzustellen. Das Schreiben von Texten in SE verlangt von den Autoren gute Englischkenntnisse und grosse Vertrautheit mit dem AECMA SE Standard. Obwohl SE die Möglichkeiten einschränkt, wie technische Informationen dargestellt werden können, bietet die Sprache den Autoren genügend Flexibilität, um die Instruktionen und Sachverhalte in unterschiedlichen Weisen ausdrücken zu können.

Die Sprache AECMA SE besteht aus einem kontrollierten allgemeinen Vokabular (ca. 950 Wörter) und aus 57 Schreibregeln. Das allgemeine Vokabular ist ausdrucksstark genug, um die technischen Informationen aus dem Anwendungsgebiet angemessen darstellen zu können. Die einzelnen Wörter sind aufgrund von Einfachheits- und Wiedererkennbarkeitskriterien ausgewählt worden. In Fällen von Bedeutungsgleichheit (bzw. Bedeutungsähnlichkeit) wurde für das Vokabular die Entscheidung getroffen, dass immer nur ein Leitwort aus einer Reihe von Synonymen berücksichtigt wird. Beispielsweise ist das Verb *start* anstelle von *begin*, *commence*, *initiate* oder *originate* ins AECMA Vokabular aufgenommen worden. Dem Prinzip folgend: eine Bedeutung - ein Wort. Bei mehreren verfügbaren Bedeutungen für ein Wort wird nur eine Bedeutung ausgewählt, und zwar unter Ausschluss von allen anderen Bedeutungen. Dem Prinzip folgend: ein Wort - eine Bedeutung. Zum Beispiel ist das Verb *fall* nur unter der Bedeutung *to move down by the force of gravity* verfügbar und nicht unter der Bedeutung *decrease*.

Das AECMA Wörterverzeichnis gleicht einem Thesaurus. Es listet alle zulässigen Wörter als Schlüsselwörter in der ersten Spalte einer Tabelle auf (vgl. das Beispiel

unten). Diese Schlüsselwörter sind zur besseren Kennzeichnung in Grossbuchstaben geschrieben und zusammen mit der entsprechenden Wortart verzeichnet (*START (v)*). Neben diesen Wörtern erscheint in der ersten Spalte der Tabelle eine Auswahl von unzulässigen Wörtern, die in Kleinbuchstaben geschrieben sind (*actuate (v)*). Unzulässige Wörter müssen in SE durch zulässige Alternativen paraphrasiert werden. Die alternativen Vorschläge sind in Grossbuchstaben geschrieben und erscheinen in der zweiten Spalte (*START, OPERATE*). Darüber hinaus informiert die zweite Spalte über die Gebrauchsweise von zulässigen Schlüsselwörtern (*To cause movement or operation.*). Die dritte Spalte zeigt, wie die Schlüsselwörter für zulässige Wörter gebraucht werden können (*START THE APU.*) oder wie die zulässigen Alternativen zu verwenden sind (*START THE MOTOR.*). Die vierte Spalte der Tabelle führt Textbeispiele auf, die in SE nicht akzeptabel sind, da in ihnen unzulässige Schlüsselwörter gebraucht werden (*Actuate the motor.*).

Ausschnitt AECMA Wörterverzeichnis

Keyword (part of speech)	Assigned Meaning / USE	APPROVED EXAMPLE	Not Acceptable
actuate (v)	START, OPERATE	START THE MOTOR. OPERATE THE HANDPUMP.	Actuate the motor. Actuate the handpump.
. . .	. . .	. . .	. . .
START (v), STARTS, STARTED, STARTED	To cause movement or operation.	START THE APU.	

Die technischen Autoren sind frei, ausser den zulässigen Wörtern komplexe Nomen als technische Namen (= Technical Names) zu definieren oder Verben zu bestimmen, die einen Herstellungsprozess (= Manufacturing Process) bezeichnen. Sobald aber eine Verwendungsweise für ein Wort festgelegt worden ist, wird vom AECMA Standard (Rule: 1.12) verlangt, dass dieses Wort konsistent gebraucht wird, insbesondere gilt diese Konsistenzbedingung auch für Technical Names (z.B. *penetrant spray*, *logo light*) und Manufacturing Processes (z.B. *insulate*, *magnetize*).

Die 57 Schreibregeln informieren die Autoren in einer präskriptiven Art und Weise, wie sie die Sprache AECMA SE verwenden müssen. Die Schreibregeln sind in neun Sektionen unterteilt: (1) Worte, (2) Nominalphrasen, (3) Verben, (4) Sätze, (5) Prozeduren, (6) Deskriptives Schreiben, (7) Warnungen und Vorsichtsmassnahmen, (8) Satzzeichen und Worthäufigkeit, (9) Schreibtechniken. Um die informelle Form dieser Regeln zu veranschaulichen, geben wir hier je ein Beispiel für die ersten vier Sektionen:

Ausschnitt AECMA Schreibregeln

Section 1	Words
Rule: 1.5	You can use words that are Technical Names.
Section 2	Noun Phrases
Rule: 2.3	When appropriate, use an article (the, a, an) or a demonstrative adjective (this, these) before a noun.
Section 3	Verbs
Rule: 3.6	Use the active voice. Use only the active voice in procedural writing, and as much as possible in descriptive writing.
Section 4	Sentences
Rule: 4.4	Use connecting words to join consecutive sentences that contain related thoughts.

Es hat sich gezeigt, dass der Umfang des AECMA Standards beträchtliche kognitive Leistungen von den technischen Autoren erfordert. Die Autoren müssen die Schreibregeln und die detaillierte Information über zulässige und unzulässige Wörter für die Textproduktion aktiv verfügbar haben. Um die technischen Autoren im Schreibprozess zu unterstützen und um standardtreue Texte zu garantieren, ist bei Boeing ein syntaxbasierter SE Checker entwickelt worden [Wojcik et al. 90, Wojcik et al. 93, Wojcik & Holmback 96].

Technisch gesehen, beruht der SE Checker auf einem Grammatikformalismus, der stark von GPSG [vgl. Gazdar et al. 85] beeinflusst ist. Der SE Checker besteht aus vier Komponenten: einem Tokeniser, einem Parser, einer Grammatik und einem Lexikon.

Der Tokeniser identifiziert mit Hilfe des Lexikons die einzelnen Wörter, Satzzeichen und Sonderzeichen des Eingabesatzes. Der Parser analysiert die Tokens unter Verwendung der Grammatik (ca. 320 Regeln) und produziert sämtliche möglichen syntaktischen Analysen. Während der Analyse werden den einzelnen Wörtern und Konstituenten numerische Gewichte zugeordnet, aufgrund derer die Berechnung der besten Analyse erfolgt. Anschliessend wird der geparste Satz auf unzulässige Wörter und grammatikalische Fehler untersucht. Der SE Checker markiert die unzulässigen Wörter und macht Korrekturvorschläge von der Art, wie sie der technische Autor auch im AECMA Standard finden würde. Der SE Checker kann ungefähr 90% der Eingabesätze korrekt analysieren [Wojcik et al. 93:39].

Beim Entwurf des AECMA SE Checkers waren zwei Punkte ganz wesentlich: Erstens wollte man auf eine vollautomatische Textkorrektur verzichten und zweitens sollte nur eine konservative Fehlerbehandlung gemacht werden.

Der erste Punkt ist deshalb wichtig, weil eine vollautomatische Textkorrektur grosse Mengen an semantischem Wissen über den Anwendungsbereich verlangen würde. Semantische Information ist aber im syntaxbasierten SE Checker nicht verfügbar. Gegenwärtig wird der SE Checker inkrementell um eine semantische und pragmatische Komponente erweitert [Wojcik & Holmback 96:26 ff.]. Dieser Enhanced Grammar, Style and Content (EGSC) Checker setzt auf dem SE Checker auf. Die Erweiterungen überprüfen vor allem die Wortbedeutungen aufgrund von Selektionsrestriktionen und helfen, die beste syntaktische Analyse auszuwählen. Die Strategie des SE Checkers ist so, dass die semantische Komponente und die pragmatische Komponente nur Hilfsfunktionen übernehmen. Wenn keine semantische Repräsentation erzeugt werden kann, dann verlässt sich der Checker ausschliesslich auf die syntaktische Analyse.

Der zweite Punkt, d.h. die konservative Fehlerbehandlung, beruht auf der Erfahrung, dass die technischen Autoren eher akzeptieren, wenn der Checker gewisse Fehler nicht erkennt, als wenn der Checker inkorrekte Fehlermeldungen liefert [Wojcik & Holmback 96:30].

4.2.3 Perkins Approved Clear English (PACE)

Perkins Engines Ltd. ist eine englische Firma, die Dieselmotoren herstellt. Die Firma unterhält ein Vertriebsnetz in über 160 Ländern und exportiert 85% ihrer Produktion. Der grosse Exportanteil macht die Übersetzung von technischen Dokumenten in verschiedene Sprachen (vor allem Französisch, Deutsch, Italienisch und Spanisch) notwendig. Insbesondere Unterhaltsdokumente, die Mechaniker und Techniker zur Revision und Reparatur der Motoren benötigen, verlangen einen hohen Grad an Präzision und müssen - wenn immer möglich - in den entsprechenden Landessprachen vorliegen.

Um Mehrdeutigkeiten und unverständliche Ausdrucksweisen (= Jargon) aus den technischen Quelltexten zu beseitigen, hat sich die Firma Perkins Engines bereits 1980 entschlossen, die kontrollierte natürliche Sprache Perkins Approved Clear English (PACE) einzuführen. Ausser besseren Resultaten für konventionelle und automatische Übersetzungen erhoffte sich die Firma von der Sprache PACE auch eine Verbesserung der Verständlichkeit von Quelltexten für Nichtmuttersprachler. Dieser Aspekt ist deshalb wichtig, weil aus Kostengründen nicht alle technischen Dokumente in die entsprechenden Zielsprachen übersetzt werden können [Pym 90].

Die Sprache PACE besteht aus einem kontrollierten Wörterbuch und zehn einfachen Schreibregeln. Das Wörterbuch führt all die Wörter auf, die in den technischen Dokumenten zulässig sind. Von den 2500 Einträgen sind ungefähr 10% Verben. Der Aufbau des Wörterbuchs ist durch das Prinzip "ein Wort für eine Bedeutung" bestimmt und verzeichnet für jedes zulässige Wort genau eine Bedeutung, einschliesslich Artikel, Konjunktionen, Pronomen und Präpositionen.

Beispielsweise ist das Adjektiv *right* nur im Bedeutungsgegensatz zu *left* verfügbar. Der Gebrauch von *right* im Sinn von *correct* ist ausgeschlossen. Das Adjektiv *correct* wiederum kann nur in der Bedeutung von *conforming to a standard, opposite of wrong* verwendet werden. Im Gegensatz zu diesen Wörtern sind technische Ausdrücke nicht definiert, da sie als bekannt vorausgesetzt werden [Pym 90:85]. Im Fall von Homographen bestimmt das Wörterbuch diejenigen Wortarten, die verwendet werden können. Zum Beispiel kann das Wort *seal* in PACE sowohl als Nomen wie auch als Verb gebraucht werden, während das Wort *stroke* bloss als Nomen zulässig ist [Newton 92:47].

Die Sprache PACE wird durch zehn informelle Schreibregeln vervollständigt. Diese

Schreibregeln geben den technischen Autoren Ratschläge, damit sie kurze und einfache Sätze bilden und bestimmte Konstruktionen explizit machen. Die Regeln wurden zusammen mit einem Fachmann der Universität Bradford ausgearbeitet. Das Ziel bestand darin, die Instruktionen und technischen Sachverhalte bereits in den Quelltexten so präzis und klar darzustellen, dass der Text möglichst eindeutig übersetzt werden kann. Das Resultat der Übersetzung sollte nicht mehr mit grossem Aufwand nachbearbeitet werden müssen. Die zehn Schreibregeln sind in zwei Gruppen von Regeln bzw. Ratschlägen aufgeteilt [Pym 90:85 ff.]:

PACE Schreibregeln

With rules 1-5 the general advice is to keep it short and simple

1. keep sentences short
2. omit redundant words
3. order the parts of the sentences logically
4. do not change constructions in mid sentence
5. take care with the logic of *and* and *or*.

The general advice of rules 6-10 is to make it explicit

6. avoid elliptical constructions
7. do not omit conjunctions or relatives
8. adhere to the PACE dictionary
9. avoid strings of nouns
10. do not use *-ing* unless the word appears thus in the PACE dictionary.

Ausser dem zulässigen Vokabular sind im PACE Wörterbuch Textbeispiele zu finden, die illustrieren, wie Sätze in den PACE Standard überführt werden können. Im folgenden Beispiel werden die Schreibregeln 5, 8 und 9 auf einen Satz der vollen natürlichen Sprache angewendet:

(47) *Loosen the dynamo or alternator mounting and adjustment link fasteners.*

Als Ergebnis der Regelanwendungen resultierten zwei PACE Sätze:

(48) *Loosen the pivot fasteners of the dynamo or of the alternator.*

(49) *Loosen also the fasteners of the adjustment link.*

Die 5. Schreibregel sorgt dafür, dass die koordinierten Konstituenten besser sichtbar werden. Die Information wird auf zwei Sätze verteilt, wobei das Verb wiederholt werden muss. Die 8. Schreibregel schliesst im Satz (48) aus, dass das Nomen *mounting* für die Bezeichnung von Drehverschlüssen (*pivot fasteners*) verwendet wird. *Mounting* ist im Wörterbuch nur unter der Bedeutung *bracket used to support a piece of equipment* verfügbar. Die 9. Schreibregel führt im Satz (49) dazu, dass das zusammengesetzte Nomen *adjustment link fasteners* eine explizite Struktur bekommt (*fasteners of the adjustment link*), so dass der Kopf (*fasteners*) und der Modifikator (*adjustment link*) der Struktur sichtbar werden. PACE Sätze dürfen nicht länger als maximal 15-20 Wörter sein. Entsteht bei der Umformung ein längerer Satz, dann müssen aus diesem zwei Sätze gebildet werden.

Die Verwendung der kontrollierten natürlichen Sprache PACE hat zu einer der erfolgreichsten Anwendungen in der Geschichte der maschinellen Übersetzung geführt [Newton 92:46]. Der PACE Ansatz zeigt, was erreicht werden kann, wenn eine kontrollierte natürliche Sprache in einem begrenzten (technischen) Anwendungsbereich zum Einsatz kommt. Die PACE Texte, die bei Perkins Engines entstehen, sind Modelle von stilistischer Homogenität und terminologischer Konsistenz [Newton 92:47]. Das kontrollierte Vokabular und die kleine Menge der Schreibregeln führen bereits zu einer verblüffenden qualitativen Verbesserung der übersetzten Texte. Dank der Verwendung von PACE konnte die Firma Perkins Engines die Übersetzungskosten und die Übersetzungsdauer beträchtlich reduzieren - in einigen Fällen um 50 bis 70% [Pym 93:4]. Zur Zeit entwickelt Perkins Engine zusammen mit der Language Technology Group der Universität Edinburgh einen syntaktischen Checker für die Sprache PACE, um die technischen Autoren beim Schreibprozess noch besser unterstützen zu können [Douglas & Hurst 96].

4.3 Subsprachen als Spezifikationssprachen

Es werden nun eine Reihe von Ansätzen untersucht, die eine Subsprache als Spezifikationssprache nutzen, um möglichst eindeutige und präzise SASen zu schreiben. Diese Subsprachen werden durch den Computer analysiert und in eine formale Sprache übersetzt. Wie sich zeigen wird, verwenden diese Ansätze Eigenschaften der zugrundeliegenden Subsprache, um die Spezifikationssprache implizit zu beschränken. Der Nachteil bei dieser Vorgehensweise ist, dass die Spezifikationssprachen in die Abhängigkeit des jeweiligen Anwendungsgebietes geraten und nur schwer universell

nutzbar sind. Bei diesem Überblick konzentrieren wir uns auf die Probleme, zu denen diese Vorgehensweise führt, und interessieren uns vor allem für die folgenden drei Punkte: Erstens für die linguistischen Eigenschaften der vorgestellten SASen, mit denen die einzelnen Ansätze getestet worden sind. Zweitens für die Rolle, die den Systembenutzern im Spezifikationsprozess zukommt, und drittens für die technische Realisierung der einzelnen Systeme. Nicht immer sind diese drei Punkte für alle Ansätze in den Veröffentlichungen in voller Breite ausgearbeitet worden. Hinzu kommt, dass diese Ansätze meistens nur in prototypischen Realisierungen überprüft werden konnten. Trotzdem liefert die Diskussion aller Ansätze zusammen einen guten Ausgangspunkt, um auf Schwachstellen aufmerksam zu machen, die auftreten, wenn eine Subsprache (mehr oder weniger ad hoc) als Spezifikationssprache eingesetzt wird.

4.3.1 SAFE

In [Balzer et al. 78] wird das Spezifikationssystem SAFE (Specification Acquisition From Experts) vorgestellt. Der SAFE Prototyp ist ein ganz früher Versuch, ausführbare Spezifikationen aus partiellen Beschreibungen abzuleiten, die in einer normalisierten Form der natürlichen Sprache vorliegen. Die grundlegende Idee dieses Ansatzes besteht darin, die partiellen Beschreibungen während des Ableitungsprozesses mit Hilfe eines computergestützten Werkzeugs, das auf kontextuelles Wissen zugreifen kann, zu vervollständigen. Der Ableitungsprozess setzt sich aus drei Phasen (Linguistik, Planung, Meta-Evaluation) zusammen: Während der linguistischen Phase liest der SAFE Prototyp die syntaktisch normalisierte natürlich-sprachliche SAS und teilt den Input in eine strukturelle und eine prozessorientierte Beschreibung auf. Anschliessend wird die strukturelle Beschreibung aufgrund von anwendungsspezifischem Wissen vervollständigt, das in einer Wissensbasis vorliegt oder durch Benutzerinteraktion verfügbar wird. Während der Planungsphase wird die grobe Programmstruktur aus der ergänzten strukturellen Beschreibung und der prozessorientierten Beschreibung erzeugt. In der abschliessenden Meta-Evaluationsphase wird die grobe Programmstruktur so verfeinert, dass eine ausführbare Spezifikation beziehungsweise ein lauffähiges Programm - in der abstrakten Sprache AP2 - entsteht, das eine relationale Datenbank als Grundlage für die Datenmanipulation benutzt.

Wir interessieren uns hier weniger für den Ableitungsprozess als für die linguistischen Eigenschaften der syntaktisch normalisierten natürlich-sprachlichen SAS, die den Ausgangspunkt für den Ableitungsprozess bildet. Bevor wir einen Textausschnitt, der mit dem SAFE Prototyp getestet worden ist, linguistisch genauer untersuchen, brauchen wir einige Hintergrundsinformationen über den Problemkontext.

Der zu untersuchende Spezifikationstext beschreibt eine Multiplexerkomponente in einem Kommunikationssystem für Satelliten. Die Komponente hat die folgenden Aufgaben: Sie empfängt vom Kontroller des Satellitenkanals eine Tabelle (*SOL*). Die Tabelle besteht aus einem Eintrag für einen Abonnenten (*SUBSCRIBER*) und Einträgen für sonstige Benutzer (*RATS*). Ein Abonnenteneintrag setzt sich aus der Benutzeridentifikation und der Übertragungslänge zusammen. Für die übrigen Benutzer sind bloss die Übertragungslängen in der Tabelle vermerkt. Aufgrund der Tabelle berechnet die Komponente die Übertragungsdauer *(RELATIVE TRANSMISSION TIME)* und den Übertragungszeitpunkt *(CLOCK TRANSMISSION TIME)*. Das Ergebnis der Berechnung wird - falls es sich um einen Abonnenten handelt - in einen (neuen) Übertragungszeitplan (*TRANSMISSION SCHEDULE*) eingetragen. Die Ergebnisse für die übrigen Benutzer werden der Reihe nach dem Abonnenteneintrag nachfolgend in den Übertragungszeitplan eingefügt. Der Übertragungszeitplan wird anschliessend von einer weiteren - hier nicht spezifizierten - Komponente gelesen, die den Satellitenkanal zu den entsprechenden Zeitpunkten schaltet. Soweit zu den Sachverhalten, die den Ausgangspunkt für den folgenden Spezifikationstext bilden:

(50) *((THE SOL)*
(IS SEARCHED)
FOR
(AN ENTRY FOR (THE SUBSCRIBER)))

(51) *(IF ((ONE)*
(IS FOUND))
((THE SUBSCRIBER´S (RELATIVE TRANSMISSION TIME))
(IS COMPUTED) ACCORDING-TO ("FORMULA-1")))

(52) *((THE SUBSCRIBER´S (CLOCK TRANSMISSION TIME))*
(IS COMPUTED) ACCORDING-TO ("FORMULA-2")))

(53) *WHEN ((THE (TRANSMISSION TIME))*
(HAS BEEN COMPUTED))
((IT)
(IS INSERTED)
AS (THE (PRIMARY ENTRY))
IN (A (TRANSMISSION SCHEDULE))))

(54) *FOR (EACH RATS ENTRY)*
(PERFORM)
(: ((THE RATS´S (RELATIVE TRANSMISSION TIME))
(IS COMPUTED) ACCORDING-TO ("FORMULA-1"))
((THE RATS´S (CLOCK TRANSMISSION TIME))
(IS COMPUTED) ACCORDING-TO ("FORMULA-2"))))

(55) *((THE RATS (TRANSMISSION TIMES))*
(ARE ENTERED)
INTO (THE SCHEDULE))

Es fällt sofort auf, dass der Text bereits vor der maschinellen Verarbeitung syntaktisch normalisiert worden ist. Um syntaktische Mehrdeutigkeiten bei der Analyse zu vermeiden, sind die Konstituenten in den Sätzen von Hand geklammert worden. Andere linguistische Probleme (z.B. fehlende Operatoren, unterspezifizierte Referenz, implizite Information infolge von Passivkonstruktionen) werden erst bei der Verarbeitung mit Hilfe der anwendungsspezifischen Wissensbasis oder durch Benutzerinteraktion aufgelöst.

Klammern kennzeichnen zusammengehörende Konstituenten. Zum Beispiel ist die Nominalphrase *(THE SUBSCRIBER´S (RELATIVE TRANSMISSION TIME))* in (51) als Struktur bestimmt worden, die aus einem inneren nominalen Teil *(RELATIVE TRANSMISSION TIME)* und einem äusseren attributiven Teil *(THE SUBSCRIBER´S)* besteht. Neben der syntaktischen Strukturierung übernehmen die Klammern auch die Rolle von Satzzeichen. Ausser dem Doppelpunkt in (54) finden wir keine expliziten Satzzeichen im Text.

Bindestriche fassen einzelne Wörter zu Einheiten zusammen. Beispielsweise kennzeichnet ein Bindestrich, dass es sich bei der zusammengesetzten Präposition *ACCORDING-TO* in (51) um eine Einheit handelt. Ebenso fasst ein Bindestrich in (51) die Bestandteile des Ausdrucks *"FORMULA-1"* zusammen. Ausserdem zeigen die Anführungszeichen an, dass es sich hier um einen Eigennamen handelt.

Mit Ausnahme der imperativen Verbform *PERFORM* in (54) stehen alle Verben im Indikativ. Sie werden in der passiven Form gebraucht und ihre Zeitformen sind entweder Präsens (simple present) oder Perfekt (simple present perfect). Die passiven Verbformen beschreiben Ereignisse, ohne explizit zu machen, wer oder was diese Ereignisse verursacht. Beispielsweise wird in (51) vorausgesetzt, dass es einen Agen-

ten gibt, der die Übertragungsdauer eines Abonnenten aufgrund einer bestimmten Formel berechnet. Solche "täterabgewandten" Aussagen könnten vermieden werden, wenn der Autor das grammatische Subjekt durch eine Präpositionalphrase explizit nennen oder im Satz eine aktive Verbform verwenden würde. Anderenfalls muss diese Information während der linguistischen Verarbeitung durch die anwendungsspezifische Wissensbasis von SAFE oder durch Benutzerinteraktion zur Verfügung gestellt werden.

Der Spezifikationstext enthält einfache und zusammengesetzte Sätze. Der einfache Satz (50) wird durch die definite Nominalphrase *(THE SOL)* im Singular eingeleitet, die eine bestimmte Tabelle (*SOL*) bezeichnet. Im einfachen Satz (55) wird die definite Nominalphrase *(THE RATS (TRANSMISSION TIMES))* im Plural gebraucht. Damit der Satz überhaupt korrekt interpretiert werden kann, muss der bestimmte Artikel *THE* durch den Allquantor *EACH* ersetzt werden.

Die zusammengesetzten Sätze treten in Form von Satzgefügen oder Satzverbindungen auf. Der zusammengesetzte Satz (51) wird durch den konditionalen Nebensatz *IF ((ONE) (IS FOUND))* eingeleitet und bildet zusammen mit dem nachfolgenden Hauptsatz ein Satzgefüge. Der konditionale Nebensatz setzt eine Bedingung voraus, die erfüllt sein muss, damit das im Hauptsatz ausgedrückte Ereignis zutreffen kann. Genaugenommen sind es zwei Hauptsätze, deren Ereignisse im Skopus der im konditionalen Nebensatz ausgedrückten Bedingung liegen. Neben dem Hauptsatz in (51) hängt auch der Satz (52) vom Nebensatz in (51) ab. Um diese Abhängigkeit explizit zu machen, muss für die linguistische Verarbeitung der fehlende Operator *AND THEN* zwischen die beiden Hauptsätze eingefügt werden. Der zusammengesetzte Satz (53) besteht aus einem einleitenden temporalen Nebensatz *WHEN ((THE (TRANSMISSION TIME)) (HAS BEEN COMPUTED))*, indem das Genannte zeitlich dem im Hauptsatz Gegebenen vorangeht.

Interessant ist der Satz (54), weil es sich hier im ersten Teilsatz mit der imperativen Verbform, um einen allquantifizierten Satz handelt, der zwei Hauptsätze in seinem Skopus hat. Die syntaktische Gleichrangigkeit der beiden Hauptsätze muss für die Verarbeitung durch die Ergänzung des fehlenden Operators *AND* explizit gemacht werden.

Der Spezifikationstext enthält Sätze, die durch anaphorische Verweisformen untereinander verknüpft sind. Eine anaphorische Verweisform und ihr Antezedens (z.B. eine

vorausgehende Nominalphrase) denotieren jeweils das gleiche Objekt. Beispielsweise bezieht sich das indefinite Pronomen *(ONE)* in (51) auf die Nominalphrase *(AN ENTRY FOR (THE SUBSCRIBER))* in (50) und das Personalpronomen *(IT)* in (53) auf die definite Nominalphrase *(THE (TRANSMISSION TIME))* im gleichen Satz. Untersucht man die Verwendungsweise der definiten Nominalphrase *(THE (TRANSMISSION TIME))* im Satz (53) genauer, dann stellt man fest, dass sie wiederum referentiell gebraucht wird. Ohne Sachkenntnisse ist es schwierig, das korrekte Antezedens aufzufinden, da die definite Nomialphrase *(THE (TRANSMISSION TIME))* unterspezifiziert ist. Im Textausschnitt stehen zwei Kandidaten zur Auswahl: Die textuell nähere Nomialphrase *(THE SUBSCRIBER´S (CLOCK TRANSMISSION TIME))* in (52) oder die textuell weiter entfernte Nominalphrase *(THE SUBSCRIBER´S (RELATIVE TRANSMISSION TIME))* in (51). Der korrekte anaphorische Bezug kommt nur bei der Wahl der näherliegenden Konstituente zustande. Wie der Spezifikationstext zeigt, ist diese Art von unsicherer kontextueller Referenz kein Einzelfall. Ähnliche Problem tauchen bei der Verarbeitung der definiten Nominalphrase *(THE SCHEDULE)* in (55) auf.

Es sind solche linguistischen Phänomene, die in unkontrollierten natürlich-sprachlichen Spezifikationstexten zu Fehlinterpretationen mit schwerwiegenden Folgen für den nachfolgenden Software-Entwicklungsprozess führen. Der SAFE Prototyp versucht diese Probleme zu lösen, indem der natürlich-sprachliche Text bereits vor der linguistischen Verarbeitung soweit wie möglich syntaktisch normalisiert und disambiguiert wird und dann während der Verarbeitung mit Hilfe einer anwendungsspezifischen Wissensbasis oder durch Benutzerinteraktion vervollständigt wird. Wie wir bereits gesehen haben und [Balzer et al. 78:232] in ihrer Arbeit auch betonen, ist der Aufbau einer solchen Wissensbasis eine arbeitsintensive und fehleranfällige Angelegenheit. Sie führt im Prinzip zu der gleichen Art von Problemen wie das Schreiben einer eindeutigen, konsistenten und vollständigen SAS, wobei der zusätzliche Abstraktionsprozess das Problem eher noch verschärft.

4.3.2 TELL

In [Saeki et al. 89] leitet ein Modellierer mit Hilfe eines Hypertextsystems eine formale Spezifikation aus einer natürlich-sprachlichen Problembeschreibung ab. Als Fallbeispiel wird von einer Problembeschreibung für ein Liftkontrollsystem ausgegangen. Das Ergebnis bildet eine formale Spezifikation in der Sprache TELL, deren Semantik durch eine temporale Logik gegeben ist. Der Ableitungsprozess besteht aus zwei Aktivitäten: Entwurf und Ausarbeitung. Während des Entwurfs (erste Aktivität) wird die Systemstruktur interaktiv aus der informellen Problembeschreibung extrahiert und als

Ergebnis ein Modulentwurfsdokument erzeugt, das das Gerüst für die formale Spezifikation bildet. Zu diesem Zweck werden alle Nomen und Verben, die in der Problembeschreibung auftauchen mit einem Softwarekonzept aus der objektorientierten Modellierung (z.B. Klasse, Attribut, Methode und Beziehung/Botschaft) assoziiert. Da eine homomorphe Abbildung zwischen der natürlich-sprachlichen Beschreibung und dem objektorientierten Modell nicht immer möglich ist, müssen vor allem die Verben in "atomare Prädikate" zerlegt werden. Der Konstruktionsprozess verläuft interaktiv: Der Computer bestimmt automatisch nominale und verbale Kandidaten aus der informellen Problembeschreibung aufgrund von vorliegenden Wörterbüchern, ohne den Text jedoch zu parsen. Der Modellierer wählt die Kandidaten aus und klassifiziert sie zusammen mit ihren semantischen Rollen in Tabellen. Synonyme ersetzt der Modellierer durch ihr Leitwort. Die Tabellen und eine Reihe von Regeln, die kausale Beziehungen zwischen Ereignissen beschreiben, helfen die objektorientierten Softwarekonzepte zu identifizieren und das externe Zusammenspiel der Klassen zu bestimmen. Während der Ausarbeitung (zweite Aktivität) wird das interne Verhalten der Objekttypen bestimmt. Zu diesem Zweck wird sowohl die natürlich-sprachliche Problembeschreibung als auch das Modulentwurfsdokument in mehreren Zyklen verfeinert und umgeschrieben bis jeder Teil der (informellen) Problembeschreibung semantisch mit einem Teil der formalen Spezifikation in Bezug gebracht werden kann. Der Ansatz zeigt, dass der Übergang von einer natürlich-sprachlichen Problembeschreibung zu einem objektorientierten Modell äusserst schwierig ist. Da die Abbildung zwischen den beiden Beschreibungsebenen nicht immer homomorph ist, kann es auch kein einfaches Regelwerk geben, wonach sich diese Transformation automatisieren lässt. Es braucht immer einen Modellierer mit viel Sachwissen, der ausgehend von der natürlich-sprachlichen Problembeschreibung die notwendigen Umstrukturierungen für das objektorientierte Modell vornimmt.

4.3.3 Ishihara

In [Ishihara et al. 92] wird eine Methode für die Übersetzung einer natürlich-sprachlichen Protokollspezifikation (OSI - Open System Interconnection session protocol specification) in eine algebraische Spezifikation besprochen. Protokollspezifikationen beschreiben Sequenzen von Ereignissen, die durch ein Programm (protocol machine) ausgeführt werden. Da die natürlich-sprachlichen Sätze oft nur implizit beschreiben, zu welchem Zeitpunkt die Ereignisse ausgeführt werden, muss die Reihenfolge der einzelnen Sätze explizit gemacht werden. Die Methode löst dieses Problem, indem die Sätze mit Hilfe einer unifikationsbasierten Grammatik in die algebraische Spezifikationssprache ASL übersetzt werden und dann kontextuelle Information und Eigen-

schaften von Datentypen dazu verwendet werden, um die korrekte Reihenfolge der Ereignisse für die Ausführung der Spezifikation zu bestimmen. Die resultierende algebraische Spezifikation legt für jedes einzelne Ereignis die Vor- und Nachbedingungen explizit fest, so dass die Protokollmaschine die Ereignisse abarbeiten kann. Linguistisch interessant ist der natürlich-sprachliche Spezifikationstext, der den Ausgangspunkt für die Übersetzung bildet. Auf der einen Seite ist der Text syntaktisch normalisiert worden. Zusammengesetzte nominale Ausdrücke stehen in eckigen Klammern (*[T-DISCONNECT indication]*) und werden als syntaktische Einheiten behandelt. Diese Einheiten sind zuvor in der Spezifikation definiert worden, genauso wie die vielen Abkürzungen (*SPDU*, *SPM*, *TIM*) die im Text verwendet werden. Der Text enthält nur eine kleine Anzahl von Satzmustern, vor allem konditionale und temporale Teilsatzbeziehungen. Auf der anderen Seite bleibt der Text unkontrolliert: Die Satzlängen variieren beträchtlich und umfassen zwischen 5-50 Wörter, wobei ein Satz im Durchschnitt 15 Wörter enthält. Schwierigkeiten bereitet vor allem die mangelhafte textuelle Kohäsion. Da der Textzusammenhang nur ungenügend durch grammatische Mittel hergestellt wird, muss die abgeleitete algebraische Spezifikation durch eine grosse Menge von kontextueller Information in Form von Axiomen vervollständigt werden.

4.3.4 Kitss

In [Nonnenmann & Eddy 93] wird das natürlich-sprachliche Spezifikationssystem Kitss (Knowledge-Based Interactive Test Script System) vorgestellt, das für die Beschreibung von funktionalen Testfällen für Black-Box-Tests entwickelt worden ist. Beim Anwendungsgebiet handelt es sich um ein grosses Telefonsystem. Kitss übersetzt Testfallspezifikationen, die in einer englischen Subsprache (telephonese) geschrieben sind, in eine Menge von logischen Formen. Die Übersetzung kommt mit Hilfe eines statistischen Parsers zustande, der speziell für die Subsprache konfiguriert wurde. Der Parser berechnet nicht nur die syntaktischen Strukturen aufgrund von statistischen Wahrscheinlichkeiten, sondern auch den Skopus von Quantoren und die Referenzresolution. Wenn die statistische Analyse nicht zum Erfolg führt, greift der Parser für die Disambiguierung entweder auf ein statisches Bereichsmodell zurück oder verlangt die notwendige Information vom Benutzer. Die erzeugten logischen Formen werden durch eine Analysekomponente zu Episoden zusammengefasst, die gegenüber einem dynamischen Bereichsmodell auf Vollständigkeit geprüft werden und dann allenfalls ergänzt werden müssen [vgl. Nonnenmann & Eddy 93:29 ff.]. Wenn bei diesem Prozess Unklarheiten auftauchen, muss der Benutzer einzelne Passagen der Testfälle genauer erklären. Das Ergebnis ist ein vollständiger Testfall in

temporaler Logik, der anschliessend in ein ausführbares Testskript übersetzt wird. Die Testfälle haben ein festes Format (Ziel/Aktion/Verifikation). Die Sätze in diesen Testfällen stehen entweder im Indikativ oder im Imperativ. Die Struktur der Sätze ist sehr einfach und der Textzusammenhang wird durch formale Mittel der Grammatik - Funktionswörter und Rekurrenz (= Wiederholung gleicher sprachlicher Elemente) - klar gemacht. Negativ fällt auf, dass hier in nominalen Ausdrücken oft die Determinatoren fehlen (*Station B receives confirmation tone*) und dass reduzierte Relativsätze (*The status lamp associated with the CF button at B is lit*) und andere Partizipialkonstruktionen (*Activate call forwarding using CF access code*) verwendet werden. Diese grammatischen Konstruktionen können zu verschiedenen Analysen führen.

4.3.5 GATOR

In [Dankel et al. 94] unterstützt ein natürlich-sprachliches Spezifikationssystem mit dem Namen GATOR (GATherer Of Requirements) einen Entwickler bei der Beschreibung von neuen Anforderungen für eine Fernsprechanlage. Im Endausbau soll das System überprüfen, wie neue Anforderungen auf bereits vorliegende Anforderungen einwirken. Der Entwickler spezifiziert seine Anforderungen in der aus dem Anwendungsgebiet vertrauten Fachsprache. Für jeden Satz führt GATOR zuerst eine lexikalische Analyse durch, bei der Idiome und fachspezifische Wörter normalisiert werden. GATOR verwendet für die syntaktische Analyse einen Chart-Parser und gewichtet die einzelnen Analysen statistisch. Für die beste Analyse wird während der semantischen Interpretation ein partielles Frame (utterance frame) konstruiert. Dieses Frame wird durch Weltwissen, das in einer Interlingua vorliegt, vervollständigt und anschliessend in eine ausführbare Sprache konvertiert. Wenn ein natürlich-sprachlicher Satz mehrdeutig ist, dann muss der Entwickler die von GATOR vorgeschlagene Interpretation überprüfen und den Satz allenfalls neu formulieren. In der vorliegenden Version ist der Entwickler dafür verantwortlich, dass die in den natürlich-sprachlichen Sätzen ausgedrückte Information konsistent ist. Werkzeuge für die Konsistenz- und Integritätsprüfung sind geplant. Die Interlingua für GATOR ist aus grossen Textkorpora extrahiert worden. Die Verwendungsweise der einzelnen Wörter wurde parametrisiert und stark eingeschränkt. Für die gegenwärtige Anwendung wird davon ausgegangen, dass eine Interlingua ausreicht, die ungefähr 100-150 Verben, 400-500 Nomen, 50-60 Funktionswörter und 300 Substitutionen für Idiome und fachspezifische Wörter umfasst. Es stellt sich die Frage, wie portabel der Interlingua-Ansatz ist, wenn man davon ausgeht, dass für jedes neue Anwendungsgebiet eine Korpusanalyse notwendig ist. Ausserdem muss sorgfältig überlegt werden, ob der Anwendungsspezialist oder der Softwareentwickler die Interlingua aufbauen soll.

4.3.6 SAREL

In [Castell & Hernández 95] wird das Werkzeug SAREL vorgestellt, das die Softwareentwickler beim Schreiben von natürlich-sprachlichen Spezifikationen unterstützt, die Schreibnormen überprüft und für die Qualitätskontrolle sorgt. Die Spezifikationssprache (Englisch) ist auf die Softwareentwicklung in der Flugzeugindustrie ausgerichtet. SAREL besteht aus drei Modulen: Das erste Modul (Style Refinement Module) kontrolliert, ob die funktionalen Anforderungen den Schreibnormen genügen und führt zu diesem Zweck eine lexikalische Analyse, eine syntaktisch-semantische Analyse, eine Ambiguitätskontrolle (ambiguity control) und eine Einfachheitskontrolle (simplicity control) durch. Nach erfolgter Übersetzung konstruiert das zweite Modul (Conceptual Refinement Module) aus der semantischen Repräsentation mit Hilfe einer framebasierten Wissensbasis (Knowledge Base) eine konzeptionelle Repräsentation. Diese konzeptionelle Repräsentation wird zu den bereits bestehenden Anforderungen in Beziehung gesetzt, die in einer zweiten Wissensbasis (Requirements Base) formalisiert vorliegen. Dabei wird kontrolliert, ob die eingehende Anforderung neue Information enthält (duplication control). Das dritte Modul (Software Quality Control Module) überprüft eine Reihe von Software-Qualitätsattributen, bevor die konzeptionelle Repräsentation in die Requirements Base eingefügt wird. Linguistisch interessant ist das erste Modul. Für die lexikalische Analyse verwendet SAREL drei unterschiedliche Lexika: Ein erweitertes Lexikon, das alle Wörter aus dem Anwendungsbereich enthält. Ein zulässiges Lexikon, das das erweiterte Lexikon einschränkt und ein allgemeines Lexikon, das den Grundwortschatz der englischen Sprache beinhaltet. Während der lexikalischen Analyse werden alle Wörter, die zum erweiterten Lexikon gehören durch Synonyme aus dem zulässigen Lexikon ersetzt. Für die syntaktisch-semantische Analyse wird der ALVEY Parser [Briscoe et al. 87] verwendet, der für jeden Satz eine oder mehrere logische Repräsentationen liefert. SAREL löst ambige Sätze durch eine Reihe von vorgegebenen Regeln (ambiguity control) auf, die für bestimmte syntaktische Strukturen nur eine Lesart zulassen. Bei Fällen von semantischer Ambiguität muss der Entwickler die korrekte logische Repräsentation auswählen. Wenn SAREL aufgrund der Einfachheitskontrolle feststellt, dass mehrere funktionale Anforderungen in einem zusammengesetzten Satz auftauchen, dann wird vom Entwickler verlangt, dass er die Sachverhalte in einfachen Sätzen beschreibt. Der heikle Punkt bei diesem Ansatz ist das Conceptual Refinement Module, denn für den Anwendungsspezialisten besteht keine Möglichkeit in natürlicher Weise zu überprüfen, ob das konzeptionelle Modell seiner Intention entspricht.

4.3.7 SpecTran

In [Nelken & Francez 95] wird das interaktive System SpecTran besprochen, das die Übersetzung von "relativ uneingeschränkten" natürlich-sprachlichen Spezifikationen in temporale Logik unterstützt. SpecTran nimmt einen Spezifikationstext als Input, analysiert jeden Satz im Kontext der vorausgehenden Sätze und repräsentiert das Ergebnis der Übersetzung mit Hilfe von Diskursrepräsentationstheorie (DRT) in einer einzigen Diskursrepräsentationsstruktur (DRS) [vgl. Kamp & Reyle 93]. Die DRS dient als Zwischendarstellung und wird in einem weiteren Schritt in die temporale Logiksprache ACTL [Grumberg & Kurshan 94] übersetzt. Die erzeugten temporalen Formeln bilden den Input für einen Modellchecker. Der SpecTran Ansatz hat zum Ziel, den Verifikationsprozess zu erleichtern.

SpecTran verwendet für die Analyse des natürlich-sprachlichen Spezifikationsdiskurses einen unifikationsbasierten Grammatikformalismus. Falls ein Satz mehrdeutig ist, produziert der Parser alle alternativen syntaktischen Strukturen zusammen mit den entsprechenden DRSen. Für die Auflösung von Mehrdeutigkeit konsultiert SpecTran den Benutzer. Die Autoren von SpecTran räumen ein, dass die volle natürliche Sprache ein ungeeigneter Werkstoff ist, um präzise Spezifikationen zu schreiben, unterlassen es aber, genaue Kriterien anzugeben, wie die verwendete Sprache einzuschränken wäre.

> While completely unrestricted NL is beyond the reach of current technology, and is arguably an undesirable medium for expressing specifications, we allow the use of relatively convenient language within certain restrictions [Nelken & Francez 95:10].

Schauen wir zur Illustration einen kurzen Textausschnitt an, der von SpecTran verarbeitet wird. Der Ausschnitt stammt aus einer SAS für einen Verteiler (allocator). Der Verteiler A verteilt eine Ressource von m verschiedenen Prozessen (customer processes) C_1, C_2, ..., C_m. Die Kommunikation zwischen A und C_i kommt durch ein Paar von gemeinsamen booleschen Variablen r_i (request) und g_i (grant) zustande. Der folgende Ausschnitt beschreibt eine Zuweisung während eines Prozesses:

(56) *One cycle after r_i is activated, g_i should be asserted.*

(57) *r_i is deactivated one to six cycles later.*

(58) *Afterwards, it should be deasserted.*

Der Textausschnitt ist sprachlich sehr komplex und kompakt. In den drei Sätzen wird

nicht sauber zwischen faktischer und modaler Aussageweise unterschieden. Man findet sowohl passive Verbformen im Indikativ (*is activated*) als auch passive Verbformen mit modalartigem Charakter (*should be asserted*). In keinem der drei Sätze wird der Agent genannt, der die Ereignisse oder Zustände verursacht. Im Satz (58) taucht mit dem indefiniten Personalpronomen *it* eine anaphorische Verweisform auf, die sprachliches Wissen und kontextuelle Information verlangt, damit sich das korrekte Antezedens g_i in (56) erschliessen lässt.

Die drei temporalen Adverbien *after*, *later*, *afterwards* stiften (partielle) Kohäsion auf der Textoberfläche, indem sie signalisieren, wie die Ereignisse und Zustände auf der Zeitachse eingebettet werden. Beispielsweise bildet das Adverb *after* in (56) eine temporale Relation, die das formalisierte Ereignis (`e1:activated(X)`) aus dem ersten Teilsatz als erstes Argument und den formalisierten Zustand (`s1:asserted(Y)`) aus dem zweiten Teilsatz als zweites Argument nimmt. Für die beiden Adverbien *later* in (57) und *afterwards* in (58) sind die Argumente nicht in der gleichen Weise an der Textoberfläche ablesbar, sondern es braucht Inferenz und Sachwissen, um die temporalen Zusammenhänge erschliessen zu können. Beispielsweise wird die komplexe Nominalphrase *one to six cycles later* mit dem temporalen Adverb *later* in (57) elliptisch gebraucht, denn das zweite Argument von *later* (later than what?) erscheint nicht explizit an der Textoberfläche. Nur mit Hilfe von aussersprachlichem Sachwissen lässt sich erschliessen, dass es sich beim fehlenden Argument von *later* um den Zustand (`s1:asserted(Y)`) handelt, der im zweiten Teilsatz von (56) zum Ausdruck gebracht wird.

4.3.8 Schwachstellen von Subsprachen als Spezifikationssprachen

Alle diskutierten Ansätze versuchen, entweder Eigenschaften der Subsprache auszunutzen oder sogar Eigenschaften der zugrundeliegenden formalen Sprache zu verwenden, um die Spezifikationssprache implizit zu beschränken. Diese Vorgehensweise führt dazu, dass die einzelnen Spezifikationssprachen in eine starke Abhängigkeit vom Anwendungsgebiet geraten und für die korrekte Verarbeitung viel kontextuelle Hintergrundsinformation in einer Wissensbasis modelliert werden muss. Wie und auf welcher Abstraktionsebene dieses Wissens modelliert werden soll, wird in den Veröffentlichungen kaum angesprochen. Ausserdem fällt auf, dass die Systembenutzer ständig damit beschäftigt sind, ambige syntaktische Strukturen aufzulösen oder eine von mehreren logischen Repräsentationen auszuwählen, anstatt dass sie sich auf das Schreiben der Anforderungsspezifikation konzentrieren können. Einzelne Ansätze versuchen zwar, strukturelle Mehrdeutigkeiten in der Spezifikationssprache

dadurch zu beseitigen, dass sie die syntaktischen Strukturen vor der Verarbeitung normalisieren. Sie tun das aber nicht sehr systematisch und machen die Regeln des zugrundeliegenden Grammatikmodells nicht explizit.

Das Hauptproblem bei den diskutierten Ansätzen besteht darin, dass der Systembenutzer das Modell des verarbeitbaren Sprachumfangs oft nicht kennt und deshalb die formalen Mittel der Grammatik für die Herstellung des Textzusammenhanges nicht explizit nutzen kann. Das Resultat sind mehrdeutige und unterspezifizierte natürlichsprachliche Spezifikationstexte. Die folgende Liste gibt einen Überblick über die linguistischen Schwachstellen, auf die wir in den besprochenen Ansätzen gestossen sind:

- **Lexikalische Ebene**
  Das zulässige Vokabular ist nicht bekannt.
  Die verwendeten Synonyme und Abkürzungen sind nicht definiert.

- **Syntaktische Ebene**
  Die nominalen Ausdrücke werden ohne Determinatoren verwendet.
  Das zugrundeliegende logische Subjekt fehlt in den Passivkonstruktionen.
  Die Verwendung von Partizipialkonstruktionen führt zu Unterspezifikation.
  Die Anbindung von Präpositionalphrasen ist nicht eindeutig.
  Die zulässigen Satzbaupläne sind nicht bekannt.
  Die Teilsatzbeziehungen sind unterspezifiziert.
  Die Sätze sind für eine effiziente maschinelle Verarbeitung zu lang.

- **Semantische Ebene**
  Der Skopus der Quantoren und die Bindung der Konjunktionen ist mehrdeutig.
  Der Skopus von konditionalen Nebensätzen ist nicht klar.
  Die Interpretation von Modalverben ist unklar.
  Die temporale Abfolge der Ereignisse und Zustände ist unterspezifiziert.

- **Pragmatische Ebene**
  Der Bezug von anaphorischen Verweisformen ist unklar.
  Die elliptischen Konstruktionen verlangen kontextuelle Information.

Um diese Schwachstellen zu beheben, schlagen wir die Verwendung einer kontrollierten natürlichen Spezifikationssprache vor, die unabhängig vom Anwendungsbereich ist und besser auf die Bedürfnisse bei der Anforderungsspezifikation abgestimmt ist.

4.4 Kontrollierte natürliche Sprachen als Spezifikationssprachen

Im Gegensatz zu den verschiedenen Ansätzen in der Industrie, wo kontrollierte natürliche Sprachen für die Dokumentation und die maschinelle Übersetzung eingesetzt werden, sind bisher kaum Versuche unternommen worden, explizit kontrollierte natürliche Sprachen für die Anforderungsspezifikation nutzbar zu machen. Eine Ausnahme bildet die Arbeit von [Macias & Pulman 93, Pulman & Rayner 94, Macias & Pulman 95], in der eine kontrollierte natürlich-sprachliche Schnittstelle zu einem universellen Sprachverarbeitungssystem (CLARE/CLE) vorgestellt wird. Wir diskutieren diesen Ansatz in der Folge eingehender und berücksichtigen wiederum dieselben Punkte wie bei der Untersuchung von Subsprachen als Spezifikationssprachen.

4.4.1 CLARE

CLARE (Contextual Reasoning and Cooperative Response Framework for the Core Language Engine) ist ein universelles natürlich-sprachliches Sprachverarbeitungswerkzeug, das linguistische und beweistheoretische Verfahren miteinander kombiniert [Alshawi et al. 92, Pulman et al. 93]. Den Kern von CLARE bildet die Core Language Engine (CLE), ein anwendungsunabhängiger Sprachprozessor für die Verarbeitung von Texten der englischen Sprache [Alshawi 92]. CLARE ist am SRI International in Cambridge entwickelt worden und umfasst neben der CLE eine Menge von Werkzeugen für die Entwicklung von interaktiven natürlich-sprachlichen Anwendungen. CLARE wurde vor allem für die Entwicklung von Datenbank-Abfragesystemen, von Dialogsystemen und von maschinellen Übersetzungssystemen verwendet.

Interessant in unserem Zusammenhang ist die experimentelle Schnittstelle zu CLARE /CLE, die für das Schreiben von kontrollierten natürlich-sprachlichen Spezifikationstexten entwickelt wurde. Obwohl als Anwendungsbereich vor allem sicherheitskritische Systeme ins Auge gefasst wurden, ist die entwickelte Schnittstelle nicht für einen speziellen Anwendungsbereich ausgelegt.

Die Schnittstelle kontrolliert den Schreibprozess durch die Vorgabe einer kleinen Menge von Satzschemata, die festlegen, wie Teilsätze miteinander verknüpft werden können. Der Ansatz schränkt die zulässige Spezifikationssprache auf eine Teilmenge der englischen Sprache ein, so dass sich bestimmte Formen von struktureller Mehrdeutigkeit durch wohldefinierte Strukturierung ausschliessen lassen. CLARE übersetzt die Sätze in eine logische Form, die den Ausgangspunkt für den formalen Nachweis von Sicherheitsaspekten und für die Überprüfung der Systemzuverlässigkeit bilden.

Der Entwicklung der Schnittstelle gingen linguistische Untersuchungen von bestehenden SASen voraus [Pulman 94, Macias & Pulman 95]. Diese Untersuchungen haben gezeigt, dass natürlich-sprachliche SASen eine Reihe von Nachteilen für die effiziente Verarbeitung und die eindeutige Repräsentation der textuellen Information haben. Erstens, und wenig überraschend, wiesen die untersuchten Spezifikationstexte eine grosse Menge von Mehrdeutigkeiten auf allen linguistischen Ebenen (Morphologie, Syntax, Semantik und Pragmatik) auf. Zweitens hat sich herausgestellt, dass die durchschnittliche Satzlänge in den Spezifikationstexten sehr gross ist. Sie beträgt ungefähr 25-30 Wörter. Die automatische und effiziente Verarbeitung von solch langen Sätzen ist beim gegenwärtigen Stand der Technik nicht möglich. Drittens hat sich gezeigt, dass die Satz- und Sonderzeichen in den Texten sehr uneinheitlich gebraucht werden. Viertens liess sich nachweisen, dass die Autoren in den Texten nur eine kleine Menge wiederkehrender Satzschemata verwenden, um ihre Information darzustellen. Unter den häufigsten Satzschemata fanden sich Satzgefüge mit Verhältnissätzen, wie zum Beispiel konditionale Teilsatzbeziehungen (if <condition clause> then <action clause>) und temporale Teilsatzbeziehungen (before <condition clause>, <action clause>). Ferner bildeten Listen eine andere häufig verwendete Darstellungsform. So können Listen 30-40% eines Spezifikationstextes ausmachen und sind Quellen für Unterspezifikation. Hierzu ein Beispiel aus [Macias & Pulman 95:311]:

(59) *The wash sequence initiation procedure will:*
a. Co-ordinate the filter wash requests.
b. Formulate a wash queue if necessary.
c. Start the wash sequence.

Bei dieser Darstellungsform fehlen die Konnektoren zwischen den einzelnen Sätzen. Es lässt sich nicht feststellen, ob der Agent (*wash sequence initiation procedure*) die einzelnen Punkte (a-c) der Liste deterministisch oder nichtdeterministisch ausführt.

Um strukturelle Mehrdeutigkeiten, übergrosse Satzlängen und unregelmässige Satzzeichensetzung besser in den Griff zu bekommen, unterstützt die Schnittstelle zu CLARE/CLE den Autor beim Schreiben der SAS durch eine menügesteuerte Metagrammatik. Die Metagrammatik gibt die zulässigen Satzschemata vor, was zu einer besseren Strukturierung der grammatischen Verhältnisse zwischen den Teilsätzen führt. Mit Hilfe der Satzschemata können einfache Teilsätze zu komplexen Sätzen zusammengebaut werden. Der Autor kann aus den folgenden Satzschemata auswählen:

- BEFORE Statement
- AFTER Statement
- WHEN Statement
- IF Statement
- LET Statement
- Simple Statement

Der Autor wählt ein Satzschema aus und kann dann je nachdem entweder rekursiv andere Satzschemata aufrufen oder einfache Sätze (*Simple Statement*) eingeben. Wählt der Autor beispielsweise das *BEFORE Statement* aus, dann werden zwei weitere Menüpunkte verfügbar:

- BEFORE
  Type a Statement (<condition clause>)
  Type a Statement (<action clause>)

Der Benutzer wählt den ersten Menüpunkt an und gibt den ersten Teilsatz (<condition clause>) ein:

(60) *The transfer procedure starts.*

Der erste Teilsatz (60) wird von CLARE sogleich verarbeitet. CLARE prüft, ob der Teilsatz grammatisch korrekt ist und erzeugt eine (oder mehrere) logische Formen. Findet CLARE bei der Analyse mehr als eine logische Form, dann muss der Autor den Teilsatz disambiguieren. Die logische Analyse wird zusammen mit dem eingegebenen Teilsatz abgespeichert. Erst nachdem der erste Teilsatz vollständig verarbeitet worden ist, kann der Autor den zweiten Menüpunkt anwählen und den zweiten Teilsatz (<action clause>) eingeben:

(61) *The operator must define the loading process.*

Dieser zweite Teilsatz (61) wird in der gleichen Weise verarbeitet wie der erste Teilsatz. Als Ergebnis entsteht das vollständige Satzgefüge (62), das einen temporalen Nebensatz und einen Hauptsatz enthält:

(62) *BEFORE*
the transfer procedure starts,
the operator must define the loading process.

Listen werden durch eine Erweiterung des menübasierten Ansatzes behandelt. Sollen

Sätze in Listenform aufgeführt werden, dann muss der Autor die Sätze durch vorgegebene Schlüsselwörter (Konjunktionen) koordinieren. Die Schlüsselwörter sind über ein Menü verfügbar. Zum Beispiel wird das Schlüsselwort *AND THEN* als sequentielle Konjunktion interpretiert, die die Reihenfolge der Sätze in einer Liste determiniert (vgl. auch (64)).

Pronomen und definite Nominalphrasen führen oft zu Unsicherheiten bei der Referenzauflösung. In CLARE wird dieses Problem dadurch gelöst, dass globale Namen im Deklarationsteil (63) der Spezifikation eingeführt werden können. Der Autor definiert einen globalen Namen mit Hilfe des *LET Statements* und setzt den Namen mit einer Nominalphrase in Beziehung. Beim Schreiben des Spezifikationstextes (64) können die Namen anstelle der Nominalphrasen verwendet werden.

(63) *LET V be the input valve.*
LET P be the pump attached to V.
LET OP be the operator.

(64) *OP executes the starting procedure WHEN*
OP opens V
AND THEN
OP activates P.

Obwohl diese Technik funktioniert, führt die Umsetzung zu einer nicht sehr natürlichen Darstellung. Die *LET Statements* gleichen Definitionen im Deklarationsteil einer prozeduralen Programmiersprache. Eine natürlichere Alternative führt die globalen Namen als Appositionen von nominalen Ausdrücken ein. Die Namen können später im Text (65) anstelle der Nominalphrasen gebraucht werden:

(65) *An operator, OP, executes the starting procedure WHEN*
OP opens the input valve, V,
AND THEN
OP activates the pump, P.

Wir haben erwähnt, dass der Autor ambige Sätze interaktiv disambiguieren muss. Wenn CLARE einen Satz verarbeitet, der zu mehr als einer logischen Analyse führt, dann wird für jede Lesart des Satzes eine Paraphrase erzeugt. Der Satz

(66) *The operator has stopped the process on the menu.*

mit der Präpositionalphrase *on the menu* führt zu zwei möglichen logischen Analysen.

Entweder wird das Nomen (*process*) oder das Verb (*stop*) durch die Präpositionalphrase modifiziert. CLARE bildet für den Satz (66) zwei Paraphrasen und macht die Information für den Autor in einer kontrollierten Sprache zugänglich, dem sogenannten Logician's English [Macias & Pulman 95:315]:

(67) *There is some operator OP, some process PRO (such that there is a menu ME, and PRO is on ME), and a past event E, such that E is an event of OP stopping PRO.*

(68) *There is some menu ME, some operator OP, some process PRO, and some past event E, such that E is an event of OP stopping PRO, and E is on ME.*

Der Autor wählt diejenige Paraphrase aus, die seiner intendierten Interpretation entspricht. Theoretisch ist es möglich, Sätze in voller natürlicher Sprache aus den logischen Repräsentationen zu generieren, da es sich bei CLARE um ein bidirektionales System handelt. Es gibt jedoch keine Garantie dafür, dass die Paraphrasen in voller natürlicher Sprache nicht wiederum mehrdeutige Strukturen einführen.

4.4.2 Schwachstellen von kontrollierten natürlichen Sprachen als Spezifikationssprachen

Die kontrollierte natürlich-sprachliche Schnittstelle zu CLARE/CLE bringt einige entscheidende Vorteile im Vergleich zu den subsprachlichen Ansätzen. Die Verwendung der menübasierten Metagrammatik mit den expliziten Satzschemata führt dazu, dass

- weniger strukturelle Ambiguitäten und Unterspezifikationen entstehen,
- deutlich kürzere Sätze gebildet werden,
- die Anzahl der Satzzeichen minimiert wird.

Ausserdem steht ein mächtiger Mechanismus zur Verfügung, der es erlaubt, gewisse Formen von anaphorischer Referenz durch eindeutige Rekurrenz von Namen aufzulösen, die durch *LET Statements* oder Appositionen eingeführt werden können.

Trotzdem hat auch diese kontrollierte Schnittstelle zu CLARE/CLE einige Schwachstellen, die vor allem darauf zurückzuführen sind, dass die zugrundeliegende Sprache nicht auf eine konstruktive Art und Weise entworfen worden ist. Die Schnittstelle setzt auf einem Sprachprozessor auf, der für die Analyse von voller natürlicher Sprache entwickelt worden ist. Für das Schreiben einfacher Sätze und den Aufbau des zulässigen Vokabulars sind überhaupt keine Einschränkungen getroffen worden. Streng

genommen bleibt das Konzept der kontrollierten natürlichen Sprache auf halbem Weg stecken. Offen bleiben eine Reihe von Fragen, die die linguistische Adäquatheit des Ansatzes betreffen:

- Welche lexikalischen Einschränkungen sind nötig?
- Welche phrasalen Einschränkungen braucht es?
- Welche Satzbaupläne für einfache Sätze sind festzulegen?
- Wie werden Modalverben interpretiert?
- Wie werden Ellipsen behandelt?
- Sind die Paraphrasen reversibel?

Ausserdem stellen sich folgende Fragen, die die Entwicklung, Validierung und Ausführung der Spezifikation betreffen:

- Wie baut der Anwendungsspezialist das fachspezifische Vokabular auf?
- Wie kann der Anwendungsspezialist die Spezifikation validieren?
- Ist die Spezifikation symbolisch ausführbar?

Wir haben nun verschiedene Ansätze untersucht, wo Subsprachen und in einem Fall eine kontrollierte natürliche Sprache für die Anforderungsspezifikation eingesetzt werden. Die aus diesen Untersuchungen gewonnenen Erkenntnisse und die dabei fixierten Schwachstellen bilden zusammen mit den bekannten kontrollierten natürlichen Sprachen aus der Industrie den Ausgangspunkt für die Entwicklung einer eigenen universellen kontrollierten natürlichen Sprache für die Spezifikation von faktischen Anforderungen.

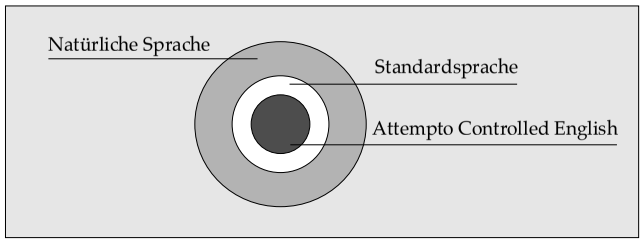
5. Die Sprache Attempto Controlled English (ACE)

Mit ACE bieten wir dem Anwendungsspezialisten eine Spezifikationssprache an, die einerseits die konzeptionelle Lücke zwischen informellen und formalen Spezifikationsmethoden schliesst und andererseits die Schwachstellen von bisherigen natürlich-sprachlichen Spezifikationssprachen behebt. Obwohl die Sprache ACE im Prinzip auch unabhängig von einem Computersystem zur Spezifikation von Anforderungen auf Papier gebraucht werden kann, entfaltet die Sprache ihr volles Potential erst durch die Einbettung in das Attempto Spezifikationssystem. ACE Spezifikationen können hier eindeutig paraphrasiert, repräsentiert, validiert und sogar ausgeführt werden.

5.1 Die Philosophie von ACE

ACE ist eine kontrollierte natürliche Sprache, die für das Schreiben von faktischen Anforderungen ausgelegt ist. ACE besteht aus drei Komponenten: Aus einem Vokabular mit anwendungsspezifischen Inhaltswörtern und vordefinierten Funktionswörtern, aus einer präzis definierten Grammatik, die eine Teilmenge der englischen Standardsprache umfasst, und aus einer Menge von linguistischen Schreibprinzipien, die den Gebrauch der Sprache durch den Anwendungsspezialisten regeln. Die Sprache ACE ist sachverhaltsorientiert und methodenneutral. Einfache Aussagesätze in ACE stellen Sachverhalte dar und bilden die kleinsten Einheiten, die noch Orientierung bieten, indem sie Entitäten zueinander in Beziehung setzen [vgl. Braun et al. 96:286]. ACE ist keiner bestimmten Software-Entwicklungsmethode verpflichtet, da die Ergebnisse methodenneutral erarbeitet werden [vgl. Ortner & Schienmann 96:113]. ACE ist mächtig genug, um Systeme zu spezifizieren, die durch endliche Automaten modelliert werden können. Die Sprache ACE kann effizient durch einen Computer verarbeitet und eindeutig in eine ausführbare logische Sprache übersetzt werden.

Attempto Controlled English



ACE ist prinzipienbasiert und besteht - im Gegensatz zu den meisten Subsprachen und anderen kontrollierten natürlichen Sprachen - aus einer echten Teilmenge der englischen Standardsprache.

Software-Anforderungsspezifikationen in ACE sind textuelle Sichten auf formale Spezifikationen in Logik. Die textuellen Sichten erscheinen für den Anwendungsspezialisten informell, obwohl sie tatsächlich formal und maschinell verarbeitbar sind. Zwischen einer textuellen Sicht in ACE und einer formalen Spezifikation in Logik besteht eine Korrespondenz, die durch eine eindeutige Übersetzung der Sicht zustande kommt. Die Übersetzung weist der textuellen Sicht die Semantik der zugrundeliegenden formalen Sprache zu. Die Korrespondenz zwischen der textuellen Sicht und der formalen Spezifikation verbindet die gute Lesbarkeit einer (kontrollierten) natürlichen Sprache mit der Präzision einer formalen Sprache.

Die Anwendungsspezialisten schreiben den Spezifikationstext auf der Sichtebene, indem sie wahre oder hypothetische Aussagen über einen Diskursbereich in ACE machen und dazu ihr fachspezifisches (d.h. inhaltsbezogenes) Vokabular aus dem Anwendungsbereich verwenden können. Durch diese Vorgehensweise können die Anwendungsspezialisten unmittelbare, sachverhaltsorientierte Aussagen über die wahrgenommenen Phänomene machen, ohne dass sie die fachspezifischen Begriffe erst in einer formalen Sprache kodieren müssen. Die Anwendungsspezialisten bauen das inhaltsbezogene Vokabular während des Spezifikationsprozesses selber auf. Ausser den erwähnten linguistischen Prinzipien verlangt die Verwendung von ACE kein Wissen über formale Methoden oder computerlinguistische Verfahrensweisen.

Die Sprache ACE ist in das Attempto Spezifikationssystem eingebettet, das die natürlich-sprachlichen Spezifikationstexte liest und eindeutig in Diskursrepräsentationsstrukturen - eine strukturierte Form von Prädikatenlogik erster Stufe - übersetzt. Die grammatische Analyse ist deterministisch und gleicht in dieser Hinsicht der Analyse einer Programmiersprache. Der Chart-Parser des Attempto Systems erzeugt bei der Analyse einen Syntaxbaum als syntaktische Repräsentation, eine Diskursrepräsentationsstruktur als formale Repräsentation und eine Paraphrase. Die Steuerung des Analyseprozesses wird durch Merkmalsunifikation erreicht. Die resultierenden Diskursrepräsentationsstrukturen basieren auf einer um Ereignisse und Zustände erweiterten Diskursrepräsentationstheorie (DRT-E). Die Diskursrepräsentationsstrukturen können entweder direkt durch einen Metainterpreter oder nach einer Übersetzung in Hornklauseln durch den Prologinterpreter verarbeitet werden. Um das

Resultat der Analyse dem Anwendungsspezialisten zugänglich zu machen, erzeugt der Chart-Parser während der Verarbeitung der Spezifikation eine Paraphrase in ACE, die alle Substitutionen und Interpretationen verdeutlicht. Wenn der Anwendungsspezialist mit der vorgeschlagenen Interpretation nicht einverstanden ist, dann muss er den Satz gemäss den Schreibprinzipien von ACE neu formulieren. Für Validierungszwecke kann der Anwendungsspezialist die Spezifikation in ACE befragen oder symbolisch ausführen. Fragen werden in sogenannte Frage-Diskursrepräsentationsstrukturen übersetzt, für die anschliessend Lösungen durch Inferenz in der vorliegenden Wissensbasis gesucht werden. Aufgrund der berechneten Lösungen werden Antworten in der Sprache ACE erzeugt. Die symbolische Ausführung der Spezifikation erlaubt es dem Anwendungsspezialisten, das Verhalten der Spezifikation in ACE zu beobachten und zu überprüfen. Die Ausführbarkeit der Spezifikation verlangt zusätzliche situationsspezifische Information, die der Anwendungsspezialist - oder allenfalls der Softwareentwickler - mit Hilfe der Sprache ACE zur Verfügung stellen kann.

5.2 Linguistische Grundlagen für ACE

ACE ist eine Spezifikationssprache, die sich an Anwendungsspezialisten ohne linguistische Spezialausbildung richtet. Allein die linguistischen Schreibprinzipien, auf denen ACE aufbaut, bilden das Rüstzeug für den korrekten Sprachgebrauch. Diese Prinzipien sind normativ-präskriptiver Art und müssen für den Anwendungsspezialisten leicht erlernbar und erinnerbar sein. Nun ist es aber so, dass die Begründung dieser linguistischen Prinzipien und die Entwicklung der Sprache ACE eine theoretische Abstützung erfordern, die besser durch eine deskriptive Grammatik geleistet werden kann. Da der interessierte Anwendungsspezialist vielleicht mehr über die linguistischen Grundlagen der von ihm verwendeten Sprache erfahren möchte, werden zuerst einige Grundbegriffe erklärt, die bei der Entwicklung von ACE eine Rolle spielen. Es wird gezeigt, welche verschiedenen Satzarten und Satzformen in ACE zu berücksichtigen sind und nach welchen Kriterien sie klassifiziert werden. Für die Beschreibung des Satzbaus wird eine Konstituentenanalyse vorgeschlagen, die die versteckte hierarchische Struktur von Sätzen klar macht. Anforderungsspezifikationen bestehen nicht aus einzelnen Sätzen, sondern aus einer Summe von Sätzen, die sich zu Paragraphen und schliesslich zu einem ganzen Text verbinden. Die Beziehung zwischen den einzelnen Sätzen lässt sich oft an sprachlichen Kohäsionsmitteln (z.B. Rekurrenz von Elementen, Substitution von Elementen und Verknüpfung von Elementen) ablesen. Mit Hilfe der Diskursrepräsentationstheorie (DRT) als logischer Zielsprache

für ACE ist es möglich, einige dieser textuellen Bezüge systematisch zu behandeln. Es wird eine Erweiterung der klassischen DRT vorgestellt, bei der Ereignisse und Zustände - sogenannte Eventualitäten (E) - reifiziert werden. Verben können Ereignisse oder Zustände im Anwendungsbereich denotieren und werden aufgrund dieser Eigenschaften klassifiziert. Diese erweiterte Theorie heisst DRT-E und ist speziell auf die adäquate Verarbeitung von ACE Spezifikationen abgestimmt. Die DRT-E ermöglicht eine systematische Übersetzung von untereinander verbundenen ACE Sätzen in eine strukturierte Form der Prädikatenlogik erster Stufe, wobei speziell berücksichtigt wird, wie die Anwendungsspezialisten das inhaltsbezogene Vokabular klassifizieren.

5.2.1 Terminologische Vorbemerkungen

Dieses Kapitel richtet sich an Anwendungsspezialisten, die keine linguistischen Vorkenntnisse haben, und führt in die Terminologie ein, die für das Verständnis der sprachtheoretischen Grundlagen von ACE Voraussetzung sind. Linguisten können dieses Kapitel überspringen.

Satzarten

ALLGEMEIN. Als *Satzarten* werden feste wiederkehrende sprachliche Muster bezeichnet, die sich sich mit Hilfe von formalen Kriterien (z.B. Verbmodus, Verbstellung, Einleitewörter, Satzschlusszeichen) bestimmen lassen. Für Anforderungsspezifikationen macht es Sinn, formal drei Satzarten zu unterscheiden: *Aussagesätze*, *konditionale Sätze* und *Fragesätze*. Aufgrund von formalen und inhaltlichen Kriterien lassen sich dann Zusammenhänge zwischen Satzarten und Diskursfunktionen herstellen. So können den drei Satzarten die Diskursfunktionen *faktische Aussagen machen, bedingte Aussagen machen* und *Fragen stellen* zugeordnet werden.

Für die Bestimmung der Satzarten spielen die Verben bzw. die Verbformen eine wichtige Rolle. An den Verbformen sind unterschiedliche Merkmale (Person, Numerus, Tempus, Modus und Genus Verbi (= Aktiv/Passiv)) ablesbar. Als finite Verbformen bezeichnet man Verbformen, die nach Person und Numerus bestimmt sind. Mit dem Indikativ als Verbmodus zeigt man eine bestimmte Aussageweise an. Der Indikativ wird als Normalmodus aufgefasst, da er einen Sachverhalt als real (oder allenfalls als real möglich) hinstellt. Die finiten Verbformen sind in der englischen Sprache an bestimmte Positionen gebunden. Von der Normalstellung wird gesprochen, wenn die finite Verbform dem Subjekt folgt.

Aussagesätze wie (69) sind die formal am wenigsten markierte Satzart. Die finite Verb-

form steht normalerweise im Indikativ und folgt dem Subjekt.

(69) *The staff user adds a copy of a book to the library.*

Inhaltlich beschreiben Aussagesätze Sachverhalte, indem sie über Entitäten in einem Anwendungsbereich, auf die man sprachlich Bezug nimmt, Aussagen machen. In der Sprechakttheorie werden einem Aussagesatz zwei Aspekte zugeschrieben: erstens eine Proposition (= Aussagegehalt) und zweitens eine Illokution (= Geltungsanspruch). Auf der einen Seite wird mit einem Aussagesatz gesagt, wie sich eine Sache im Anwendungsbereich verhält, wenn der Satz wahr ist. Auf der anderen Seite wird mit einem Aussagesatz behauptet, dass es sich so verhält, wie es von ihm ausgesagt wird. Aussagesätze sind *repräsentative* Sprechakte, bei denen ein Sprecher die Wahrheit der ausgedrückten Propositionen geltend macht [vgl. Searle 69].

Konditionale Sätze setzen sich aus mindestens einem konditionalen Nebensatz und mindestens einem bedingten Hauptsatz zusammen und bilden ein konditionales Satzgefüge. In der Linguistik werden konditionale Nebensätze allein oft als *Konditionalsätze* bezeichnet. Um Missverständnisse auszuschliessen, wird in der Folge mit dem Ausdruck *konditionaler Satz* bzw. *konditionales Satzgefüge* immer ein zusammengesetzter Satz der Form *IF P1 THEN P2* bezeichnet. Das konditionale Verhältnis zwischen den Teilsätzen wird in Satz (70) beispielsweise durch die nebensatzeinleitende Konjunktion *if* und das den Hauptsatz markierende Adverb *then* angezeigt.

(70) *If a staff user returns a copy of a book then the copy is available.*

Inhaltlich dienen konditionale Sätzgefüge dazu, um in einem Spezifikationstext bedingte Aussagen bzw. Regelaussagen zu machen. Aus einem vorausgesetzten (bzw. als Regel gewussten) Sachverhalt *P1*, der im Nebensatz ausgesagt wird und als Bedingung gekennzeichnet ist, wird auf einen bedingten Sachverhalt *P2* im Hauptsatz geschlossen. *P2* ist genau dann wahr, wenn alle Bedingungen, die in *P1* vorausgesetzt werden, wahr sind. Es sind hier nur konditionale Nebensätze von Interesse, bei denen der vorausgesetzte Sachverhalt real möglich ist. Die finite Verbform im Nebensatz steht deshalb immer im Indikativ.

Fragesätze treten hauptsächlich als Entscheidungsfragen oder Ergänzungsfragen auf und werden durch ein Fragezeichen abgeschlossen. Entscheidungsfragen (= Satzfragen) entstehen durch Subjekt-Hilfsverb-Inversion wie in (71) und - wo nötig - durch *do*-Einsetzung wie in (72).

(71) *Is the copy available?*

(72) *Does the staff user return the copy?*

Inhaltlich geht es bei Entscheidungsfragen darum, festzustellen, ob ein Sachverhalt, der durch eine Frage zum Ausdruck gebracht wird, zutrifft oder nicht. Auf Entscheidungsfragen kann mit *yes* oder *no* geantwortet werden.

Ergänzungsfragen (= Wortfragen) werden durch die Verwendung von Frageausdrücken gebildet. Die Frageausdrücke können unterschiedlichen Wortarten angehören. Entweder werden zur Bildung von Ergänzungsfragen Fragepronomen (z.B. *who*) wie in (73), Fragedeterminatoren (z.B. *which*) wie in (74) oder Frageadverbien (z.B. *where*) wie in (75) verwendet. Ausserdem verlangt die Bildung des Fragesatzes in (75) eine Anhebung des Frageausdrucks an den Satzanfang, *do*-Einsetzung und Subjekt-Operator-Inversion.

(73) *Who returns the copy?*

(74) *Which copy carries a number?*

(75) *Where does the staff user add the copy to?*

Inhaltlich steht bei Ergänzungsfragen nicht der gesamte im Satz ausgedrückte Sachverhalt zur Debatte, sondern es wird nur ein bestimmter Aspekt des Sachverhaltes erfragt, und zwar derjenige, der mit dem Frageausdruck assoziiert wird.

ATTEMPTO CONTROLLED ENGLISH (ACE). Bei der Beschreibung der Sprache ACE werden wir drei Satzarten unterscheiden: Aussagesätze, konditionale Sätze und Fragesätze (Entscheidungs- und Ergänzungsfragen). Aussagesätze werden in ACE dazu gebraucht, um faktische Sachverhalte aus einem Anwendungsbereich festzuhalten, d.h. um faktische Aussagen zu machen. Konditionale Sätze werden dazu verwendet, um diejenigen Vorbedingungen festzusetzen, unter denen ein bezeichneter Sachverhalt zutreffen kann, d.h. um bedingte Aussagen zu machen. Fragesätze werden zu Validierungszwecken gebraucht, d.h. um Fragen an eine übersetzte Spezifikation zu stellen.

Die Verbformen in diesen ACE Sätzen sind nur in ganz bestimmten Ausprägungen zulässig, und zwar sind nur finite Formen erlaubt, die in der 3. Person Singular (oder Plural) stehen, die das einfache Präsens (= *simple present tense*) als Tempus haben, den Indikativ als Modus und das Aktiv als Genus Verbi (= Verbalgenus).

Satzformen

ALLGEMEIN. Von der Satzform her lassen sich in der englischen Sprache *einfache* und *zusammengesetzte* Sätze unterscheiden. Einfachen Sätzen liegt eine finite Verbform zugrunde. Vom Verb hängt hauptsächlich ab, wie die syntaktische Umgebung im Satz vorstrukturiert ist und welche Elemente realisiert sein müssen. Ein Beispiel für einen einfachen Satz ist:

(76) *The staff user checks out a copy.*

Zusammengesetzte Sätze sind komplexe Konstruktionen, die aus Teilsätzen bestehen, zwischen denen unterschiedliche grammatische Abhängigkeiten vorliegen. Wenn ein zusammengesetzter Satz aus selbständigen Teilsätzen, d.h. Hauptsätzen, besteht, die koordiniert (= nebengeordnet) sind, handelt es sich um eine *Satzverbindung*. Beispielsweise werden die beiden Hauptsätze in

(77) *The staff user adds a copy to the library and the staff user creates a catalogue entry.*

durch die koordinierende Konjunktion *and* verbunden. Die Hauptsätze in (77) können ohne merkliche Veränderung der Aussage auch als einfache Sätze für sich allein stehen. Wenn ein zusammengesetzter Satz aus (mindestens) einem Teilsatz, d.h. einem Nebensatz, besteht, der einem anderen Teilsatz grammatisch subordiniert (= untergeordnet) ist, dann handelt es sich um ein *Satzgefüge*. Das folgende Satzgefüge

(78) *The staff user who adds a copy to the library creates a catalogue entry.*

besteht aus einem Hauptsatz, in den ein Nebensatz eingebettet ist. Der Nebensatz *who adds a copy to the library* wird durch das subordinierende Relativpronomen *who* eingeleitet.

ATTEMPTO CONTROLLED ENGLISH (ACE). Bei der Beschreibung von ACE werden einfache und zusammengesetzte Sätze unterschieden, wobei zusammengesetzte Sätze auf eine konstruktive Art und Weise aus einfachen Sätzen durch die Verwendung von Koordinatoren und Subordinatoren gebildet werden können. Dort wo Teilsätze durch Koordinatoren verbunden werden, werden wir in ACE von Satzverbindungen sprechen und dort wo Subordinatoren Teilsätze verfugen von Satzgefügen.

Satzbau

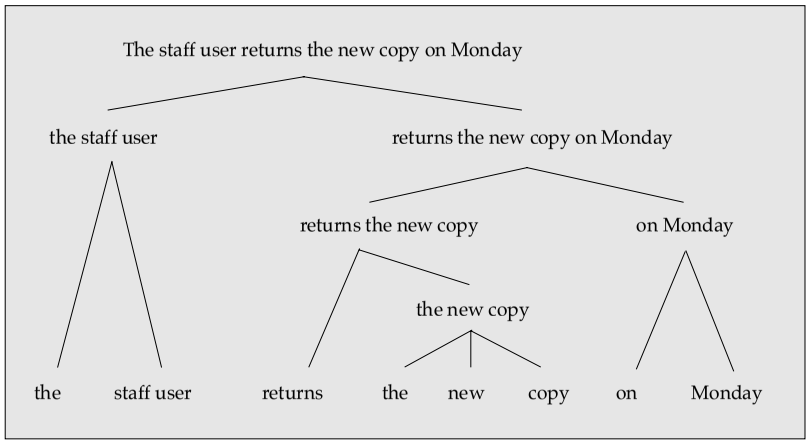
ALLGEMEIN. Natürlich-sprachliche Sätze bestehen nicht bloss aus linearen Sequenzen von Wörtern, sondern hinter ihrer Linearität verbirgt sich eine hierarchische Struktur. Der Bau eines Satzes kann als eine hierarchisch definierte Abfolge von Konstituenten

beschrieben werden. Konstituenten sind Wörter sowie Folgen von Wörtern, die als linguistische Einheiten Teile einer hierarchisch höheren linguistischen Einheit bilden. Zur Darstellung der Konstituentenstruktur verwendet man Baumdiagramme. Der Satz

(79) *The staff user returns the new copy on Monday.*

hat folgende Konstituentenstruktur:

Baumdiagramm



Die Darstellung zeigt, wie sich ein Satz in seine Konstituenten gliedert, und verdeutlicht, wie die Wörter als minimale Konstituenten zuunterst im Strukturbaum erst über mittelbare Konstruktionen bzw. Phrasen mit der Ebene des Satzes verbunden sind. Phrasen sind Satzteile von relativer Selbständigkeit, die im Gegensatz zu minimalen Konstituenten syntaktisch direkt verwendungsfähig sind. Für jedes Wort wird angenommen, dass es zu einer Wortklasse gehört und zusammen mit einer Menge von syntaktischen und semantischen Merkmalen in einem Lexikon verzeichnet ist. Die Klassenzugehörigkeit und die Merkmalstruktur ist entscheidend dafür, wie ein Wort in eine Phrase eingebaut werden kann. Beispielsweise werden *staff user* und *copy* als Nomen klassifiziert, weil sie Merkmale haben, die für diese Wortklasse charakteristisch sind. Ähnlich werden aufgrund ihrer Merkmale *the* als Determinator, *returns* als Verb, *new* als Adjektiv, *on* als Präposition, und *Monday* als Eigennamen klassifiziert. Einzelne obligatorische Wörter - die als sogenannte lexikalische Köpfe funktionieren - verlangen oft eine bestimmte Anzahl anderer Konstituenten mit ganz spezifischen

Merkmalen um sich herum, bevor sie als Phrasen verwendungsfähig werden. Die Eigenschaft von Wörtern, andere Konstituenten einzufordern, bezeichnet man als Selektion und spricht in diesem Zusammenhang vom Selektionsrahmen dieser Wörter, der im Lexikon mitverzeichnet ist. Schauen wir das Baumdiagramm etwas genauer an.

Das Nomen *staff user* ist allein nicht als Phrase verwendungsfähig, sondern es muss ergänzt werden. Diese Funktion übernimmt ein sogenannter Spezifikator in Form eines Determinators (*the*), der eine Konstituente abschliessen kann. Zusammen bilden der Determinator und das Nomen die Nominalphrase *the staff user*, wobei die Phrase die kategoriale Prägung durch den nominalen Kopf erhält.

Das Verb *returns* kann erst eine verbale Konstituente bilden, nachdem es aufgrund seines Selektionsrahmens ein (obligatorisches) Komplement in Form einer Nominalphrase (*the new copy*) selektioniert hat. Die selektionierte Nominalphrase enthält ein Adjektiv (*new*) - bzw. eine Adjektivphrase - als zusätzliches Attribut. Adjektive sind Wörter, die phrasal verwendungsfähig sind, ohne dass sie syntaktisch ergänzt werden müssen.

Die verbale Konstituente *returns the new copy* befindet sich auf einer Zwischenebene und verbindet sich mit der Präpositionalphrase *on Monday* zur Verbalphrase *returns the new copy on Monday* auf der nächsthöheren Ebene. Die Präpositionalphrase *on Monday* wird deshalb auf der Zwischenebene angebunden, weil sie als (optionales) Adjunkt unabhängig von den Selektionseigenschaften des Verbs ist. Die Präpositionalphrase selber besteht aus der Präposition *on*, die ein Komplement in Form einer Nominalphrase (*Monday*) selektioniert. Beim Eigennamen *Monday* handelt es sich wiederum um ein Wort, das zugleich als Phrase syntaktisch verwendungsfähig ist und nicht zuerst ergänzt werden muss.

Neben dem Komplement und dem Adjunkt, die zusammen die internen Ergänzungen der Verbalphrase bilden und gemeinsam als Prädikat des Satzes funktionieren, selektioniert das Verb *returns* noch eine externe Ergänzung, die ausserhalb der Verbalphrase liegt. Diese externe Ergänzung wird durch die Nominalphrase *the staff user* realisiert und funktioniert als Subjekt des Satzes.

Wichtig für den Satzbau ist die Unterscheidung zwischen Komplementen und Adjunkten. Komplemente werden immer durch eine obligatorische Konstituente realisiert und vervollständigen eine grammatische Kategorie. Lässt man eine obligatorische Konstituente weg, so führt das entweder zu einem ungrammatischen Satz oder zu einer massiven Bedeutungsveränderung des Satzes. Das Weglassen der Nominalphrase *the new copy* im Beispielsatz (79) bringt eine Bedeutungsveränderung mit sich, wobei der entstehende Satz (80) nach wie vor grammatisch bleibt.

(80) *The staff user returns on Monday.*

Im Gegensatz zu Komplementen sind Adjunkte immer optional, da ihre formalen Ausprägungen nicht durch den lexikalischen Kopf selektioniert werden. Adjunkte unterscheiden sich von Komplementen auch dadurch, dass sie bezüglich ihrer linearen Abfolge freier sind.

(81) *On Monday, the staff user returns.*

Stellenweise wird in der Literatur zwischen Attributen und Adjunkten unterschieden [vgl. Radford 88:254]. Folgt man dieser Unterscheidung, dann wird der Begriff *Attribut* meistens auf einen optionalen Modifikator bezogen, der vor einem lexikalischen Kopf steht, und der Begriff *Adjunkt* (oder *Apposition*) auf einen optionalen Modifikator, der einem lexikalischen Kopf folgt.

Die vorgestellte Beschreibung des Satzbaus geht davon aus, dass natürliche Sprachen auf einer lexikalischen und einer phrasalen Ebene organisiert sind. Zusätzlich zu dieser formalen Beschreibung, die über grammatische Kategorien (Wortklassen und Phrasen) spricht, sind bei der Diskussion auch die grammatischen Funktionen (Subjekt, Prädikat, Komplement, Adjunkt, Kopf, usw.) miteinbezogen worden, die die Konstituenten im Satz haben. Zwischen der formalen und der funktionalen Beschreibung des Satzbaus bestehen komplexe Beziehungen, die konzeptionell und terminologisch auseinandergehalten werden müssen. Von der Form her gehören beispielsweise die Konstituenten *the staff user* und *the new copy* zu derselben grammatischen Kategorie, d.h. sie bilden Nominalphrasen. Von ihrer Funktion her, d.h. von ihren grammatischen Rollen, die sie im Satz spielen, unterscheiden sich die beiden Konstituenten jedoch erheblich. Die Konstituente *the staff user* funktioniert als Subjekt und die Konstituente *the new copy* als Komplement, oder genauer als direktes Objekt. Die Unterscheidung von Form und Funktion wird unter anderem dann relevant, wenn man die Rolle von Fragesätzen untersucht.

Zwischen grammatischen Funktionen und Ergänzungsfragen bestehen enge Beziehungen, die nicht aufgrund der grammatischen Kategorien erklärt werden können.

Beispielsweise kann das Subjekt des Beispielsatzes (79) durch eine spezifische Ergänzungsfrage wie (82) erfragt werden:

(82) *Who returns the new copy?*

Teile des Prädikats von (79) können auf ähnliche Weise durch Ergänzungsfragen bestimmt werden. So kann das Komplement, das durch die Nominalphrase *the new copy* realisiert wird, durch die folgende Frage ermittelt werden:

(83) *What does the staff user return?*

Sowohl in (82) als auch in (83) handelt es sich formal um Nominalphrasen, nach denen gefragt wird, doch spielen sie funktional völlig unterschiedliche Rollen im Aussagesatz.

Genauso kann das Attribut der Nominalphrase *the new copy* in (79) durch eine Ergänzungsfrage wie (84) erfragt werden:

(84) *Which copy does the staff user return?*

Und schliesslich kann die Information, die an der Adjunktposition von (79) steht, durch eine Ergänzungsfrage wie (85) ermittelt werden, wobei die Form des Fragesatzes

(85) *When does the staff user return the new copy?*

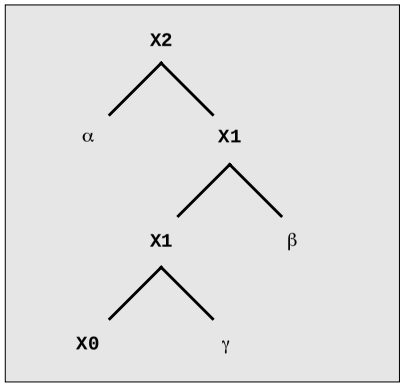
nicht unmittelbar mit der formalen Ausprägung der erfragten Konstituente zusammenhängt. Die Adjunktposition in (79) ist durch die Präpositionalphrase *on Monday* realisiert. Im Prinzip wäre an dieser Stelle aber auch eine Adverbialphrase (z.B. *tomorrow*) denkbar, die durch (85) erfragt werden kann.

Andererseits hat die Untersuchung des Beispielsatzes (79) gezeigt, dass die Struktur eines natürlich-sprachlichen Satzes - allein von der Form her - als eine Hierarchie von geordneten Konstituenten aufgefasst werden kann. Diese Konstituentenstrukturen können durch eine Menge von (kontextfreien) Phrasenstrukturregeln beschrieben werden. Phrasenstrukturregeln sind Ersetzungsregeln für syntaktische Kategorien, die anstelle von Konstituenten stehen. Sie haben die allgemeine Form *C -> A B*. Eine solche Regel ist eine Anweisung, die auf der linken Seite des Pfeils stehende syntaktische Kategorie *C* der Reihe nach durch die auf der rechten Seite stehenden syntaktischen Kategorien *A* und *B* zu ersetzen [vgl. Chomsky 57, Borsley 91].

X-bar-Theorie

In der X-bar-Theorie wird versucht, die Konstituentenstrukturen so weit wie möglich zu verallgemeinern und dahinter ein abstraktes und universelles Strukturierungsprinzip zu sehen, das für alle Sprachen prinzipielle Gültigkeit hat [vgl. Chomsky 70, Jackendoff 77, Radford 88]. Dieses Strukturierungsprinzip wurde im sogenannten X-bar-Schema formuliert, das hier als Baumdiagramm dargestellt ist:

X-bar-Schema



Das X-bar-Schema besagt, dass jede phrasale Kategorie `X2` einen lexikalischen Kopf `X0` hat. Der lexikalische Kopf wird durch obligatorische und optionale Elemente ergänzt. Zwischen dem lexikalischen Kopf `X0` und der phrasalen Kategorie `X2` gibt es möglicherweise mehrere kategoriale Zwischenebenen `X1`. Während γ das Komplement von `X0` ist, bezeichnet β das Adjunkt von `X1` und α den Spezifikator von `X2`. Die phrasalen Kategorien an diesen Positionen (α, β, γ) müssen intern selber einen lexikalischen Kopf enthalten.

Der lexikalische Kopf `X0` bildet zusammen mit seinem Komplement γ die nächsthöhere Kategorie `X1`.

`X1 -> X0` γ

Die Zwischenkategorie `X1` wird eine Projektion des lexikalischen Kopfes genannt, da wichtige syntaktische und semantische Merkmale des lexikalischen Kopfes auf die phrasale Ebene übertragen werden. Ist beispielsweise der lexikalische Kopf ein Nomen

(*n0*) mit dem Merkmal *Singular*, dann zeigt auch die höhere Kategorie *n1* dieses Merkmal.

Im Gegensatz zu einem Komplement verbindet sich ein Adjunkt β nicht mit einer lexikalischen Kategorie *X0*, sondern mit einer Zwischenkategorie *X1*.

X1 -> X1 (β)

Die Anzahl der Adjunkte kann theoretisch beliebig erweitert werden, wobei sich die entstehenden Konstituenten syntaktisch gleich verhalten wie eine Konstituente, die aus dem lexikalischen Kopf und den Komplementen allein besteht. Aus diesem Grund wird die Zwischenkategorie *X1* innerhalb des X-bar-Schemas rekursiv eingeführt. Die runden Klammern zeigen an, dass es sich bei Adjunkten um optionale Elemente handelt.

Die Zwischenkategorie *X1* bildet zusammen mit einem Spezifikator α, z.B. in Form eines Determinators, die nächsthöhere phrasale Kategorie *X2*.

X2 -> α X1

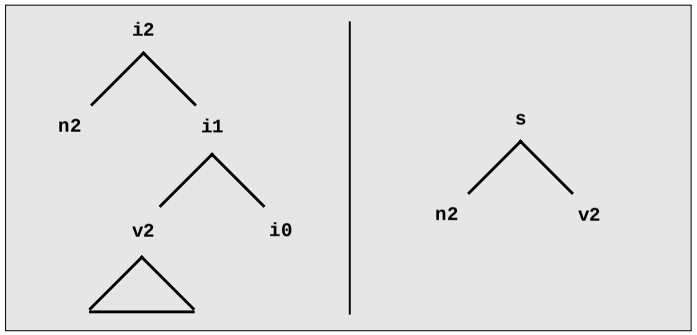
Die entstehende Projektionsebene wird auch maximale Projektion des lexikalischen Kopfes genannt und schliesst eine Phrase ab. Im Unterschied zu Komplementen und Adjunkten, die stets maximale Projektionen, d.h. phrasale Kategorien, sein müssen, kann ein Spezifikator auch eine lexikalische Kategorie sein.

Drei Punkte sind hier wichtig: Erstens lässt das X-bar-Schema offen, ob ein Wort ergänzt werden muss, um als Phrase verwendungsfähig zu sein. Das hängt allein davon ab, wie der Selektionsrahmen des Wortes im Lexikon definiert ist. Zweitens lässt das X-bar-Schema offen, in welcher Richtung die Projektionslinien verzweigen, da diese in der Regel nicht lexikalisch determiniert sind (und einzelsprachlichen Parametrisierungen gehorchen). Beispielsweise steht die Adjektivphrase *new* im Satz (79) links vom nominalen Kopf und die Präpositionalphrase *on Monday* rechts vom verbalen Kopf. Drittens beschränkt das X-bar-Schema bestimmte Phrasenstrukturen implizit; so lässt sich am Schema leicht ablesen, dass beispielsweise eine Phrasenstrukturregel der Form *v2 -> p0 n2* nicht zulässig ist.

Das X-bar-Schema ist ursprünglich nur für die vier lexikalischen Kategorie *Nomen*

(n0), *Verb (v0)*, *Adjektiv (a0)* und *Präposition (p0)* postuliert worden. Später sind im Rahmen der generativen Grammatik auch Sätze, Fragesätze und Nebensätze in das X-bar-Schema integriert worden [vgl. Chomsky 86, Fanselow & Felix 87, Farke 94]. Aus theoretischen Gründen wird für Sätze angenommen, dass eine eigene obligatorische Kategorie existiert, die die Kongruenz von Subjekt und Prädikat - also von Nominalphrase und Verbalphrase - durch die Merkmale für Finitheit, Person und Numerus festlegt. Dieser Merkmalkomplex bildet den funktionalen Kopf *i0* (*i* für engl. *inflection*) eines Satzes, der gemäss dem X-bar-Schema zur maximalen Phrase *i2* (= *s*) projiziert. Für einen Standardsatz ergibt sich somit die Struktur im linken Teil der untenstehenden Abbildung mit der Nominalphrase *n2* als Spezifikator und der Verbalphrase *v2* als Komplement der funktionalen Kategorie *i0*. Die spezielle Motivation für diese Analyse ist hier nicht wichtig, wir kürzen daher auch die Notation ab und schreiben solche Strukturen künftig so wie im rechten Teil der Abbildung dargestellt.

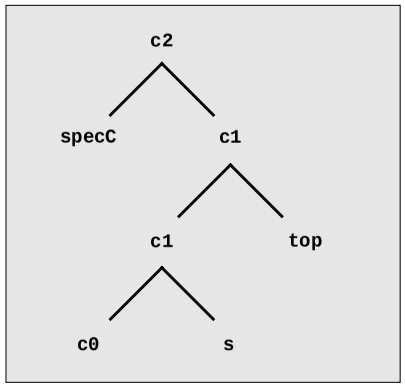
X-bar-Schema (Standardsatz)



Fragesätze können nicht durch das obenstehende X-bar-Schema basisgeneriert sein, weil die Frageausdrücke in einer satzexternen Position stehen (bzw. durch einen grammatischen Prozess dorthin bewegt worden sind). Das Gleiche gilt für Relativpronomen und topikalisierte Nominalphrasen. Aus diesem Grund liegt es nahe, diese externen Positionen in das X-bar-Schema zu integrieren. Es werden in der Regel drei sogenannte Positionskategorien unterschieden: die *c0*-Position, die *top*-Position und die *specC*-Position. Die Position bzw. Kategorie *c0* (auch *comp* genannt) enthält vor allem Satzkonjunktionen und Hilfsverben als lexikalische Elemente, jedoch keine maximalen Projektionen. Die Kategorie *c0* wird deshalb als funktionaler Kopf aufgefasst und bildet zusammen mit *s* als Komplement die erste Projektionsebene *c1* der Complementizer-Phrase *c2*. Im Gegensatz zu den Elementen von *c0* sind *wh*-Phrasen

(= Frageausdrücke und Relativpronomen) maximale Projektionen (*n2* oder *p2*) und erfüllen alle Eigenschaften, die einen Spezifikator charakterisieren. Auch topikalisierte Nominalphrasen, die als Folge einer Quantorenanhebung in einer satzexternen Position stehen, sind maximale Projektionen. Sie unterscheiden sich jedoch von *wh*-Phrasen dadurch, dass sie lokal gebunden sind. Für *wh*-Phrasen und topikalisierte Nominalphrasen werden deshalb zwei unterschiedliche Positionen postuliert. Im Fall von *wh*-Phrasen ist das die Spezifikatorposition *specC* und im Fall von topikalisierten Nominalphrasen die Position *top*. Ob es sich bei *top* um einen Spezifikator oder um ein Adjunkt handelt, ist umstritten [vgl. Fanselow & Felix 87:196, Farke 94:40]. Die Complementizer-Phrase *c2* wird auch (intuitiver) als *s1* bezeichnet. Unter Uniformitätsaspekten können alle Sätze als *c2* bzw. *s1* aufgefasst werden, denen unterschiedliche formale Ausprägungen zugrunde liegen.

X-bar-Schema (Complementizer-Phrase)



ATTEMPTO CONTROLLED ENGLISH (ACE). Für die Beschreibung und die Implementation der Sprache ACE ist das X-bar-Schema instruktiv, weil es zeigt, dass Phrasen und Sätze in natürlichen Sprachen systematisch aufgebaut sind und ihre Eigenschaften im Prinzip lokal beschrieben werden können. Im Gegensatz zur generativen Grammatik, die auf die vom X-bar-Schema erzeugten Grundstrukturen (= Tiefenstrukturen) Bewegungsregeln, d.h. Transformationsregeln, anwendet, um zu den zulässigen syntaktischen Variationen (= Oberflächenstrukturen) zu gelangen, wird für die Beschreibung von ACE ein sogenannter monostraler Ansatz verwendet, der ohne Transformationen auskommt und von einer einzigen syntaktischen Oberflächenstruktur ausgeht. Das führt dazu, dass lexikalische Kategorien und syntaktische Phrasen durch einen kontextsensitiven Formalismus beschrieben werden müssen, wo grammatische Kategorien mit partiell spezifizierten Merkmalstrukturen annotiert sind, um beispielsweise ungebundene Abhängigkeiten bei Frage- und Relativsätzen oder lokale Abhängigkeit bei topikalisierten Nominalphrasen behandeln zu können [vgl. Borsley 91:92 ff., Bennett 95:147 ff.].

5.2.2 Diskursrepräsentationstheorie

Motivation

Die korrekte formale Repräsentation eines natürlich-sprachlichen Spezifikationstextes verlangt neben der Verarbeitung von einzelnen Sätzen und ihren Konstituenten auch die Berücksichtigung der satzübergreifenden Beziehungen. Diese Textbezüge werden durch sprachliche Kohäsionsmittel hergestellt und ermöglichen den Ausdruck von komplexen propositionalen Inhalten. Phänomene wie pronominale Referenz, Ellipsen und Informationen über temporale Zusammenhänge können nur erfolgreich behandelt werden, wenn der vorausgehende Text bzw. der geäusserte Diskurs miteinbezogen wird. Die klassische Diskursrepräsentationstheorie (DRT) stellt einen Formalismus bereit, der die Bedeutung der Sätze im Kontext expliziert [vgl. Kamp 81, Guenthner et al. 86, Hess 91, Kamp & Reyle 93]. In der DRT wird jeder Satz relativ zu einer Diskursrepräsentationsstruktur (DRS) des vorausgehenden Textes übersetzt und modelltheoretisch interpretiert. Die DRS repräsentiert die textuelle Information der vorausgehenden Sätze in strukturierter Form und erlaubt eine regelbasierte Auflösung der textuellen Bezüge durch den DRS-Konstruktionsalgorithmus während der Verarbeitung eines Satzes. Die Bedeutung eines Satzes kommt also nicht isoliert zustande, sondern sie ist gegeben durch die Informationszunahme, die eine bestehende DRS bei der Verarbeitung des Satzes erfährt. Das führt zu einer erweiterten DRS, die die Hintergrundsinformation für die noch zu verarbeitenden Sätze liefert. Für ACE wird eine Variante der klassischen DRT eingeführt und mit einer Theorie für Eventualitäten (= Ereignisse und Zustände) kombiniert [vgl. Parsons 90:20 ff., Asher 93:57 ff., Kamp & Reyle 93:504 ff.]. Diese Kombination hat immer noch die Mächtigkeit der Prädikatenlogik erster Stufe und wird in ACE als DRT-E bezeichnet.

Die DRT-E geht davon aus, dass Verben in natürlich-sprachlichen Äusserungen Diskursreferenten für Eventualitäten einführen, und zwar in ähnlicher Weise wie indefinite Nominalphrasen Diskursreferenten für Individuen einführen. Als wichtigstes Unterscheidungsmerkmal zwischen reifizierten Ereignissen und Zuständen in der DRT-E gilt, dass Ereignisse sich konzeptionell auf einen Zeitpunkt beziehen (ohne dass dabei eine Aussage über ihre Laufzeit gemacht wird), wohingegen Zustände ab einem

Zeitpunkt gelten und solange andauern, bis sie explizit aufgehoben werden. In ACE bestimmt die textuelle Reihenfolge der Konstituenten die zeitliche Anordnung der Eventualitäten implizit, solange keine temporalen Verknüpfungen oder Diskurssignale auftauchen, die die vorgegebene Linearität explizit aufheben. Die Laufzeit von Eventualitäten kann in ACE durch adjungierte Zeitangaben - in Form von Adverbien und Präpositionalphrasen - in den Verbalphrasen spezifiziert werden, was bei der Übersetzung in die DRT-E zu Modifikatoren von Ereignissen und Zuständen führt.

Diskursrepräsentationsstrukturen

Die DRT-E repräsentiert einen ACE Spezifikationsdiskurs *SD*, der aus einer Menge von zusammenhängenden Sätzen besteht, durch eine einzige logische Einheit, eine sogenannte Diskursrepräsentationsstruktur (DRS). Eine DRS *K* ist ein geordnetes Paar $<U, Con>$, wobei *U* aus einer Menge von Diskursreferenten besteht und *Con* eine Menge von Bedingungen in fester Reihenfolge ist, die bezüglich der Diskursreferenten gelten sollen. Bei den Diskursreferenten *U* handelt es sich um Variablen. Die Diskursreferenten können weiter in Individuenreferenten (*I*), Ereignisreferenten (*E*) und Zustandsreferenten (*S*) unterteilt werden. Die Bedingungen *Con* können einfach oder zusammengesetzt sein.

Einfache Bedingungen

Einfache Bedingungen sind atomare Formeln, die aus einem Prädikatssymbol und einem Tupel von Termen bestehen. Ein Term ist entweder ein Diskursreferent, eine Konstante oder ein zusammengesetzter Term. Terme stehen für Entitäten aus dem Diskursbereich und Prädikatssymbole bezeichnen Relationen zwischen den Entitäten. Einfache Bedingungen können weiter unterteilt werden in Grundbedingungen und vordefinierte Bedingungen. Grundbedingungen haben die Form $P(U_1, \ldots, U_n)$, wobei *P* ein *n*-stelliges Prädikatssymbol ist und U_1 bis U_n Terme sind. Vordefinierte Bedingungen sind atomare Formeln mit einer speziellen Form und einer festgelegten semantischen Funktion (z.B. Ereignis, Zustand, Modifikation, usw.). Beispielsweise besteht die vordefinierte Bedingung $event(E, P(I_1, \ldots, I_n))$ aus dem Prädikatssymbol *event* und den zwei Termen *E* und $P(I_1, \ldots, I_n)$. *E* ist ein Ereignisreferent und steht für ein punktuelles Ereignis. $P(I_1, \ldots, I_n)$ ist der Term, der die Relation benennt.

Einfache Bedingungen haben die folgenden Formen und Funktionen:

Einfache Bedingungen

Form	Funktion
$P(U_1, \ldots, U_n)$	Relation
$I_2=I_1$	Identität
named(I,'N')	Benennung
state(S,P(I)) event(E,P($I_1, \ldots, I_n$))	Eventualität
M(S,T) M(E,T)	Modifikation
$R(E_1, \ldots, E_n)$	Reihenfolge

Zu dieser Tabelle sind die folgenden vier Punkte anzumerken:

Erstens handelt es sich bei der Identitätsbedingung $I_2=I_1$ um eine atomare Formel, die in Infixnotation geschrieben ist und ein Gleichheitszeichen (=) als Prädikatssymbol hat. Beispielsweise führt der nominale anaphorische Bezug eines Personalpronomens in ACE zu einer Gleichsetzung der Individuenreferenten in der DRT-E, wobei I_2 für das Personalpronomen steht und I_1 für diejenige Nominalphrase, auf die die Anapher verweist.

Zweitens steht in der vordefinierten Bedingung *named(I,'N')* der Individuenreferent *I* für einen Namensträger und die Konstante *'N'* für einen Eigennamen. Die Aufgabe von *'N'* besteht darin, *I* als einen Vertreter der Namensträger durch Benennung korrekt zu identifizieren. Beispielsweise führt eine Nominalphrase mit dem Eigennamen *Yolande* in der DRT-E zu einer Bedingung der Form *named(I,'Yolande')*.

Drittens stehen Bedingungen der Form *M(S,T)* oder *M(E,T)* für die Modifikation von Eventualitäten. Das Prädikatssymbol *M* bezeichnet ein Element aus der offenen Menge *{location, origin, direction, time, frequency, duration, manner, instrument, comitative, ...}*. *S* oder *E* steht für eine Eventualität und *T* für einen Term, der die Eventualität benennt. Bei der Übersetzung entsteht beispielsweise aus dem Adverb *downstairs* ein lokaler Modifikator der Form *location(E,downstairs)* in der DRT-E und aus der Präpositionalphrase *on Monday* ein temporaler Modifikator der Form *time(E,on(I))*, *named(I,'Monday')*.

Viertens steht die vordefinierte Bedingung $R(E_1, \ldots, E_n)$ für eine zeitlich nicht-lineare

Anordnung von koordinierten Ereignissen. Dabei kann es sich entweder um eine gleichzeitige Anordnung oder um eine zeitlich unbestimmte Anordnung der Ereignisse handeln. Solche nicht-linearen Anordnungen müssen in ACE durch explizite Diskurssignale (*at the same time, in any temporal order*) auf der Sprachoberfläche angezeigt werden. In der DRT-E ist das Prädikatssymbol `R` dann eine Instanz aus der Menge `{at_same_time, any_temporal_order}`. Im Gegensatz dazu führt eine zeitlich lineare Anordnung von koordinierten Ereignissen zu keiner zusätzlichen Bedingung, da sich der zeitliche Verlauf an der textuellen Reihenfolge orientiert, die in ACE die unmarkierte Normalform bildet.

Zusammengesetzte Bedingungen

Zusammengesetzte Bedingungen bestehen aus einzelnen DRSen (K_i), die durch logische Operatoren (`IF ... THEN ...`, `NOT`, `OR ... OR ...` und `EITHER ... OR ...`) miteinander verbunden sind. Zusammengesetzte Bedingungen haben folgende Formen und Funktionen:

Zusammengesetzte Bedingungen

Form	Funktion
IF K_1 THEN K_2	Implikative Bedingung
NOT K	Negation
OR K_1 OR K_2	Inklusive Disjunktion
EITHER K_1 OR K_2	Exklusive Disjunktion

Es fällt auf, dass in der Tabelle kein logischer Operator für die Konjunktion (`AND`) auftaucht. Wenn zusammengesetzte Bedingungen eine explizite Konjunktion aufweisen, wie das in $C_1, \ldots, C_n$, AND $C_{n+1}, \ldots, C_m$ der Fall ist, dann wird die Konjunktion in der DRT-E nicht geschrieben. Die einzelnen Bedingungen werden akkumuliert und bilden eine Sequenz der Form $C_1, \ldots, C_n, C_{n+1}, \ldots, C_m$.

DRS Konstruktion

Die Konstruktion einer DRS setzt die Anwendung einer syntaktischen Analyse auf einen Spezifikationsdiskurs voraus, so dass die syntaktische Struktur der Sätze durch einen DRS-Konstruktionsalgorithmus verarbeitet werden kann. Eine DRS $K = \langle U, Con \rangle$ kommt dadurch zustande, dass eine syntaktische Komponente zuerst jedem Satz S_i des Spezifikationsdiskurses $SD = \langle S_1, \ldots, S_n \rangle$ eine eindeutige Analyse $[S_i]$ zuordnet. Der DRS-Konstruktionsalgorithmus operiert dann auf den einzelnen syntaktischen Analysen $[S_i]$ und arbeitet diese der Reihe nach ab, wobei der Algorithmus zu Beginn von einer leeren Ausgangs-DRS K_0 ausgeht. Für die syntaktische Analyse $[S_1]$ erzeugt der DRS-Konstruktionsalgorithmus eine Menge von Diskursreferenten U_1 (aufgrund der Nomen und Verben) und eine Menge von Bedingungen Con_1 (aufgrund aller Wörter und ihrer Beziehungen untereinander) und fügt diese Information in die Ausgangs-DRS K_0 ein. Dabei werden die Diskursreferenten U_1 in das Diskursuniversum U_0 eingesetzt und die Bedingungen Con_1 der Reihe nach zu der noch leeren Bedingungsmenge Con_0 hinzugefügt. Falls in Con_1 bereits reduzierbare Bedingungen vorliegen (die zum Beispiel durch Verweise innerhalb des Satzes oder durch Synonyme und Abkürzungen entstanden sind), werden diese durch den DRS-Konstruktionsalgorithmus aufgelöst. Daraus ergibt sich die bereinigte DRS K_1. Die Verarbeitung des zweiten Satzes $[S_2]$ resultiert wiederum in einer Menge von Diskursreferenten U_2 und Bedingungen Con_2, die die bestehende DRS K_1 unter Berücksichtigung der bereits vorliegenden Bedingungen zur DRS K_2 erweiteren, wobei nun auch satzübergreifende Beziehungen - wie anaphorische Referenzen und syntaktische Ellipsen - durch den DRS-Konstruktionsalgorithmus aufgelöst werden.

Die DRT-E ist nicht an eine bestimmte syntaktische Theorie für die Konstruktion von DRSen gebunden. Für eine angemessene Diskursrepräsentation wird einzig eine eindeutige syntaktische Analyse der Eingabesätze vorausgesetzt. Bei unserer Realisierung gehen wir so vor, dass die einzelnen Eingabesätze der Reihe nach durch eine unifikationsbasierte Phrasenstrukturgrammatik analysiert werden und die DRS gleichzeitig - während der syntaktischen Analyse eines Satzes - konstruiert wird. Die syntaktische Analyse und der DRS-Konstruktionsalgorithmus sind eng miteinander verzahnt.

DRS Interpretation

Eine DRS $K = \langle U, Con \rangle$ wird durch die Einbettung in ein prädikatenlogisches Modell $M = \langle D, F \rangle$ interpretiert. Ein Modell M ist ein Ausschnitt der Welt und besteht aus einem nicht-leeren aussersprachlichen Diskursbereich D (das sind eine Menge von Entitäten E) und einer Einbettungsfunktion F, die jeden Diskursreferenten aus dem Universum U auf eine Entität E in D abbildet, so dass die Entitäten von M alle in der DRS K ausgedrückten Bedingungen Con erfüllen. Ein Spezifikationsdiskurs SD ist genau dann wahr, wenn es für die dazugehörige DRS K mindestens eine Einbettung F in das Modell M gibt. Das heisst, nur wenn die DRS K in das fixierte Modell M eingebettet werden kann, ohne dass Widersprüche auftreten, ist der Spezifikationstext SD wahr.

Beispiel: Einfache DRS

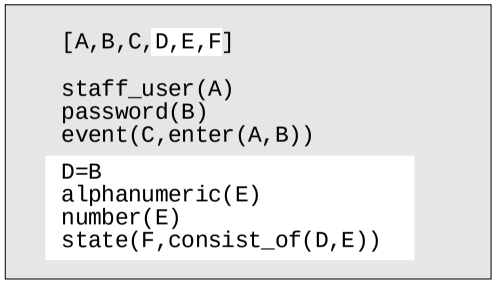
Betrachten wir zuerst eine einfache DRS, die durch die Verarbeitung eines Spezifikationsdiskurses entsteht, der die folgenden beiden Aussagesätze umfasst:

(86) *A staff user enters the password.*

(87) *It consists of an alphanumeric number.*

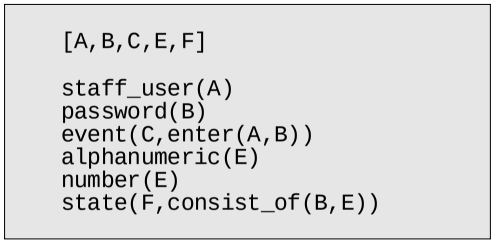
Zwischen diesen beiden Aussagesätzen gibt es eine anaphorische Beziehung, die während der syntaktischen Analyse berücksichtigt werden muss. Die anaphorische Referenz des Personalpronomens (*it*) kann nur korrekt aufgelöst werden, wenn es auf die vorausgehende definite Nominalphrase (*the password*) bezogen wird. Der Spezifikationsdiskurs wird syntaktisch analysiert und in die untenstehende Haupt-DRS übersetzt.

Einfache DRS



Das Universum der Haupt-DRS besteht aus einer Menge von existentiell quantifizierten Diskursreferenten in eckigen Klammern (`[]`). Danach folgt eine Menge von einfachen Bedingungen in fester Reihenfolge, in denen die entsprechenden Diskursreferenten auftreten. Der Aufbau der Haupt-DRS geht von der textuellen Reihenfolge der beiden Aussagesätze aus und beginnt mit der leeren Ausgangs-DRS K_0. Die Verarbeitung des ersten Satzes (86) führt zu den drei Diskursreferenten *A, B, C* und zu den drei Bedingungen - auf dunklem Hintergrund -, die für die drei Diskursreferenten gelten. Diese Information erzeugt aus der leeren DRS K_0 die DRS K_1. Der zweite Satz (87) wird im Kontext der nun vorliegenden DRS K_1 verarbeitet. Dadurch wird die formale Repräsentation der anaphorischen Referenz zwischen dem Personalpronomen *it* in (87) und der definiten Nominalphrase *the password* in (86) auf der Grundlage von syntaktischen Bedingungen (Numerus, Gender) und textueller Beschränkung (minimale textuelle Distanz) möglich. Diese Vorgehensweise ist eine starke Vereinfachung, die aber im Rahmen einer kontrollierten Sprache dort zu Eindeutigkeit führt, wo sonst oft nur nicht-linguistische Plausiblitätskriterien weiterhelfen. Die Verarbeitung des zweiten Satzes setzt die drei Diskursreferenten *D, E, F* und die vier Bedingungen - auf hellem Hintergrund - in die DRS K_1 ein, was zur DRS K_2 führt. Die entstehende Identitätsbedingung `D=B` drückt die Gleichheit zwischen den beiden Diskursreferenten *D* und *B* aus. Auf der linken Seite des Gleichheitszeichens befindet sich der Diskursreferent *D*, der für das Personalpronomen *it* steht und die Anapher bildet. Auf der rechten Seite steht der Diskursreferent *B*, der aus der definiten Nominalphrase *the password* abgeleitet worden ist und das Antezedens für das Personalpronomen *it* bildet. Da die beiden Diskursreferenten (auf die gleiche Entität in der Welt) koreferieren, kann die Identitätsbedingung `D=B` in einem weiteren Schritt durch den DRS-Konstruktionsalgorithmus aufgelöst werden. Daraus resultiert die bereinigte DRS K_2:

Bereinigte DRS



Diese DRS K_2 ist genau dann in einem Modell *M* wahr, wenn man Entitäten ***a***, ***b***, ***c***, ***e*** und ***f*** in *M* findet, die den Diskursreferenten *A, B, C, E* und *F* so zugeordnet werden können, dass diese Entitäten von *M* alle in der DRS K_2 ausgedrückten Bedingungen erfüllen. Das heisst, K_2 ist genau dann wahr, wenn ***a*** der Mitarbeiter, ***b*** das Passwort und ***e*** eine alphanumerische Nummer in *M* sind und in *M* ***c*** dasjenige Ereignis ist, wo ***a b*** eingibt und ***f*** derjenige Zustand ist, in dem ***b e*** hat.

Beispiel: Komplexe DRS

Konditionale Sätze, allquantifizierte Nominalphrasen, Konstituentennegation und disjunktive Verknüpfungen führen komplexe untergeordnete DRSen in die Haupt-DRS ein. Zu Demonstrationszwecken erweitern wir den oben eingeführten Spezifikationsdiskurs mit den beiden Aussagesätzen (86) und (87) um den folgenden konditionalen Satz:

(88) *If a copy of MacWorld is checked out to a borrower and the staff user returns the copy then it is available.*

Die Übersetzung dieses konditionalen Satzes erweitert die bestehende DRS folgendermassen:

Erweiterte DRS

```
[A,B,C,E,F,H]

staff_user(A)
password(B)
event(C,enter(A,B))
alphanumeric(E)
number(E)
state(F,consist_of(B,E))

named(H,'MacWorld')
IF
  [G,I,J,K,L,M]
  copy(G)
  of(G,H)
  borrower(I)
  state(J,checked_out_to(G,I))
  staff_user(K)
  K=A
  copy(L)
  L=G
  event(M,return(K,L))
  THEN
    [N,O]
    N=L
    state(O,available(N))
```

Die konditionale DRS besteht aus den zwei Sub-DRSen K_3 und K_4, die durch die logische Operation *IF ... THEN ...* miteinander verbunden sind. Die erste Sub-DRS K_3 enthält die Diskursreferenten und Bedingungen, die aus dem konditionalen Nebensatz (*a copy of MacWorld is checked out to a borrower and the staff user returns the copy*) abgeleitet worden sind, und bildet die Antezedens-DRS. Von dieser DRS K_3 hängt die Sub-DRS K_4 ab, die die Diskursreferenten und Bedingungen aus dem Hauptsatz (*it is available*) enthält und die Konsequens-DRS bildet.

Zu dieser konditionalen DRS sind drei Punkte anzumerken:

Erstens werden im konditionalen Satzgefüge (88) die beiden definiten Nominalphrasen *the staff user* und *the copy* anaphorisch gebraucht. Ausserdem taucht wiederum ein Personalpronomen (*it*) in (88) auf, das sich hier auf die definite Nominalphrase *the copy* im Nebensatz bezieht (die bereits anaphorisch gebraucht wird). Das führt in der konditionalen DRS zu entsprechenden Repräsentationen. Beispielsweise wird die definite Nominalphrase *the copy* in der Antezedens-DRS durch den Diskursreferenten `L` und die zwei Bedingungen `copy(L)` und `L=G` repräsentiert, die anschliessend durch den DRS-Konstruktionsalgorithmus reduziert werden können. Ebenso kann die Identitätsbedingung `N=L` in der Konsequens-DRS, die für die pronominale Referenz (*it* → *the copy*) steht, reduziert werden.

Zweitens sind der Diskursreferent `H` und die Bedingung `named(H,'MacWorld')`, die vom Eigennamen *MacWorld* abgeleitet wurden, nicht in der konditionalen DRS zu finden, sondern sind in die Haupt-DRS eingetragen worden. Dies ist deshalb geschehen, weil Eigennamen ihre Träger direkt identifizieren. Der Träger des Eigennamens wird mit demjenigen Individuenreferent assoziiert, welcher aus dem Eigennamen abgeleitet wurde. Nachdem einmal eine Identifikation hergestellt worden ist, bleibt ein Eigennamen auf dieselbe Entität anwendbar. Durch den Eintrag in die Haupt-DRS wird garantiert, dass der Diskursreferent `H` als Antezedens für nachfolgende Anaphern zugänglich bleibt.

Drittens ist die Antezedens-DRS der Konsequens-DRS übergeordnet. Diese strukturelle Eigenschaft der konditionalen DRS beschränkt den Skopus der Diskursreferenten und die Zugänglichkeit der Diskursreferenten für den anaphorischen Bezug implizit. Im Gegensatz zu den (existentiell quantifizierten) Diskursreferenten in der Haupt-DRS ist jeder neu eingeführte Diskursreferent in der Antezedens-DRS implizit universell quantifiziert und hat weiten Skopus über die implizit existentiell quantifizierten Diskursreferenten in der Konsequens-DRS. Ausserdem sind die Diskursreferenten in der Antezedens-DRS von der Konsequens-DRS aus zugänglich, aber nicht umgekehrt. Kataphern (d.h. textuelle Vorausverweise) sind in ACE nicht zulässig [vgl. Kamp 81:316 ff.].

Die (bereinigte) DRS *IF* K_3 *THEN* K_4 ist in einem Modell `M` genau dann wahr, wenn jede Einbettungsfunktion von K_3 erweitert werden kann auf eine Einbettungsfunktion von K_4. Somit müssen sich Entitäten **`g`**, **`h`**, **`i`**, **`j`** und **`m`** im Modell `M` finden lassen, die man den Diskursreferenten `G`, `H`, `I`, `J` und `M` zuordnen kann, so dass die Entitäten die in der Sub-DRS K_3 ausgedrückten Bedingungen im Modell `M` erfüllen. Ausserdem muss es

(dann) auch der Fall sein, dass es eine Entität **o** gibt, so dass die in der Sub-DRS K_4 ausgedrückte Bedingung (`state(O,available(N))`) im Modell `M` erfüllt ist. Diese modelltheoretische Interpretation macht keine Aussage über den kausalen Zusammenhang von K_3 und K_4, sondern nur über den logischen Zusammenhang. Das Verhältnis zwischen der konditionalen DRS und der materialen Implikation in der klassischen Logik wird in [Kamp & Reyle 93:172 ff.] eingehend diskutiert.

Strukturelle Restriktionen auf der DRS

Die Auflösung des Bezugs zwischen einem nominalen anaphorischen Ausdruck und einem Antezedens kommt in DRT-E einerseits durch die interne Struktur der DRSen - das heisst durch strukturimmanente Zugänglichkeitsbeschränkungen - und andererseits durch syntaktische Beschränkungen (Numerus, Gender) zustande. Ausser diesen strikten Beschränkungen spielt bei der Sprache ACE die Distanz zwischen einem anaphorischen Ausdruck und dem Antezedens eine entscheidende Rolle. Definitionsgemäss bildet in ACE immer die letzterwähnte zugängliche und passende Nominalphrase das Antezedens. Dies ist eine einschneidende Restriktion, die den Schreibprozess ohne Zweifel beeinflusst, doch der Gewinn ist ein Mass an Klarheit und Effizienz, wie das für die Verarbeitung von voller natürlicher Sprache nie möglich ist.

Die Zugänglichkeitsbeschränkungen der DRT-E legen fest, dass ein Diskursreferent nur mit einem Diskursreferenten der gleichen oder der übergeordneten DRS gleichgesetzt werden kann. Wie bereits angesprochen, ist in einer konditionalen DRS die Antezedens-DRS der Konsequens-DRS übergeordnet. Aus diesem Grund war es im Satz (88) überhaupt möglich, das Personalpronomens *it* auf die Nominalphrase *the copy* zu beziehen. Eine Ausnahme der Regel bilden disjunktive Verknüpfungen in der DRT-E, wo eine anaphorische Referenz - aus rein empirischen Erwägungen - auch zwischen untergeordneten DRSen zulässig ist. Beispielsweise führt die Übersetzung der folgenden disjunktiven Satzverbindung

(89) *Either a copy of MacWorld is checked out to a borrower or it is available.*

zu zwei untergeordneten DRSen (`EITHER` K_i `OR` K_{i+1}) der gleichen Einbettungstiefe. Trotzdem kann in der DRT-E von einem Diskursreferenten aus K_{i+1} auf einen Diskursreferenten aus K_i Bezug genommen werden.

Anaphorische Referenz ist nicht möglich, wenn eine indefinite Nominalphrase in einem vorausgehenden Satz im Skopus eines Negationausdrucks (90), eines universellen Quantors (91) oder einer Implikation (92) steht:

(90) *John does not own a book. *John reads it.*

(91) *Every staff user borrows a book that he reads. *He likes it.*

(92) *If John reads a book then he summarizes it. *He owns it.*

In allen drei Fällen ist der Diskursreferent für das Antezedens in einer Sub-DRS lokalisiert, die der DRS mit dem aktuellen Diskursreferenten für die Anapher untergeordnet ist. Das Problem entsteht dadurch, dass die durch die indefiniten Nominalphrasen eingeführten Diskursreferenten zu tief eingebettet werden (nämlich in einer Sub-DRS einer Negation beziehungsweise einer Konsequenz), so dass ein entsprechender Bezug nicht möglich ist. Solche Sätze werden in ACE nicht übersetzt.

In natürlich-sprachlichen Texten ist es manchmal schwierig zu entscheiden, was von den Voraussetzungen eines konditionalen Satzes abhängt, da der bedingte Bereich aus mehr als einem Hauptsatz bestehen kann und nicht immer durch einen Satzpunkt begrenzt ist. Betrachten wir den folgenden Spezifikationsausschnitt [vgl. Fuchs et al. 94]:

(93) *Every customer has a correct personal code.*

(94) *The personal code of a customer consists of a number.*

Die Übersetzung des Satzes (93) mit der allquantifizierten Nominalphrase *every customer* führt in der DRT-E zu einer implikativen Bedingung (`IF` K_i `THEN` K_{i+1}), denn der Satz kann folgendermassen paraphrasiert werden:

(95) *IF X is a customer THEN X has a correct personal code.*

Für eine adäquate Übersetzung nach DRT-E stellt sich die Frage, ob der Aussagesatz (94) vom hypothetischen Teil in (93) abhängig ist oder nicht. Da es keinen einfachen Algorithmus gibt, der dies entscheiden kann [vgl. Kamp & Reyle 93:145], muss man die entsprechenden Lesarten in ACE durch sprachliche Mittel verdeutlichen. Die erste Lesart kann "unter Kontrolle" gebracht werden, indem man für die in (94) ausgedrückte Information einen objektmodifizierenden Relativsatz bildet und den Satz (93) folgendermassen umschreibt:

(96) *Every customer has a correct personal code that consists of a number.*

Wenn es sich bei (94) jedoch um einen Aussagesatz handelt, der nicht von der hypothetischen Information in (93) abhängt, sondern als kumulative Aussage behandelt werden muss, ist es aufgrund der Zugänglichkeitsbeschränkungen der DRS nicht möglich, zwischen den Diskursreferenten für die indefinite Nominalphrase *a correct*

personal code in (93) und der definiten Nominalphrase *the personal code* in (94) einen anaphorischen Bezug herzustellen. Als Ausweg bleibt einzig, die Reihenfolge der beiden Sätze umzustellen.

5.3 Die Sprache ACE im Überblick

Wir geben hier einen kurzen Überblick über die Sprache ACE, der auf die nächsten Kapitel vorbereitet, in denen dann die Begründung der getroffenen Entscheidungen für den Sprachentwurf eingehender diskutiert wird.

Das Vokabular von ACE

Das Vokabular von ACE besteht aus zwei Hauptklassen von Wörtern: Inhaltswörter und Funktionswörter. Die anwendungsspezifischen Inhaltswörter können in die syntaktischen Kategorien *Nomen*, *Verb*, *Adjektiv* und *Adverb* unterteilt werden. Inhaltswörter bilden eine offene Wortklasse, die auf relativ unproblematische Weise Erweiterungen und Neubildungen zulässt. Diese offene Wortklasse enthält eine grosse Menge von Mitgliedern, die meist an ein entsprechendes Anwendungsgebiet gebunden sind und dort eine relativ selbständige Bedeutung haben. Die vordefinierten Funktionswörter können in die syntaktischen Kategorien *Präposition, Spezifikator, Pronomen, Koordinator, Subordinator* und *Fragewort* unterteilt werden. Funktionswörter bilden eine geschlossene Wortklasse, deren Anzahl und Zusammensetzung relativ stabil ist. Diese geschlossene Wortklasse enthält in ACE eine kleine Menge von vordefinierten Mitgliedern, die nicht an ein spezielles Anwendungsgebiet gebunden sind und im wesentlichen strukturelle Funktionen erfüllen, indem sie syntaktische und textuelle Beziehungen herstellen. Im Gegensatz zu den Funktionswörtern, die im Lexikon nur nachschlagbar sind, können Inhaltswörter durch den Anwendungsspezialisten während des Spezifikationsprozesses definiert und zum Lexikon des Attempto Systems hinzugefügt werden. Dabei wird davon ausgegangen, dass schulgrammatisches Wissen ausreicht, um neue Inhaltswörter im Lexikon zu klassifizieren.

Einfache Sätze in ACE

Die Grundeinheit einer ACE Spezifikation ist der einfache Satz. Einfache ACE Sätze folgen einem einzigen abstrakten Satzbauplan, der das syntaktische Grundschema bildet, das immer wieder anwendbar ist, indem es durch Wörter und Phrasen aus vorgegebenen Kategorien realisiert wird. Einfache ACE Sätze bestehen aus einem Subjekt und einem Prädikat. Das Prädikat setzt sich aus einem verbalen Kopf mit den obligatorischen internen Komplementen zusammen. Das Prädikat kann durch optionale Adjunkte und im Negationsfall durch ein Negationswort erweitert sein. Von der Satzart her handelt es sich bei einfachen ACE Sätzen um Aussagesätze, deren Verbformen in der dritten Person Singular (oder Plural) stehen und die ausschliesslich das Präsens (= *simple present tense*) als Tempus aufweisen. Die Aussageweise ist immer der Indikativ und die Handlungsrichtung ist das Aktiv. Die zulässige Ausprägung der

Verbform ist in einem normativen Schreibprinzip festgehalten und ist ein Beispiel aus einer Reihe von Schreibprinzipien, die den Gebrauch der Sprache regeln.

Zusammengesetzte Sätze in ACE

Aus einfachen Sätzen können mit Hilfe von Koordinatoren und Subordinatoren zusammengesetzte Sätze gebildet werden. Dabei müssen zwei Erscheinungsformen von zusammengesetzten ACE Sätzen unterschieden werden: Satzverbindungen und Satzgefüge. Eine Satzverbindung besteht aus mindestens zwei einfachen Sätzen, die durch einen Koordinator verbunden sind. Koordinatoren zeigen in ACE entweder ein kopulatives Verhältnis oder ein disjunktives Verhältnis zwischen den Teilsätzen an. Ein Satzgefüge besteht aus einem einfachen Satz als Hauptsatz und einem abhängigen Nebensatz, der durch einen Subordinator eingeleitet wird. Subordinatoren markieren Nebensätze als von ihren übergeordneten Sätzen abhängig und zeigen in ACE konditionale oder restriktive Verhältnisse an. ACE Sätze müssen komplett sein; sie dürfen aber im Text gewisse Formen von Anaphern und syntaktischen Ellipsen (Koordinationsreduktion) aufweisen, die aber syntaktisch rekonstruierbar sein müssen. Anaphern und Ellipsen machen die Sprache kompakt und ihren Gebrauch natürlich. Die zulässigen Formen von Anaphern und Ellipsen sind als Schreibprinzipien formuliert.

Fragesätze in ACE

Fragesätze werden aus einfachen Sätzen durch Inversion oder auch durch Fragewörter gebildet. Einfache Fragesätze können in ACE vom Anwendungsspezialisten dazu verwendet werden, um die Anforderungsspezifikation nach der Verarbeitung zu validieren. Mit Entscheidungsfragen kann festgestellt werden, ob ein spezifizierter Sachverhalt zutrifft oder nicht. Ergänzungsfragen können dazu verwendet werden, um bestimmte Aspekte eines spezifizierten Sachverhaltes zu untersuchen.

Normative Schreibprinzipien in ACE

Die Schreibprinzipien regeln in erster Linie den Gebrauch der Sprache und reduzieren die Anzahl der möglichen Lesarten für einen ACE Satz drastisch. Diese Schreibprinzipien und die deterministische Parsingstrategie führen zusammen dazu, dass für jeden ACE Satz nach der Übersetzung nur eine Lesart übrig bleibt, die dem Benutzer in einer Paraphrase verdeutlicht wird.

ACE auf einen Blick

Folgendes grammatische Modell - in Form eines mnemonischen Überblicks - bildet die Grundlage für die Spezifikationssprache ACE, die einerseits genügend ausdrucksstark ist (um das Bibliotheksbeispiel adäquat und leicht verständlich zu spezifizieren) und andererseits effizient von einem Computer in DRT-E übersetzt werden kann.

- **Vokabular**
  bestehend aus
  - **anwendungsspezifischen Inhaltswörtern** der Kategorien
    - Nomen (`staff user`, `borrower`, `LibDB`, …)
    - Verb (`works`, `enters`, `adds to`, `is`, …)
    - Adjektiv (`correct`, `valid`, `wrong`, …)
    - Adverb (`manually`, `daily`, `correctly`, …)
  - **vordefinierten Funktionswörtern** der Kategorien
    - Präposition (`at`, `for`, `in`, `into`, `of`, `on`, …)
    - Spezifikator (`a`, `the`, `N's`, `every`, `no`, `not`, …)
    - Pronomen (`he`, `his`, `she`, `her`, `it`)
    - Koordinator (`and`, `or`, `either … or`)
    - Subordinator (`who`, `which`, `that`, `if`)
    - Fragewort (`who`, `which`, `where`, …)
- **Einfache Sätze**
  mit dem abstrakten Satzbauplan
  Subjekt + [{ Negation } + verbaler Kopf + Komplement { + Adjunkt }]
  enthalten Inhaltswörter und Funktionswörter der Kategorien
  Präposition, Spezifikator, Pronomen, Koordinator
- **Zusammengesetzte Sätze**
  aus einfacheren Sätzen durch Koordinatoren und Subordinatoren gebildet
- **Fragesätze**
  aus einfachen Sätzen durch Inversion oder auch durch Fragewörter gebildet
- **Normative Schreibprinzipien**
  regeln den Gebrauch der Sprache

5.4 Das Vokabular von ACE

Inhaltswörter und Funktionswörter

Das Vokabular der Sprache ACE ist in die beiden Hauptklassen *Inhaltswörter* und *Funktionswörter* eingeteilt. Die Klasse der Inhaltswörter setzt sich je nach Anwendungsbereich aus einer grossen Anzahl von immer wieder neuen Mitgliedern zusammen. Inhaltswörter sind Wörter, die eine relativ selbständige (mentale) Vorstellung bewirken und einen Grossteil zur Bedeutung eines Satzes beitragen, indem sie Entitäten oder Eigenschaften von Entitäten benennen (z.B. *staff user, return, available, today*). Die Klasse der Funktionswörter besteht aus Mitgliedern, die nicht stark von den unterschiedlichen Anwendungsbereichen abhängen und ein stabiles Inventar von geringer Anzahl und hoher Frequenz bilden. Funktionswörter sind Wörter, die hauptsächlich strukturelle Funktionen erfüllen, indem sie syntagmatische, syntaktische und textuelle Beziehungen herstellen (z.B. *she, the, and, if, not, every*). Die Einteilung in Inhaltswörter und Funktionswörter macht in ACE deshalb Sinn, weil die Mitglieder dieser beiden Klassen verschieden stark an die wechselnden Anwendungsbereiche gebunden sind. Im Gegensatz zu den Inhaltswörtern müssen die Mitglieder aus der Klasse der Funktionswörter für die unterschiedlichen Anwendungsbereiche nicht jedes Mal neu definiert werden.

Kategorisierung und Lexikalisierung

Neben der übergeordneten Einteilung in Inhaltswörter und Funktionswörter werden alle Wörter in ACE aufgrund ihrer syntaktischen Eigenschaften und ihres semantischen Potentials feiner unterteilt und einer Wortkategorie zugeordnet. (Unter dem semantischen Potential wird diejenige Information verstanden, die die Wörter in Form von logischen Bedingungen zu einer DRS in der DRT-E beitragen können). Die Wortkategorien sind Klassen von Ausdrücken, welche in Sätzen gegeneinander austauschbar sind, ohne dass die syntaktische Wohlgeformtheit verloren geht. Die Inhaltswörter werden in ACE weiter in die Kategorien *Nomen*, *Verb*, *Adjektiv* und *Adverb* eingeteilt. Die Funktionswörter werden den Kategorien *Präposition, Spezifikator, Pronomen, Koordinator, Subordinator, Fragewort* zugeordnet. Diese Wortkategorien strukturieren letztendlich auch das Lexikon des Attempto Systems. Dort werden lexikalische Einträge durch komplexe Merkmalstrukturen in Form von Merkmal-Werte-Paaren dargestellt, deren Werte entweder atomare Formeln oder wiederum Merkmalstrukturen sein können.

Informell kann ein Lexikoneintrag für ACE als eine komplexe Merkmalstruktur beschrieben werden, die aus vier Komponenten besteht: einer Merkmalstruktur für die

graphische Wortform; einer Merkmalstruktur für die syntaktischen Eigenschaften der Wortform, einer Merkmalstruktur für die logischen Bedingungen, die zur DRS-Konstruktion beitragen und einer Merkmalstruktur für optionale Informationen (Synonyme und Kommentare). Bei der Lexikalisierung der Wörter gibt es eine klare Arbeitsteilung: Anwendungsspezialisten können nur Inhaltswörter zum Lexikon hinzufügen und dort modifizieren. Für die vordefinierten Funktionswörter liegt die Kompetenz bei den Experten; nur sie dürfen Funktionswörter zum Lexikon hinzufügen oder verändern.

Struktur der Wörter

Die Wörter aller Kategorien in ACE können einfach oder zusammengesetzt sein. Einfache Wörter bestehen aus einem einzelnen Element (z.B. *copy*). Wird ein einfaches Wort (z.B. *valid*) durch ein Affix (z.B. *in-*) erweitert, so spricht man von Derivation. Bei der lexikalischen Derivation findet eine Bedeutungsveränderung statt. Das aus dem Derivationsprozess entstehende Wort (z.B. *invalid*) wird in ACE wie ein selbständiges einfaches Wort behandelt und lexikalisiert. Zusammengesetzte Wörter bestehen aus zwei oder mehr Elementen und sind das Ergebnis von Wortbildungsprozessen (z.B. *bar code reader*). Werden mindestens zwei Wörter miteinander kombiniert, spricht man von einer Komposition. Bei zusammengesetzten Wörtern kann es vorkommen, dass die einzelnen Elemente durch Bindestriche verbunden (z.B. *book-end*) oder zusammengeschrieben (z.B. *bookworm*) werden. Alle drei Schreibformen sind in ACE zulässig. Wichtig ist, dass neben den einfachen Wörtern auch die zusammengesetzten Wörter eine *unteilbare* Einheit in ACE bilden, die eine grammatische Funktion und ein semantisches Potential haben. Die Anwendungsspezialisten müssen sich also nicht um die innere Struktur von zusammengesetzten Wörtern kümmern, sondern sie können sich beim Schreiben der Spezifikation ausschliesslich auf die sachlich angemessene Verwendungsweise der Wörter konzentrieren.

Synonyme und Abkürzungen

Synonyme sind Hilfsmittel bei der Textproduktion und erleichtern die Wortwahl durch Bezeichnungsalternativen. Zwei Wörter sind dann (strikte) Synonyme voneinander, wenn sie in einem Satz aus einem bestimmten Anwendungsbereich wechselseitig ersetzbar sind und nach dieser Substitution die Ausgangsbedeutung erhalten bleibt. Synonyme bilden in ACE ein Reihung mit einem dominanten Leitwort und einer Anzahl von peripheren Bezeichnungsalternativen. Der Anwendungsspezialist kann beispielsweise die Bezeichnungsalternativen *client* und *patron* als Synonym zum Leitwort *customer* definieren. Das führt im Lexikon des Attempto Systems zu bedeu-

tungsidentischen Einträgen unter dem Leitwort *customer*. In ACE ist nur strikte Synonymie zulässig, so dass in den Anforderungsspezifikationen immer eine bedingungslose Substituierbarkeit der Bezeichnungsalternativen durch das jeweilige Leitwort garantiert ist.

Abkürzungen bilden in ACE einen Spezialfall von Synonymen. Abkürzungen sind Kurzwortbildungen und entstehen aus Leitwörtern durch Reduktion der geschriebenen Form. In Anforderungsspezifikationen finden sich viele technische Ausdrücke, die aus zusammengesetzten Wörtern (z.B. *random access memory*) bestehen. Es ist sehr ermüdend diese komplexen Ausdrücke jedesmal auszuschreiben. Abkürzungen können Abhilfe schaffen und bilden eine ökonomische Alternative, solange die durchgehende Substituierbarkeit der Abkürzungen durch die entsprechenden Leitwörter im Spezifikationstext gesichert ist. Beim Schreiben der Anforderungsspezifikationen werden Abkürzungen in der Regel in Variation mit den Leitwörtern gebraucht. Um die Äquivalenzrelation zwischen einem Leitwort und einer Abkürzung während des Spezifikationsprozesses verfügbar zu haben, ordnet der Anwendungsspezialist bereits beim Lexikonaufbau die Abkürzung (z.B. *RAM*) dem Leitwort (z.B. *random access memory*) zu.

Technisch werden die Synonyme und Abkürzungen im Attempto System während der DRS-Konstruktion durch temporäre DRS-Bedingungen der Form `synonymy(Leitwort(I),Alternative(I))` dargestellt. Der DRS-Konstruktionsalgorithmus löst diese temporären Bedingungen auf, indem das alternative Wort durch das Leitwort substituiert wird. Diese Substitutionen werden dem Anwendungsspezialisten in einer Paraphrase in der Sprache ACE angezeigt.

Die Anwendungsspezialisten können jedem Inhaltswort eine beliebige Anzahl von Synonymen und Abkürzungen zuordnen. Da die Funktionswörter die grammatische Struktur der Sprache ACE stärker bestimmen als die anwendungsspezifischen Inhaltswörter und die meisten Funktionswörter keine oder nur wenige Synonyme und Abkürzungen kennen, dürfen diffizile Zuordnungen dieser Art nur von Experten vorgenommen werden.

Kommentare

Jedes Wort der Sprache ACE kann im Lexikon des Attempto Systems mit einem Kommentartext versehen werden, um einen Begriff durch eine informelle Erkennungsregel einzuführen. Erkennungsregeln helfen Entitäten aus einem Anwendungsbereich zu identifizieren und erleichtern die Verständigung über die verwendeten Begriffe unter den Anwendungsspezialisten. Beispielsweise stellt der Kommentartext *'A staff user X is a person who works at the issue desk in the library and has the permission to lend books to a borrower.'* eine informelle Erkennungsregel für das Nomen *staff user* dar. Die Anwendungsspezialisten können solche Kommentartexte bzw. Erkennungsregeln bei der Lexikalisierung der Inhaltswörter formulieren. Für die Funktionswörter muss diese Aufgabe von den linguistischen Experten übernommen werden.

5.4.1 Inhaltswörter

Zur Klasse der Inhaltswörter gehören in ACE die vier Kategorien: *Nomen*, *Verb*, *Adjektiv* und *Adverb*. Die Wörter dieser Kategorien bilden lexikalische Köpfe, die zum Teil obligatorische Komplemente selektionieren und dann als Phrasen syntaktisch verwendungsfähig sind.

Inhaltswörter

Kategorie	Beispiel
Nomen	book, staff user, LibDB
Verb	return, check out, give
Adjektiv	valid, simple, high, expensive
Adverb	slowly, manually

Die Bedeutung von Inhaltswörtern kann sich während der Zeit in einem Anwendungsbereich durch Generalisierung oder Spezialisierung verändern, was zu einer andersartigen Strukturierung und Segmentierung des Anwendungsbereiches führen kann. Durch morphologische Verfahren (Komposition, Ableitung, Kürzung), Bedeutungsübertragung oder Entlehnung aus anderen Sprachen können neue Wörter oder ganze Phrasen gebildet werden, die neue fachspezifische Konzepte aus einem Anwendungsbereich "speichern". Aus diesen Gründen ist es nicht sinnvoll, die Inhaltswörter für den Spezifikationsprozess im voraus unbesehen zur Verfügung zu stellen, sondern die Anwendungsspezialisten erarbeiten das Vokabular inkrementell oder verwenden allenfalls ein bereichspezifisches Lexikon.

5.4.1.1 Nomen

Nomen bilden lexikalische Köpfe von Nominalphrasen, die in ACE als Subjekt oder Komplement funktionieren und belebte oder unbelebte Entitäten, zeitliche Entitäten oder abstrakte Entitäten denotieren. Die nominalen Köpfe können in kontrollierter

Weise modifiziert werden. Unter morphologischem Aspekt, d.h. von der Flexion her, sind die Nomen durch das Merkmal Numerus gekennzeichnet. In ACE werden die Nomen neben dem Numerus auch nach ihrem natürlichen Geschlecht (= Gender) und ihrem semantischen Typ kategorisiert.

Numerus
Das Merkmal Numerus gibt an, ob mit einer Nominalphrase auf eine oder mehr Entitäten im Anwendungsbereich referiert wird. Das Nomen steht im Singular (`sing`), wenn nur auf eine Entität referiert wird, und im Plural (`plur`), wenn auf mehrere Entitäten referiert wird. Wenn ein Nomen in der Subjektposition steht, dann wird der Numerus durch syntaktische Kongruenz auf das finite Verb übertragen. In der weiteren Diskussion steht insbesondere die Teilkategorie des Singulars im Zentrum.

Gender
Nur sehr wenige Nomen zeigen durch ihre Endungen an, zu welchem grammatischen Geschlecht sie gehören (z.B. *actor/actress*). Besser lässt sich das grammatische Geschlecht an einer Reihe von Pronomen erkennen, die unterschiedliche Formen aufweisen und zu den Nomen in anaphorischer Kongruenz stehen. In ACE sind das die Personalpronomen (*he, she, it,* usw.) und die Relativpronomen (*who, which*). Da die meisten Nomen kein grammatisches Geschlecht anzeigen, muss für den korrekten anaphorischen Bezug das natürliche Geschlecht der Nomen berücksichtigt werden. Aus diesem Grund werden die Nomen in ACE gemäss ihrer Bindung zum natürlichen Geschlecht kategorisiert. Diese Bindung wird in ACE als *Gender* bezeichnet. Das Gender des Nomens ist entweder männlich (`masc`), weiblich (`fem`) oder sächlich (`neut`). Nomen, die von Ereignissen abgeleitet sind (z.B. *borrower*) oder einen Beruf bezeichnen (z.B. *librarian*), lassen sich gleichermassen auf männliche und weibliche Personen anwenden. Diesem Umstand wird in ACE durch den generischen Merkmalwert (*masc/fem*) Rechnung getragen.

Typ
Jedem Nomen wird in ACE ein semantischer Typ (`person`, `time`, `object`) zugeordnet. Diese Typen tragen dazu bei, die Art der Modifikation von Eventualitäten eindeutig zu bestimmen. Nominalphrasen mit einem Nomen vom Typ `person` denotieren Entitäten, die belebt sind (z.B. *John, borrower, staff user*). Nominalphrasen mit einem Nomen vom Typ `time` denotieren zeitliche Entitäten (z.B. *Monday, morning, second*). Nominalphrasen mit einem Nomen vom Typ `object` denotieren Entitäten, die entweder konkret (z.B. *book*) oder abstrakt (z.B. *algorithm*) sind. Wenn durch das

Hinzufügen einer Präpositionalphrase an eine verbale Konstituente die zugrundeliegende Eventualität modifiziert wird, dann kann aus der Präposition und dem semantischen Typ des Nomens auf die Art der Modifikation geschlossen werden. Bei der Übersetzung in die DRT-E ergibt sich beispielsweise für den Fall der adjungierten Präpositionalphrase *in the library* aufgrund der Präposition und des semantischen Typs `object` für das Nomen der lokale Modifikator `location(E,in(E)), library(I)`. Andererseits entsteht aus der adjungierten Präpositionalphrase *in the morning* mit derselben Präposition und dem nominalen semantischen Typ `time` der temporale Modifikator `time(E,in(I)), morning(I)`. Die Modifikatoren sind in ACE eindeutig mit Frageausdrücken assoziiert, so dass bei der Validierung der Spezifikation beispielsweise der Frageausdruck *where* immer auf den Modifikatortyp `location` und der Frageausdruck *when* auf den Modifikatortyp `time` bezogen werden kann.

Die Nomen werden in ACE in die drei nominalen Kategorien *Eigenname, Gattungsbezeichnung* und *Quoted String* unterteilt, und zwar aufgrund ihrer unterschiedlichen Form der Bezugsnahme auf Entitäten im Anwendungsbereich.

Eigenname
Eigennamen sind als lexikalische Köpfe von Nominalphrasen unmittelbar syntaktisch verwendungsfähig und brauchen nicht ergänzt zu werden. Eigennamen benennen Entitäten nicht dadurch, dass sie begriffliche Information liefern, sondern indem sie von der konventionalisierten Relation, die zwischen einem Namen und seinem Träger besteht, Gebrauch machen. Für das Verständnis von Eigennamen ist das gemeinsame Wissen unter den Anwendungsspezialisten oft nicht so selbstverständlich vorauszusetzen, wie das für andere Inhaltswörter der Fall ist. Durch die Aufnahme von informellen Erkennungsregeln für Eigennamen in das Lexikon ist es möglich, die korrekte Identifikation der Entitäten im Anwendungsbereich zu unterstützen.

Eigennamen werden immer gross geschrieben und können in ACE entweder durch appositive Nominalphrasen (z.B. *Meier '19'*) oder durch restriktive Relativsätze (z.B. *John who enters the password ...*) modifiziert werden.

Da derselbe Eigennamen (theoretisch) durchaus unterschiedliche Entitäten identifizieren kann, werden Eigennamen in der DRT-E nicht auf Konstanten abgebildet, sondern durch Bedingungen der Form `named(I,'<wform>')` dargestellt. Dabei ist `I` ein Individuenreferent, der für eine Entität steht, die durch die Konstante `'<wform>'` benannt wird. Solche Bedingungen werden lexikalisiert und sind bei der Verarbeitung

direkt für den Aufbau einer DRS verfügbar.

Eigenname

Merkmal	Wert	Beispiel
Wortform	`'<wform>'`	`Yolande`
Numerus	`sing | plur`	`sing`
Gender	`masc | fem | masc/fem | neut`	`fem`
Typ	`person | time | object`	`person`
Bedingung	`named(I,'<wform>')`	`named(I,'Yolande')`

Gattungsbezeichnung

Gattungsbezeichnungen sind als lexikalische Köpfe von Nominalphrasen in den meisten Fällen nicht unmittelbar verwendungsfähig, sondern sie müssen zuerst durch einen Determinator (Artikel oder Quantor) ergänzt werden, damit eine korrekte Denotation zustande kommt.

Die Gattungsbezeichnungen werden in ACE in zwei Gruppen unterteilt: in Individualnomen (z.B. *staff_user*) und in Massennomen (z.B. *copper, data*). Individualnomen weisen Numerus-Distinktion auf, sind zählbar und müssen in ACE immer zusammen mit einem Determinator als Spezifikator verwendet werden. Eine Nominalphrase mit einem Individualnomen denotiert im Anwendungsbereich entweder eine einzelne Entität aus einer Menge von gleichgearteten Entitäten (*a/the staff user*) oder alle gleichgearteten Entitäten der Menge (z.B. *every staff user*). Im Gegensatz zu Individualnomen weisen Massennomen keine Numerus-Distinktion auf, sind nicht zählbar und werden in der Regel ohne die universellen Determinatoren (*each, every, for every*) und dem indefiniten Determinator (*a*) verwendet. Massennomen denotieren entweder eine aus einer Menge von Elementen (untrennbar) zusammenhängende Entität (z.B. *copper*) oder eine aus einer Menge von Elementen zusammengefasste Entität (z.B. *data*). Die Gruppenzugehörigkeit der Gattungsbezeichnungen ist nicht immer eindeutig. Es ist deshalb die Aufgabe der Anwendungsspezialisten die Gattungsbezeichnungen gemäss der intendierten Gebrauchsweise als Individualnomen (`count`) oder Massennomen (`mass`) zu kategorisieren.

Gattungsbezeichnungen können in ACE durch vorangestellte possessive Determinatoren (z.B. *John's password*), durch attributive Adjektivphrasen (z.B. *the correct password*), durch appositive Nominalphrasen oder Zeichenketten (z.B. *the borrower John*

Smith, the message 'Invalid Number'), durch eine Teilmenge von Präpositionalphrasen (z.B. *a copy of the book*) oder durch restriktive Relativsätze (z.B. *the password that the staff user enters ...*) ergänzt oder modifiziert werden.

Nominalphrasen, die eine Gattungsbezeichnung als lexikalischen Kopf haben, führen eine einfache Grundbedingung der Form `<wform>(I)` mit einem Individuenreferenten `I` in die DRT-E ein. Das Prädikatssymbol `<wform>` der Grundbedingung wird (automatisch) aus der Wortform der Gattungsbezeichnung abgeleitet.

Gattungsbezeichnung

Merkmal	Wert	Beispiel
Wortform	`<wform>`	`staff user`
Numerus	`sing | plur`	`sing`
Gender	`masc | fem | masc/fem | neut`	`masc`
Typ	`person | time | object`	`person`
Gruppe	`count | mass`	`count`
Bedingung	`<wform>(I)`	`staff_user(I)`

Quoted String

Die nominale Kategorie *Quoted String* umfasst beliebige Zeichenketten, die durch einfache Anführungszeichen gekennzeichnet sind und einem Eigennamen oder einer Gattungsbezeichnung als Apposition nachgestellt sind (z.B. *Meier '19', the message 'Invalid Number'*). Die Quoted Strings tragen keine syntaktischen Merkmale und dienen in ACE ausschliesslich zur zusätzlichen Benennung oder Hervorhebung bestimmter Eigenschaften für bereits eingeführte Individuenreferenten. Quoted Strings sind nie selbständig verwendungsfähig. Im Gegensatz zu Eigennamen können sie nicht selbständig zum identifizierenden Bezugnehmen auf Entitäten gebraucht werden.

Die Übersetzung von Quoted Strings in die DRT-E führt zu einer vordefinierten Bedingung der Form $\texttt{string}(I_{i+1}, \texttt{'<wform>'})$ und einer (ersetzbaren) Identitätsbedingung $I_{i+1} = I_i$, die den Bezug zum nominalen Kopf herstellt.

Quoted String

Merkmal	Wert	Beispiel
Wortform	`'<wform>'`	`'Invalid Number'`

Merkmal	Wert	Beispiel
Bedingung	string(I_{i+1},'<wform>') I_{i+1} = I_i	string(I2,'Invalid Number') I2 = I1

5.4.1.2 Verb

Verben bilden lexikalische Köpfe von Verbalphrasen und selektionieren eine bestimmte Anzahl von Komplementen. Dadurch wird eine ein- oder mehrstellige Relation zwischen Entitäten hergestellt. Ein Satz ist dann vollständig, wenn alle Stellen dieser Relation besetzt sind. Verben haben ein komplexes Merkmalsystem von Formen und Funktionen und bestimmen kraft ihres Selektionsrahmens die Struktur des Satzes. Von der Form her lassen sich sieben Kategorien unterscheiden: Modus, Modalität, Tempus, Aspekt, Genus Verbi, Person und Numerus. Nicht alle diese Kategorien sind in der Sprache ACE realisiert, und von den realisierten Kategorien ist nur eine Teilmenge der möglichen formalen Ausprägungen zulässig.

Modus

Modus ist eine grammatische Kategorie des Verbs, durch die bestimmte subjektive Spechereinstellungen zu einem bezeichneten Sachverhalt ausgedrückt werden. Man unterscheidet dabei die Teilkategorien Indikativ, Konjunktiv und Imperativ, die morphologisch durch entsprechende Formen gekennzeichnet sind. Der Indikativ ist die neutrale Kategorie und wird dazu verwendet, um einen Sachverhalt als real gegeben darzustellen. Der Konjunktiv kennzeichnet einen Sachverhalt entweder als erwünscht oder irreal. Der Imperativ dient als Modus der Aufforderung. Nur mit dem Indikativ können faktische Aussagen gemacht und Sachverhalte als gegeben dargestellt werden. Die anderen beiden Modi drücken unterschiedliche Arten von Modalität aus und relativieren die durch die Aussagen gemachten Sachverhalte. In ACE ist nur der Indikativ zulässig. Mit dem Indikativ werden Sachverhalte in Aussagesätzen als real oder in konditionalen Sätzen als real möglich dargestellt. Ausser den Verben in den Aussagesätzen und den konditionalen Sätzen stehen auch die Verben in den Fragesätzen von ACE im Indikativ.

Modalität

Modalität kann bei Verben nicht nur durch morphologische Formen angezeigt werden, sondern auch mit Hilfe von lexikalischen Mitteln, und zwar durch modale Hilfsverben (*might, may, could, can, should, ought to, would, will, must*) in Verbindung mit einem reinen Infinitiv. Viele Sätze in Spezifikationstexten, die modale Hilfsverben enthalten, lassen mehr als eine mögliche Interpretation zu, da die modalen Hilfsverben in

der Regel Bezüge zu unterschiedlichen Hintergrundsbereichen herstellen. Je nach Kontext bezieht sich beispielsweise das modale Hilfsverb *can* in einem Satz wie

(97) *The staff user can enter the password.*

auf das, was der Anwendungsspezialist aufgrund seines Wissens über den Sachverhalt für möglich hält (epistemische Lesart), auf das, was dem Mitarbeiter erlaubt ist (deontische Lesart), oder auf das, wozu der Mitarbeiter fähig ist (dispositionelle Lesart). Modale Hilfsverben sind in ACE nicht zulässig, da sie die Faktizität der Aussagen relativieren, indem sie entweder das Verhältnis des Anwendungsspezialisten zur Aussage thematisieren oder ein mögliches Verhältnis zwischen dem Subjekt und Prädikat in der Aussage zum Ausdruck bringen, ohne dass sich eine eindeutige Abgrenzung der Lesarten (an der Satzoberfläche) erkennen lässt.

Tempus

Tempus ist eine grammatische Kategorie, die formal durch Verbflexion realisiert wird. Genaugenommen hat die englische Sprache nur zwei Tempuskategorien: Präsens und Präteritum. Diese beiden Kategorien sind Teil eines komplexen Systems, durch das Aussagen auf der Zeitachse lokalisiert werden. Die tempusmarkierten Verbformen können einfach (z.B. *returns, returned*) oder zusammengesetzt (z.B. *had returned, has returned, will return*) sein. Das einfache Präsens drückt (typischerweise) einen Zeitbezug aus, bei dem der Sprechzeitpunkt und der Ereigniszeitpunkt auf der Zeitachse zusammenfallen. Das einfache Präteritum stellt (typischerweise) einen Zeitbezug her, bei dem der Ereigniszeitpunkt vor dem Sprechzeitpunkt liegt. Neben diesen beiden Grundbedeutungen gibt es zahlreiche Verwendungsbedeutungen für die einfachen Tempusformen, die erst unter Berücksichtigung von anderen modifizierenden temporalen Konstruktionen (z.B. Adverbialphrasen, Präpositionalphrasen, Nebensätze) und unter Einbezug des aussersprachlichen Kontexts korrekt interpretiert werden können. Ausser den einfachen tempusmarkierten Verbformen können auch zusammengesetzte Verbformen dazu verwendet werden, um zeitliche Bezüge zur Vergangenheit, Gegenwart oder Zukunft herzustellen. Die zusammengesetzten Verbformen stehen in enger Beziehung zum Aspekt- und Modalsystem und beziehen sich nicht nur auf rein temporale Verhältnisse, sondern bezeichnen auch aspektuelle und modale Inhalte. Da die einzelnen Tempusformen oft nur lose mit Zeitpunkten verbunden sind und sich ihre Geltungsbereiche überlappen, müssen die grammatischen Zeiten in Spezifikationstexten in einer kontrollierten Art und Weise gebraucht werden. In der Sprache ACE ist nur das einfache Präsens (*simple present tense*) zulässig. Der Anwendungsspezialist kann sich mit dem Präsens auf Sachverhalte beziehen, die zum gegenwär-

tigen Zeitpunkt wahr sind oder unter genau festgelegten Vorbedingungen wahr sind. Zusätzlich zu diesen beiden Hauptverwendungsarten steht dem Anwendungsspezialist das Präsens in Fragesätzen zur Verfügung, um sich die Existenz von spezifizierten Sachverhalten oder teilhabenden Entitäten nachweisen zu lassen. Durch die Einschränkungen auf das Präsens im Spezifikationstext für die Beschreibung von wahren oder hypothetischen Sachverhalten wird erreicht, dass der Anwendungsspezialist den Text als eine lineare Abfolge von Instruktionen organisiert, so dass eine chronologische Anordnung der zugrundeliegenden Eventualitäten auf der Zeitachse entsteht. Diese Vorgehensweise verhindert komplexe temporale Bezüge im Spezifikationstext.

Aspekt

Aspekt ist eine grammatische Kategorie, die die Einstellung des Sprechers über den zeitlichen Verlauf von Eventualitäten durch morphosyntaktische Markierung am Verb zum Ausdruck bringt. Die englische Sprache hat zwei Aspektkategorien: Progressiv und Perfekt. Der progressive Aspekt kommt durch eine Form des Hilfsverbs *be* zusammen mit einem nachgestellten Partizip Präsens (z.B. *returning*) zustande und zeigt (typischerweise) an, dass ein Ereignis zu einem Referenzzeitpunkt andauert. Der perfektive Aspekt wird durch eine Form des Hilfsverbs *have* zusammen mit einem nachgestellten Partizip Perfekt (z.B. *returned*) angezeigt und stellt (typischerweise) ein Ereignis oder einen Zustand als zu einem Referenzzeitpunkt abgeschlossen und nachwirkend dar. Die Formen der beiden Aspektkategorien sind leicht erkennbar, aber die korrekte Interpretation ist äusserst schwierig, da die einzelnen Konstruktionen in Abhängigkeit von der Aktionsart der beteiligten Verben verschiedene Bedeutungsnuancierungen des zeitlichen Verlaufs kodieren. Im Gegensatz zum Aspekt bezeichnet die Aktionsart kein grammatisches Phänomen, sondern sie teilt die Verben hinsichtlich ihrer temporalen und inhaltlichen Struktur, die sie zum Ausdruck bringen, in unterschiedliche lexikalisch-semantische Kategorien ein (z.B. durative Verben, resultative Verben, iterative Verben, usw.). In ACE wird weder das grammatische Aspektsystem der englischen Sprache berücksichtigt, noch spielen inhärente lexikalisch-semantische Kriterien der Verben für die Bestimmung der Aktionsart eine zentrale Rolle. Anstelle davon werden Eventualitäten - also nicht-linguistische Entitäten - entweder als Ereignisse oder Zustände kategorisert. Ereignisse sind Entitäten, die als Zustandsübergänge aufgefasst werden können und eine endliche Laufzeit besitzen. Die Struktur der Zeit ist implizit und ergibt sich als Abstraktion aus der Abfolge der Ereignisse. Zustände sind zeitlich limitierbare Eigenschaften von Entitäten oder Beziehungen zwischen Entitäten, die möglicherweise aus Ereignissen resultieren und keine Veränderungen implizieren. Der Anwendungsspezialist unterscheidet bei der Lexikalisierung zwischen Ereignissen und Zuständen im Anwendungsbereich und ordnet diese (wahrnehmbaren) nicht-linguistischen Entitäten den einzelnen Verben beziehungsweise Verbalphrasen zu. Da Verben nicht inhärent dynamisch oder statisch sind, sondern in den Sätzen eher dynamisch oder statisch verwendet werden, kann der Anwendungsspezialist bei der Lexikalisierung durch die Zuordnung von Ereignissen oder Zuständen zu Verben die angemessene Verwendungsweise spezifizieren.

Genus Verbi

Das Genus Verbi ist eine grammatische Kategorie, die zwischen Aktiv und Passiv unterscheidet. Unter Aktiv und Passiv sind Sichtweisen zu verstehen, die nicht primär von der Verbbedeutung abhängig sind, sondern eine Beziehung zwischen dem Subjekt und dem Verb ausdrücken. Beim Aktiv besteht eine besonders enge Beziehung zwischen dem Subjekt und dem Verb. Sie macht den Träger des Sachverhaltes, der durch das Subjekt kodiert wird, zum Ausgangspunkt der Aussage. Beim Passiv tritt diese Beziehung in den Hintergrund. Anstelle des logischen Subjekts rückt eine trägerabgewandte Nominalphrase in die Subjektposition. Das führt dazu, dass das logische Subjekt entweder ganz fehlt oder durch eine optionale Präpositionalphrase ausgedrückt werden muss. Das Passiv kann (typischerweise) nur zusammen mit transitiven Verben verwendet weden. In ACE ist nur die aktive Form von Verben zulässig, da dadurch die Träger des verbalen Geschehens immer zuverlässig in der Subjektposition angezeigt werden. Hinzu kommt, dass das Aktiv im Gegensatz zum Passiv auf alle Verben anwendbar ist.

Person und Numerus

Person und Numerus sind grammatische Kategorien, die primär nominal und nicht verbal sind. Sie werden in erster Linie auf Nominalphrasen (in der Subjektposition) angewendet und kongruieren dann mit den finiten Verbformen, so dass das Subjekt mit dem Verb in Person und Numerus übereinstimmt. Ausser dem Verb *be* sind nur Verben, die im Präsens stehen, durch die Merkmale Numerus und Person markiert. Die dritte Person Singular wird (in Abhängigkeit von den phonologischen Eigenschaften des Stammes) durch Hinzufügen eines Suffixes (*-s,-es*) an den Stamm gebildet. Für die restlichen formalen Ausprägungen wird die Grundform verwendet. Da in Spezifikationstexten weder der Schreiber (erste Person) noch der Adressat (zweite Person) relevant sind, können in ACE nur Sätze verwendet werden, die in der dritten Person Singular (oder Plural) stehen.

Verbformen im Überblick

In ACE stehen die Verben in der dritten Person Singular (oder Plural) und haben den Indikativ als Modus, das Aktiv als Genus Verbi und das einfache Präsens als Tempusform. Verben mit diesen formalen Ausprägungen können von den Anwendungsspezialisten dazu verwendet werden, um Aussagen zu machen, die Sachverhalte beschreiben, welche zum gegenwärtigen Zeitpunkt wahr sind oder unter bestimmten Vorbedingungen wahr sind. Der Gebrauch des einfachen Präsens in dieser festgeschriebenen Art und Weise garantiert eine Gleichzeitigkeit des Spechzeitpunktes mit dem in der Aussage beschrieben Sachverhaltes. Darüber hinaus bewirkt die Verwendung der aktiven Verbform, dass der Anwendungsspezialist den Träger des verbalen Geschehens explizit machen muss. Durch diese Einschränkungen wird erreicht, dass der Anwendungsspezialist den Spezifikationstext als eine Abfolge von Instruktionen organisiert, wodurch implizit eine chronologische Anordnung der Ereignisse und Zustände auf der Zeitachse hergestellt wird. Dies ist ein Gewinn, wenn wir uns daran erinnern, dass die Tempusformen im Englischen oft nur lose mit der Zeit übereinstimmen und zum Teil mit dem Aspekt- und Modalsystem verzahnt sind.

Verbformen im Überblick

Kategorie	Verben im Englischen	Verben in ACE
Modus	Indikativ, Konjunktiv, Imperativ	Indikativ
Modalität	Modale Hilfsverben	—
Tempus	Präsens, Präteritum	Präsens
Aspekt	Progressiv, Perfekt	—
Genus Verbi	Aktiv, Passiv	Aktiv
Person	erste, zweite, dritte	dritte
Numerus	Singular, Plural	Singular, (Plural)

Besonderheiten

Eine Teilmenge der Verben treten in Kombination mit einer Präposition (z.B. *call for, add to*) oder einem Adverb (z.B. *leave out, bring back*) auf. Verben, die in Verbindung mit einer bestimmten Präposition vorkommen, heissen präpositionale Verben. Die teilhabenden Präpositionen sind syntaktisch relativ selbständig, denn sie funktionieren als lexikalische Köpfe von Präpositionalphrasen und selektionieren Nominalphrasen als Komplemente. Verben, die in fester Verbindung mit einem Adverb stehen, werden phrasale Verben genannt. Im Gegensatz zu Präpositionen können Adverbien von phrasalen Verben keine Komplemente selektionieren, dafür ist das ganze phrasale

Verb zuständig. Die meisten Adverbien von phrasalen Verben sind Lokaladverbien und lassen sich von der Form her nicht von Präpositionen (z.B. *out, back*) unterscheiden. Präpositionale und phrasale Verben scheinen auf den ersten Blick sehr ähnlich, doch heben sie sich durch die Stellung ihrer Präpositionen und Adverbien im Satz voneinander ab. Präpositionen stehen in ACE immer unmittelbar vor der selektionierten Nominalphrasen, z.B. in

(98) *The borrower calls for a registration form.*

Adverbien von phrasalen Verben sind hingegen in der linearen Anordnung nicht kontinuierlich und können einer Nominalphrase manchmal vorangestellt werden, z.B. in

(99) *The borrower fills out the registration form.*

und manchmal nachgestellt werden, wie z.B. in

(100) *The borrower brings the registration form back.*

Dementsprechend muss der Anwendungsspezialist bei der Lexikalisierung präpositionale Verben wie in (98) von phrasalen Verben wie in (99) oder (100) unterscheiden. Um Adverbien, die einen festen Bestandteil eines phrasalen Verbs bilden, in der weiteren Diskussion sprachlich von verbmodifizierenden Adverbien unterscheiden zu können, werden die phrasal verwendeten Adverbien als adverbiale Partikel bezeichnet.

Eine andere Teilmenge der Verben erzeugt einen intensionalen Kontext. Sogenannte intensionale Verben (z.B. *assume, believe, know*) sind in ACE nicht zulässig, da sie Komplemente einbetten, die mehrdeutig sind. Beispielsweise kann die Nominalphrase *the winner of the Pulitzer Prize* im Satz

(101) *The staff user believes that the winner of the Pulitzer Prize is in the catalogue.*

entweder extensional interpretiert werden, wenn sie dazu dient, einen bestimmten Gewinner des Pulitzer Preises zu identifizieren (= referentielle Lesart), oder aber intensional, wenn nicht ein bestimmter Gewinner im Mittelpunkt steht, sondern das Konzept angesprochen wird, welches auf den deskriptiven Inhalt der Nominalphrase zutrifft (= attributive Lesart). Ausserdem zeigt sich, dass bei intensionalen Verben die Ersetzbarkeit von extensionsgleichen Nominalphrasen nicht oder nur bedingt zulässig ist - eine Eigenschaft, die bei einem syntaxbasierten Ansatz, wie er der Sprache ACE zugrunde liegt, nicht erwünscht ist [vgl. Lyons 95:230].

ACE Verben werden im Lexikon nicht einfach als Relationen dargestellt, die eine Beziehung zwischen einem externen Komplement (Subjekt) und einer Anzahl von internen Komplementen herstellen, sondern in erster Linie als Relationen zwischen Variablen für Eventualitäten und Prädikaten, die diese Eventualitäten benennen [vgl. Alshawi 92:22]. Durch die Einführung von quantifizierten Ereignis- oder Zustandsvariablen als zusätzliche Argumente wird es in ACE möglich, optionale Modifikatoren (in Form von adjungierten Adverbial- und Präpositionalphrasen) als Aussagen über Eventualitäten zu behandeln. Es ist die Aufgabe der Anwendungsspezialisten, die Verben aufgrund ihres Selektionsrahmens als intransitiv, transitiv oder ditransitiv zu kategorisieren und festzulegen, ob sie im Anwendungsbereich dynamisch oder statisch verwendet werden, d.h. festzulegen, ob sie Ereignisse oder Zustände denotieren.

Intransitives Verb

Intransitive Verben sind einstellig, das heisst sie benötigen nur ein externes Komplement, das in ACE durch eine Nominalphrase (`n2`) in der Subjektposition realisiert wird. Das Subjekt bzw. das externe Komplement stimmt mit dem Verb in Numerus (`sing` oder `plur`) und Person überein. Falls an der Subjektposition anstelle eines Nomens (`n0`) ein Personalpronomen steht, wird sichtbar, dass das Verb dem externen Komplement den Nominativ (`nom`) als Kasus zuweist. Da die Verben in ACE ausschliesslich in der dritten Person, im Indikativ und in der aktiven Form verwendet werden dürfen, müssen diese drei Merkmale nicht lexikalisiert werden. Eine Unterscheidung muss jedoch bei den Verbformen getroffen werden, da es sich beim Verb im Fall von positiven Aussagesätzen und konditionalen Sätzen entweder um eine finite Verbform (`fin`) oder im Fall von negativen Sätzen oder Fragesätzen um eine Grundform (`base`) handeln kann. Die Übersetzung von intransitiven Verben führt in der DRT-E zu einer Bedingung der Form `event(E,<bform>(I))` oder `state(S,<bform>(I))`, wobei `E` eine Ereignisvariable ist und `S` eine Zustandsvariable. Das Prädikat `<bform>(I)` benennt eine potentielle Relation, die für den Individuenreferenten `I` gilt. Für die Darstellung des Prädikatsnamen `<bform>` wird die Grundform des verwendeten Verbs gewählt. Für phrasale intransitive Verben kommt der Prädikatsnamen durch Konkatenation der Grundform mit der adverbialen Partikel zustande (z.B. *hold on* -> `hold_on`).

Intransitives Verb

Merkmal	Wert	Beispiel
Wortform	`<wform>`	`arrives`
Verbform	`fin | base`	`fin`

Externes Komplement
Kategorie
Numerus
Kasus
	`n2`
	`n0`
	`sing | plur`
	`nom`
		`n2`
		`n0`
		`sing`
		`nom`

Bedingung	`event(E,<bform>(I)) |` `state(S,<bform>(I))`	`event(E,arrive(I))`

Transitives Verb

Transitive Verben sind zweistellig, das heisst sie benötigen neben dem externen noch ein internes Komplement, das in ACE durch eine Nominalphrase (`n2`) oder eine Präpositionalphrase (`p2`) realisiert wird. Der Anwendungsspezialist muss bei der Lexikalisierung der transitiven Verben zwischen normalen, phrasalen und präpositionalen Verben unterscheiden. Die normalen transitiven Verben selektionieren in ACE eine Nominalphrase (`n2`) als internes Komplement und weisen ihm den Akkusativ (`acc`) als Kasus zu, wobei die Kasusmarkierung nur bei den Personalpronomen sichtbar wird. Die phrasalen transitiven Verben selektionieren ebenfalls eine Nominalphrase (`n2`) als Komplement. Bei phrasalen Verben muss die Form der adverbialen Partikel im Lexikon verzeichnet sein. Im Gegensatz zu normalen und phrasalen Verben selektionieren die präpositionalen transitiven Verben eine Präpositionalphrase (`p2`) als Komplement. Auch die Form der Präposition von präpositionalen Verben muss im Lexikon verzeichnet sein. Die Übersetzung von transitiven Verben führt in der DRT-E zu einer Bedingung der Form `event(E,<bform>(I1,I2))` oder `state(S,<bform>(I1,I2))`, wobei `E` eine Ereignisvariable ist, `S` eine Zustandsvariable und das Prädikat `<bform>(I1,I2)` eine mögliche Relation benennt, die zwischen den Individuenreferenten I_1 und I_2 gilt. Die Darstellung des Prädikatsnamen `<bform>` kommt bei phrasalen und präpositionalen Verben durch Konkatenation der Grundform des Verbs mit der Präposition oder der adverbialen Partikel zustande (z.B. *bring back* -> `bring_back` oder *ask for* -> `ask_for`).

Normales transitives Verb

Merkmal	Wert	Beispiel
Wortform	<wform>	has
Verbform	fin | base	fin
Externes Komplement Kategorie Numerus Kasus	n2 n0 sing | plur nom	n2 n0 sing nom
Komplement Kategorie Kasus	n2 n0 acc	n2 n0 acc
Bedingung	event(E,<bform>(I_1,I_2)) | state(S,<bform>(I_1,I_2))	state(S,have(I1,I2))

Phrasales transitives Verb

Merkmal	Wert	Beispiel
Wortform	<wform>	brings
Verbform	fin | base	fin
Externes Komplement Kategorie Numerus Kasus	n2 n0 sing | plur nom	n2 n0 sing nom
Komplement Kategorie Partform	n2 n0 out | up | back | …	n2 n0 back
Bedingung	event(E,<bform>(I_1,I_2)) | state(S,<bform>(I_1,I_2))	event(E,bring_back(I1,I2))

Präpositionales transitives Verb

Merkmal	Wert	Beispiel
Wortform	<wform>	asks
Verbform	fin | base	fin
Externes Komplement Kategorie Numerus Kasus	n2 n0 sing | plur nom	n2 n0 sing nom
Komplement Kategorie Prepform	p2 p0 on | to | for | …	p2 p0 for

Merkmal	Wert	Beispiel
Bedingung	event(E,<bform>(I_1,I_2)) | state(S,<bform>(I_1,I_2))	event(E,ask_for(I1,I2))

Ditransitives Verb

Ditransitive Verben sind dreistellig (z.B. *give, make*), das heisst sie selektionieren neben dem externen Komplement zwei interne Komplemente. Das erste interne Komplement wird durch eine Nominalphrase (*n2*) realisiert. Das zweite interne Komplement ist entweder eine Nominalphrase (*n2*) oder eine Präpositionalphrase (*p2*). Ditransitive Verben weisen den internen Komplementen den Akkusativ (*acc*) als Kasus zu, wenn sie durch Nominalphrasen realisiert werden, wobei die Kasusmarkierung wiederum nur bei den Personalpronomen sichtbar wird. Falls eine Präpositionalphrase als zweites Komplement selektioniert wird, dann handelt es sich beim präpositionalen Kopf um eine der präpositionalen Formen *to, for* oder *with* (z.B. *send to, design for, mix with*). Viele ditransitive Verben, die zwei Nominalphrasen als Komplemente nehmen, können auch durch ein präpositionales ditransitives Verb ausgedrückt werden. Dabei vertauschen das erste und das zweite Komplement ihre phrasalen Inhalte. Beispielsweise selektioniert das ditransitive Verb *gives* im Satz

(102) *The staff user gives the borrower the book.*

zwei Nominalphrasen als interne Komplemente. Hingegen realisiert das präpositionale ditransitive Verb *give to* im Satz

(103) *The staff user gives the book to the borrower.*

eine Nominalphrase (*the book*) als erstes Komplement und eine Präpositionalphrase (*to the borrower*) als zweites Komplement. Obwohl die Komplementpositionen unterschiedlich besetzt sind, ist die Bedeutung der Sätze (102) und (103) identisch. Diese Beziehung wird bei der Verarbeitung der Sprache ACE nicht automatisch hergestellt, weil das zu einer impliziten Transformation im Lexikon führen würde, die für den Anwendungsspezialisten möglicherweise nur schwer nachvollziehbar ist. Wenn der Anwendungsspezialist die beiden Sätze (102) und (103) aber dennoch zueinander in Bezug setzen möchte, dann muss er das explizit machen, indem er einen konditionalen Satz verwendet:

(104) *If a staff user gives a borrower a book then he gives the book to the borrower.*

Die Übersetzung von ditransitiven Verben führt in der DRT-E zu einer potentiellen Bedingung der Form *event(E,<bform>(I1,I2,I3))* mit einem Ereignisreferenten und drei Individuenreferenten.

Ditransitives Verb

Merkmal	Wert	Beispiel
Wortform	<wform>	gives
Verbform	fin | base	fin
Externes Komplement Kategorie Numerus Kasus	n2 n0 sing | plur nom	n2 n0 sing nom
1. Komplement Kategorie Kasus	n2 n0 acc	n2 n0 acc
2. Komplement Kategorie Prepform	p2 p0 to | for | with	p2 p0 to
Bedingung	event(E,<bform>(I1,I2,I3))	event(E,give_to(I1,I2,I3))

Kopulatives Verb

Ein Sonderfall ist das kopulative Verb *be*, das im Lexikon des Attempto Systems vordefiniert ist. Syntaktisch handelt es sich bei *be* um ein Hilfsverb, das ein externes Komplement und ein sogenanntes prädikatives Komplement zueinander in Beziehung setzt. Prädikative Komplemente sind obligatorisch, haben aber keinen eigentlichen Argumentstatus. Das Subjekt bzw. das externe Komplement wird in ACE durch eine Nominalphrase (*n2*) realisiert und das prädikative Komplement entweder durch eine Nominalphrase (*n2*), durch eine Adjektivphrase (*a2*) oder durch eine Präpositionalphrase (*p2*).

Speziell verhält sich das kopulative Verb *be* auch in Fragesätzen. Im Fall von Entscheidungsfragen werden das Subjekt und die finite Verbform invertiert im Gegensatz zu allen anderen Verben in ACE, die Entscheidungsfragen durch *do*-Einsetzung, Subjekt-Operator-Inversion und verbale Grundformen bilden. Diese Eigenart des kopulativen Verbs wird durch das Merkmal *inv(+)* im Lexikon angezeigt.

Mit Hilfe des kopulativen Verbs *be* können in ACE drei unterschiedliche Arten von Konzepten zum Ausdruck gebracht werden. Erstens kann mit *be* eine Identitätsaussage gemacht werden, und zwar indem eine Identitätsrelation zwischen zwei Entitäten hergestellt wird. Man nennt diese Gebrauchsweise auch "*be* der Identität". Zweitens kann mit *be* eine Attribuierung realisiert werden, und zwar indem einer Entität eine Eigenschaft zugeschrieben wird, so dass ein neuer Zustand in der Spezifikation resultiert. Man spricht bei dieser prädikativen Verwendungsweise von *be* auch von "*be* der Prädikation". Drittens kann mit *be* ein Begriff im Definitionsteil der eigentlichen Spezifikation durch eine logische Gleichung (bzw. Äquivalenz) erklärt und vereinbart werden, und zwar indem ein Ausdruck syntaktisch durch einen oder mehrere andere Ausdrücke ersetzt wird. In ACE wird diese Realisierung als "*be* der Explikation" bezeichnet.

Be der Identität

Im Satz (105) kommt eine Identitätsaussage dadurch zustande, dass ein Eigenname in der Subjektposition durch das kopulative Verb *be* mit einer definiten Nominalphrase in der Komplementposition verbunden wird.

(105) *Hofstadter is the author of the book.*

Satzbaupläne dieser Art, mit einem Eigennamen (*pn*) oder einer definiten Nominalphrase (*'the' n1*) als Subjekt (= externes Komplement), dem kopulativen Verb *be* als Bindeglied und einem Eigennamen oder einer definiten Nominalphrase als Komplement führen bei der Übersetzung in die DRT-E immer zu einer Identitätsbedingung der Form *is(I1,I2)*.

Kopulatives Verb (Identität)

Merkmal	Wert	Beispiel
Wortform	<wform>	is
Verbform	fin	fin
Inversion	inv(+)	inv(+)
Externes Komplement Kategorie Numerus Kasus	n2 { pn | 'the' n1 } n0 sing | plur nom	n2 n0 sing nom
Komplement Kategorie Numerus Kasus	n2 { pn | 'the' n1 } n0 sing | plur nom	n2 n0 sing nom
Bedingung	<wform>(I1,I2)	is(I1,I2)

Die Bedingung `is(I1,I2)` zeigt an, dass es sich bei einer Identitätsaussage um eine assertierte Form der Identität handelt, die einen essentiellen Bestandteil der Aussage bildet. Diese Form der Identität muss in ACE von der sogenannten stipulierten Identität (`I2=I1`) unterschieden werden, die durch anaphorische Referenz zustande kommt und als vereinbarter Bestandteil von zwei im Prinzip unabhängigen Aussagen auftritt [vgl. Kamp & Reyle 93:260]. Beide Formen der Identität werden während der Verarbeitung durch den DRS-Konstruktionsalgorithmus aufgelöst, indem die Individuenreferenten miteinander unifiziert werden. Das Ergebnis der Referenzresolution wird dem Systembenutzer in beiden Fällen in einer Paraphrase angezeigt. Für die Identitätsaussage im Satz (105) ergibt sich die folgende Paraphrase:

(106) *Hofstadter is [identical to] the author of the book.*

Das Diskurssignal *identical to* in eckigen Klammern verdeutlicht, wie das Verb *be* analysiert worden ist. Die Paraphrase selber ist ein zulässiger ACE Satz und kann vom Attempto System verarbeitet werden, wenn zuvor die eckigen Klammern entfernt werden.

Be der Prädikation

Im Unterschied zu Satz (105) wird das kopulative Verb *be* in den folgenden beiden Sätzen nicht mit einem Eigennamen oder einer definiten Nominalphrase in der Komplementposition verwendet, sondern im Satz (107) mit einer Adjektivphrase und im Satz (108) mit einer indefiniten Nominalphrase.

(107) *The book is available.*

(108) *The book is a bestseller.*

In beiden Fällen handelt es sich um ein syntaktisch notwendiges Attribut, welches einer Entität, die in der Subjektposition beschrieben wird, eine Eigenschaft zuschreibt und in der Spezifikation zu einem neuen Zustand führt. Durch die Verwendung des Negationsausdrucks *is not* kann einer Entität eine Eigenschaft wieder abgesprochen bzw. ein bestehender Zustand aufgehoben werden.

Im Fall einer Prädikation trägt das kopulative Verb *be* bei der Übersetzung eine partielle Bedingung der Form `state(S,P)` und eine Identitätsbedingung der Form `I2=I1` bei. `S` steht für eine neue Zustandsvariable und `P` für dasjenige Prädikat, das aus dem prädikativen Komplement abgeleitet werden kann. Für die beiden Beispielsätze sind das die Prädikate `available(X)` bzw. `bestseller(I2)`. Daraus ergeben sich

durch Unifikation die beiden Zustände `state(S,available(X))` und `state(S,bestseller(I2))`. Im Gegensatz zur indefiniten Nominalphrase führt die Adjektivphrase mit dem einstelligen Adjektiv selber keinen neuen Individuenreferenten ein, sondern ihr referentielles Argument `X` muss mit dem Individuenreferenten `I1` aus der Subjekt-Nominalphrase identifiziert werden. Rein technisch gesehen, macht das jedoch keinen Unterschied zu der Verarbeitung der indefiniten Nominalphrase, denn sowohl der Individuenreferent `I2` der indefiniten Nominalphrase als auch die Variable `X` für die Adjektivphrase müssen durch eine Identitätsbedingung `I2=I1` bzw. `X=I1` an den Individuenreferenten `I1` aus der Subjekt-Nominalphrase gebunden werden.

Kopulatives Verb (Prädikation)

Merkmal	Wert	Beispiel
Wortform	`<wform>`	`is`
Verbform	`fin`	`fin`
Inversion	`inv(+)`	`inv(+)`

Externes Komplement
Kategorie
Numerus
Kasus
	`n2`
	`n0`
	`sing | plur`
	`nom`
		`n2`
		`n0`
		`sing`
		`nom`

Komplement
Kategorie
Numerus
	`a2 | 'a' n1`
	`a0 | n0`
	`sing | plur | _`
		`'a' n1`
		`n0`
		`sing`

Bedingung	`state(S,P)` $I_2=I_1$	`state(S,P)` `I2=I1`

Im Satz (109) wird das Subjekt nicht durch eine Adjektivphrase oder eine Nominalphrase modifiziert, sondern durch eine lokative Präpositionalphrase (`p2`).

(109) *The book is in the library.*

Die Übersetzung des kopulativen Verbs *be* führt hier zu einer Bedingung der Form `state(S,<bform>(I))`, wobei die Grundform *be* - ähnlich wie das bei einstelligen Verben der Fall ist - als Prädikatsnamen verwendet wird. Der Individuenreferent `I` ist aus der Subjekt-Nominalphrase abgeleitet worden. Die Präpositionalphrase trägt ihrerseits zwei Bedingungen zur DRT-E bei, die den Zustand `S` modifizieren. Konkret handelt es sich im Beispielsatz (109) um die beiden Bedingungen `location(S,in(I2))` und `library(I2)`.

Kopulatives Verb (Prädikation)

Merkmal	Wert	Beispiel
Wortform	`<wform>`	`is`
Verbform	`fin`	`fin`
Inversion	`inv(+)`	`inv(+)`

Externes Komplement
Kategorie
Numerus
Kasus
	`n2`
	`n0`
	`sing | plur`
	`nom`
		`n2`
		`n0`
		`sing`
		`nom`

Komplement
Kategorie
Numerus
	`p2`
	`p0`
	`sing | plur`
		`p2`
		`p0`
		`sing`

Bedingung	`state(S,<bform>(I))`	`state(S,be(I))`

Be der Explikation

Das kopulative Verb *be* wird im Satz (110) dazu verwendet, um einen Ausdruck durch eine präzisierende Explikation zu definieren und dadurch von anderen Ausdrücken zu unterscheiden.

(110) *A borrower is a user who has a membership card.*

Hier handelt es sich um eine logische Definition, die einen Ausdruck (*borrower*) einer allgemeineren Klasse bzw. Gattungsbezeichnung (*user*) zuordnet und dann die spezifischen Eigenschaften des Ausdrucks festlegt (*who has a membership card*), die ihn von anderen Mitgliedern der Klasse (z.B. von einem *staff user*) unterscheiden. Das kopulative Verb *be* verbindet den neu eingeführten Ausdruck auf der linken Seite mit den definierenden Ausdrücken auf der rechten Seite und stellt eine logische Gleichung bzw. eine Äquivalenz her. Definitionen stehen nicht in der eigentlichen ACE Spezifikation, sondern sie müssen ausserhalb des Spezifikationstextes festgelegt werden. Die Übersetzung der Definition im Satz (110) führt für das Verb *be* eine Bedingung der Form $isa(I_1, I_2)$ ein. Der definierte Ausdruck kann anschliessend beim Schreiben der Spezifikation gleichberechtigt zum definierenden Ausdruck gebraucht werden.

5.4.1.3 Adjektiv

Adjektive bilden lexikalische Köpfe von Adjektivphrasen und brauchen - mit Ausnahme von zweiwertigen Adjektiven (z.B. *compatible with*) - nicht durch ein zusätzliches Komplement ergänzt zu werden, um syntaktisch verwendungsfähig zu sein. In ACE können einstellige Adjektive als Modifikatoren in Nominalphrasen (= attributive

Stellung) oder ein- und zweistellige Adjektive als prädikative Komplemente in kopulativen Konstruktionen (= prädikative Stellung) gebraucht werden. Nicht alle Adjektive können beide syntaktischen Funktionen in der englischen Sprache wahrnehmen. Bestimmte einstellige Adjektive werden nur in attributiver Stellung verwendet (z.B. *atomic, former*), andere (meistens) nur in prädikativer Stellung (z.B. *ready, safe*). Aus diesem Grund muss der Systembenutzer die einstelligen Adjektive bei der Lexikalisierung unterschiedlich kategorisieren. Ausschliesslich in attributiver oder prädikativer Stellung verwendbare Adjektive erhalten den Wert `attr` bzw. `pred` zugewiesen. Für Adjektive mit freier Stellung ist der Wert `attr/pred` vorgesehen. Das Adjektiv *correct* kann sowohl in attributiver Stellung (111) als auch in prädikativer Stellung (112) gebraucht werden und muss deshalb den Merkmalwert `attr/pred` tragen.

(111) *Every correct password consists of an alphanumeric number.*

(112) *Every password is correct.*

Alle zweistelligen Adjektive erhalten hingegen automatisch das Merkmal `pred` zugewiesen, da sie zusammen mit einem präpositionalen Komplement immer in prädikativer Stellung (113) verwendet werden müssen.

(113) *Every password is compatible with an alphanumeric number.*

Die korrekte Interpretation von Adjektiven scheint schwierig zu sein [vgl. Parson 90:43, Gal et al. 91:130, Allen 95:372]. Eine Teilmenge der Adjektive (z.B. *blue, correct, valid*) ist intersektiv, d.h. die Adjektive schränken die Referenzmenge eines nominalen Ausdrucks (`n1`) ein. Beispielsweise denotiert die Nominalphrase *every correct password* im Anwendungsbereich die Schnittmenge aller korrekten Entitäten mit der Menge aller Entitäten, die Passwörter sind. Eine andere Teilmenge der Adjektive (z.B. *big, long, slow*) ist skalar, d.h. die Adjektive beschreiben eine messbare Eigenschaft, die relativ zum modifizierten nominalen Ausdruck interpretiert werden muss. Beispielsweise ist ein langes Passwort erheblich kürzer als ein langer Satz. Für die korrekte Interpretation muss hier der Kontext in Betracht gezogen und auf Weltwissen zurückgegriffen werden. Eine weitere Teilmenge der Adjektive ist intensional (z.B. *alleged, former, possible*), d.h., diese Adjektive beschreiben einen nominalen Ausdruck unter verschiedenen möglichen Aspekten und Einschätzungen, die nur schwer intersubjektiv überprüfbar sind. Intensionale Adjektive sind - aus den gleichen Gründen wie modale Konstruktionen - in ACE nicht zulässig.

Intersektive und skalare Adjektive werden bei der Übersetzung in die DRT-E gleich

behandelt, d.h. im Fall von skalaren Adjektiven wird kein Vergleichsmassstab irgendwelcher Art formalisiert, sondern es ist der Anwendungsspezialist, der bei der Ausführung der Spezifikation den Bezug zum aktuellen Kontext herstellt und das "Mass der Dinge" ist [vgl. Parsons 90:43 ff.]. Im Fall von attributiven Adjektiven führt die Übersetzung in der DRT-E zu einer Bedingung der Form `<wform>(I)`. Adjektive in prädikativen Konstruktionen ergeben in der DRT-E eine Bedingung der Form `state(S,<wform>(I))` oder `state(S,<wform>(I1,I2))` - je nachdem - ob es sich um ein- oder zweistelliges Adjektiv handelt. Diese Bedingungen kommen aber erst auf der phrasalen Ebene zustande, wo der DRS-Konstruktionsalgorithmus die partielle Bedingung für das kopulative Verb *be* (`state(S,P)`) mit der Bedingung für das prädikative Adjektiv (`<wform>(I)` oder `<wform>(I1,I2)`) durch Unifikation zusammenführt.

Einstelliges Adjektiv

Merkmal	Wert	Beispiel
Wortform	`<wform>`	`correct`
Stellung	`attr | pred | attr/pred`	`attr/pred`
Grad	`pos | compa | super`	`pos`
Bedingung	`<wform>(I)`	`correct(I)`

Zweistelliges Adjektiv

Merkmal	Wert	Beispiel
Wortform	`<wform>`	`compatible`
Stellung	`pred`	`pred`
Grad	`pos | compa | super`	`pos`

Komplement
Kategorie
Prepform
	`p2`
	`p0`
	`in | to | with | …`
		`p2`
		`p0`
		`with`

Bedingung	`<wform>(I1,I2)`	`compatible_with(I1,I2)`

Durch Komparation skalarer Adjektive können in ACE den Entitäten Eigenschaften in gradueller Abstufung zu- oder abgesprochen werden. Die Vergleichsformen der Adjektive werden entweder durch Flexionsformen (z.B. *bigger, biggest*) oder mit Hilfe von Gradadverbien gebildet (z.B. *as powerful as, more powerful, most powerful*). Mit dem Positiv (`pos`) wird die Normalform des Adjektivs bezeichnet (z.B. *big*) oder der gleiche Grad einer Eigenschaft gekennzeichnet (z.B. *as big as*). Mit dem Komparativ (`compa`)

wird ausgedrückt, dass zwei Entitäten hinsichtlich einer Eigenschaft ungleich sind (z.B. *bigger than*). Der Superlativ (`super`) bringt zum Ausdruck, dass einer Entität aus einer Menge von Entitäten der höchste Grad einer Eigenschaft zukommt (z.B. *biggest*). Vergleichsformen führen in der DRT-E zu Bedingungen der Form `state(S,P(I1,I2))`, wobei der Prädikatsnamen `P` allenfalls durch Konkatenation mit den Gradadverbien zustande kommt (z.B. `more_powerful(I1,I2)`, `most_powerful(I1,I2)`). Die regelmässigen Steigerungsformen für den Komparativ und den Superlativ werden bei der Lexikalisierung automatisch abgeleitet. Eine kleine Anzahl von Adjektiven (z.B. *first, next, last*) haben keine Steigerungsformen und müssen vom Anwendungsspezialisten bei der Lexikalisierung gekennzeichnet werden, so dass ihnen das Merkmal (`pos`) zugewiesen werden kann.

Viele Adjektive haben das gleiche Suffix wie verbale Partizip Perfektformen und sind von diesen abgeleitet (z.B. *broken, checked out, closed*). Solche Adjektive können in ACE dazu verwendet werden, um einen Bezug zu einem vorausgehenden Ereignis herzustellen und um Zustände einzuführen:

(114) *If a staff user checks out a copy of a book to a borrower*
then the book is checked out to the borrower.

Diese Adjektive können in attributiver und prädikativer Stellung gebraucht werden und dürfen nicht mit Passivkonstruktionen verwechselt werden, da diese in ACE nicht zulässig sind.

5.4.1.4 Adverb

Adverbien bilden lexikalische Köpfe von Adverbialphrasen und können in ACE selber nicht weiter modifiziert werden. Adverbien weisen keine einheitliche Form auf, doch werden viele Adverbien durch Suffixbildung von Adjektiven abgeleitet (z.B. *correct -> correctly, normal -> normally, slow -> slowly*). Verbmodifizierende Adverbien sind optionale Modifikatoren von Eventualitäten und können in ACE nur in der Adjunktposition von Verbalkonstruktionen (`v1`) stehen. Das heisst, sie stehen entweder direkt vor dem verbalen Kopf wie im Satz (115) oder folgen allenfalls einem obligatorischen Komplement wie im Satz (116).

(115) *The staff user slowly enters the password.*

(116) *The staff user enters the password slowly.*

Mit verbmodifizierenden Adverbien können in Spezifikationen - ganz allgemein ge-

sprochen - die Umstände angegeben werden, die eine Eventualität genauer bestimmen. Adverbiale Umstandsbestimmungen führen in der Spezifikation zu nichtfunktionalen Anforderungen, die durch den Anwendungsspezialisten validiert werden müssen. Wie wir später sehen werden, können in ACE Eventualitäten nicht nur durch Adverbien, sondern auch durch adjungierte Präpositionalphrasen, die in den Verbalkonstruktionen stehen, modifiziert werden.

Die Kategorisierung von Adverbien führt je nach Gliederungsaspekt, der verfolgt wird, zu ganz unterschiedlichen Einteilungen in nichtabzählbare Teilklassen von Modifikatortypen [vgl. Somers 87]. In ACE werden die Adverbien so kategorisiert, dass sie mit Ergänzungsfragen in Beziehung gebracht werden können, so dass der Anwendungsspezialist die modifizierenden Elemente in der Spezifikation erfragen kann. Lokaladverbien können eine Eventualität im physikalischen Raum lokalisieren, indem entweder ein Ort (z.B. *downstairs*) oder eine Richtung (z.B. *outwards*) angezeigt wird. Der Ort lässt sich mit dem Frageadverb *where* erfragen und die Richtung mit *where ... to*. Durch Temporaladverbien können Eventualitäten in Bezug auf einen Zeitpunkt (z.B. *today*), auf einen Zeitraum (z.B. *long*) oder auf eine Frequenz hin (z.B. *daily*) näher bestimmt werden. Als temporales Frageadverb steht *when* zur Verfügung, um nach einem Zeitpunkt zu fragen. Um nach einem Zeitraum oder einer Frequenz zu fragen, wird das Frageadverb *how* in Verbindung mit den Temporaladverbien *long* oder *often* gebraucht. Durch Adverbien der Qualität (= Art und Weise) wird eine Eventualität qualitativ bestimmt (z.B. *correctly*). Mit dem Frageadverb *how* allein kann die Art der Qualität erfragt werden.

Die Übersetzung von verbmodifizierenden Adverbien führt in der DRT-E zu einer Bedingung der Form `M(S,T)` oder `M(E,T)`, wobei der Prädikatsnamen `M` eine Instanz aus der Menge `{location, direction, time, duration, frequency, manner}` ist, `S` oder `E` für eine Eventualität steht und `T` eine adverbiale Konstante ist, die die Eventualität benennt. Jedes Element der Menge `M` ist in ACE eindeutig einem Frageadverb aus der Kategorie *Fragewort* zugeordnet.

Adverbien

Modifikator	Fragewort	Beispiel	Bedingung
Ort	where	downstairs	`location(E,downstairs)`
Richtung	where ... to	outwards	`direction(E,outwards)`
Zeitpunkt	when	today	`time(E,today)`

Modifikator	Fragewort	Beispiel	Bedingung
Zeitraum	how long	long	`duration(E,long)`
Frequenz	how often	daily	`frequency(E,daily)`
Art und Weise	how	correctly	`manner(E,correctly)`

Ausser den verbmodifizierenden Adverbien und den Frageadverbien kennt ACE noch zwei andere Typen von Adverbien, die syntaktische Funktionen haben: Gradadverbien in Vergleichsformen und adverbiale Partikel in phrasalen Verben. Zudem sind eine Reihe von Adverbien (z.B. *each* und *together*) für Disambiguierungsaufgaben vorgesehen, und zwar um in Pluralsätzen entweder kollektive oder distributive Lesarten zu erzwingen [vgl. Covington 96].

5.4.2 Funktionswörter

Zur Klasse der Funktionswörter gehören in ACE die sechs Kategorien: *Präposition, Spezifikator, Pronomen, Koordinator, Subordinator* und *Fragewort*. Diese Kategorien umfassen weitere Teilkategorien, die an entsprechender Stelle diskutiert werden. Im Gegensatz zu den Inhaltswörtern sind die Funktionswörter vordefiniert und können von den Anwendungsspezialisten nicht verändert oder erweitert werden.

Funktionswörter

Kategorie	Beispiel
Präposition	at, for, in, into, of, on, ...
Spezifikator	a, there is a, the, N's, each, every, for every, no, there is no, not
Pronomen	he, she, it, him, her
Koordinator	and, or, either ... or
Subordinator	if ... then, who, which, that
Fragewort	who, which, where, how, does, ...

Die meisten Funktionswörter erfüllen vor allem strukturelle Aufgaben, indem sie syntagmatische, syntaktische oder textuelle Beziehungen herstellen. Syntagmatische Beziehungen entstehen in ACE beispielsweise durch Koordinatoren (*and, or, either ... or*), die vollständige Phrasen und Teilsätze der gleichen syntaktischen Kategorie verbinden, wobei bei komplexen Verknüpfungen die Abfolge aufgrund von Vorrangsregeln bestimmt wird. Syntaktische Abhängigkeitsbeziehungen können in ACE beispielsweise durch Subordinatoren (*if ... then, who, which, that*) ausgedrückt werden, die konditionale und restriktive Verhältnisse zwischen den Teilsätzen anzeigen. Textuelle Beziehungen kommen in ACE beispielsweise durch die anaphorische Verwendungsweise des definiten Artikels (*the*) in Nominalphrasen und durch Personalpronomen (*he, she, it, him, her*) anstelle der nominalen Information in Nominalphrasen zustande. Mit anaphorischen Konstruktionen werden Identitätsbeziehungen zu vorausgehenden nominalen Konstruktionen im Text hergestellt, um dadurch Bezugsgleichheit (Koreferenz) auszudrücken.

Eine vom Anwendungsbereich unabhängige Definition der Funktionswörter ist deshalb möglich, weil einerseits ihre Anzahl klein ist, andererseits ihre Funktion stabil ist und gleichzeitig eine grosse Aufkommenshäufigkeit in den unterschiedlichen Spezifikationstexten vorliegt.

5.4.2.1 *Präposition*

Syntaktisch funktionieren Präpositionen (`p0`) als lexikalische Köpfe von Präpositionalphrasen (`p2`) und selektionieren in ACE ausschliesslich Nominalphrasen (`n2`) als Komplemente. Beispielsweise handelt es sich bei der Konstituente *from the library* um eine Präpositionalphrase, in der die Präposition *from* als lexikalischer Kopf funktioniert. Die Präposition selektioniert die Nominalphrase *the library* als Komplement und weist ihr (im Prinzip) Kasus zu. Der syntaktische Status von Präpositionalphrasen in ACE ist entweder der eines Komplements oder der eines Adjunkts.

Die Kategorie der Präpositionen besteht aus einer kleinen Menge von Mitgliedern (u.a. *of, in, to, for, with, at, on, into*), die ein sehr hohes Aufkommen in Spezifikationstexten haben und oft bloss syntaktische Anschlussfunktion übernehmen [vgl. Alexejew et al. 73:192 ff.]. In dieser Hinsicht zählen die Präpositionen zu den Funktionswörtern. Auf der anderen Seite verhalten sich Präpositionen aber ähnlich wie relationale Inhaltswörter, denn sie können entweder Eventualitäten aufgrund eines festgelegten Verhältnisses modifizieren (z.B. *removes the copy from the library*) oder Entitäten zueinander in Beziehung setzen (z.B. *a copy of the book*).

Präpositionen in Komplementen

Als interne Komplemente ergänzen Präpositionalphrasen (`p2`) in ACE entweder einen verbalen Kopf (`v0`) wie in (117) zu einer Verbalphrase (`v2`) oder einen adjektivischen Kopf (`a0`) wie in (118) zu einer Adjektivphrase (`a2`). Die Präpositionen (`p0`) üben dabei syntaktische Anschlussfunktion aus und weisen der abhängigen Nominalphrase (`n2`) Kasus zu.

(117) *The borrower [v2 [v1 [v0 asks] [p2 [p1 [p0 for] [n2 a registration form]]]]].*

(118) *Every password is [a2 [a1 [a0 compatible] [p2 [p1 [p0 with] [n2 a number]]]]].*

Die Präpositionen werden in beiden Sätzen von den lexikalischen Köpfen (`v0`/`a0`) selektioniert und können nicht ausgetauscht werden, da sie Bestandteil der lexikalischen Information sind. Das Merkmal *Prepform* im Lexikon sorgt dafür, dass der lexikalische Kopf jeweils zusammen mit der passenden Präposition verwendet wird. Da diese Präpositionen nur zur syntaktischen Verknüpfung dienen und keine eigene Bedeutung haben, werden sie für die Darstellung in der DRT-E mit der Grundform des selektionierenden Wortes konkateniert (z.B. `ask_for(`I_1`,`I_2`)` und `compatible_with(`I_1`,`I_2`)`). Damit der Prädikatsname bei der Verarbeitung nicht jedes Mal neu konstruiert werden muss, ist die konkatenierte Form im Lexikon beim selektionie-

renden Wort abgespeichert. Diese Spezialisierung ermöglicht eine effiziente maschinelle Verarbeitung.

Präpositionen in Komplementen

Merkmal	Wert	Beispiel
Wortform	`<wform>`	`out`
Prepform	`out | to | with | …`	`out`

Externes Komplement
Kategorie
Kasus
	`n2`
	`n0`
	`acc`
		`n2`
		`n0`
		`acc`

Bedingung	`(absorbed)`	`(absorbed)`

Präpositionen in postnominalen Komplementen

Eine ähnliche interne Strukturierung wie bei Verbal- und Adjektivphrasen findet man bei Nominalphrasen. Nominale Köpfe können in ACE bestimmte Typen von Präpositionalphrasen als Komplemente selektionieren. Bei den Präpositionalphrasen handelt es sich um sogenannte *of*-Konstruktionen (auch *of*-Genitiv genannt), die aus dem präpositionalen Kopf *of* und einer abhängigen Nominalphrase bestehen (z.B. *of Physics*). Obwohl die Abhängigkeit dieser *of*-Konstruktionen vom lexikalischen Kopf weniger stark ist als die Abhängigkeit von verbalen Komplementen, gibt es eine Vielzahl von Daten, die darauf hindeuten, dass es sich bei *of*-Konstruktionen um nominale Komplemente handelt [vgl. Radford 88:175 ff.]. Es zeigt sich, dass Komplemente immer näher beim nominalen Kopf stehen als Adjunkte. Aus diesem Grund ist der Satz (119) in der englischen Sprache zulässig, aber nicht der Satz (120), bei dem das Adjunkt dem Komplement vorausgeht.

(119) *The staff user removes a [n1 [n1 [n0 book] [p2 of Physics]] [p2 with a damaged cover]].*

(120) **The staff user removes a [n1 [n1 [n0 book] [p2 with a damaged cover]] [p2 of Physics]].*

Diese beiden natürlich-sprachlichen Sätze sind keine ACE Sätz, sondern sie dienen dazu, um den Unterschied zwischen postnominalen Komplementen und Adjunkten klar zu machen. In ACE sind nur postnominale Komplemente zulässig, die durch eine *of*-Konstruktion realisiert werden und einen nominalen Kopf ergänzen. Postnominale Adjunkte, die eine *n1*-Kategorie modifizieren, sind hingegen nicht erlaubt, da diese Modifikatoren zu Ambiguität führen können.

Der Satz

(121) *The staff user removes a [n1 [n0 book] [p2 of Physics]].*

ist in ACE zulässig, denn er enthält mit der Präpositionalphrase *of Physics* ein postnominales Komplement, das den nominalen Kopf *book* ergänzt. Der Kopf (*n0*) und das Komplement (*p2*) bilden zusammen eine Konstituente der Kategorie *n1*. Genaugenommen sind auch *of*-Konstruktionen unterspezifiziert und deshalb mehrdeutig, doch lassen sich mit ihnen in ACE transitive Relationen zwischen mehreren Entitäten herstellen. Dadurch können beim Schreiben von Spezifikationen Entitäten indirekt aufeinander bezogen werden.

Nomen wie *book, copy* und *number* werden häufig relational verwendet. Das heisst, Nominalphrasen, die solche Nomen als lexikalische Köpfe aufweisen, können binäre Relationen erzeugen, indem das Grundwort (z.B. *book*) zu einem Bestimmungswort (z.B. *Physics*) in Beziehung gesetzt wird, das in der natürlichen Sprache implizit gedacht oder durch eine Präpositionalphrase explizit genannt wird. Das Bestimmungswort führt entweder eine zusätzliche Bedingung ein, die das Grundwort genauer bestimmt, oder wirkt als Kohäsionsmittel, das einen anaphorischen Bezug herstellt. Binäre Relationen dieser Art müssen in ACE immer explizit durch *of*-Konstruktionen eingeführt werden, wie zum Beispiel in den folgenden Sätzen:

(122) *The book of Hofstadter is in the catalogue.*

(123) *A copy of the book is available.*

(124) *The staff user writes the number of the copy down.*

Den *of*-Konstruktionen in diesen drei Sätzen lassen sich recht unterschiedliche inhaltliche Leistungen zuordnen, die jeweils durch Paraphrasen verdeutlicht werden können. In (122) gibt die *of*-Konstruktion möglicherweise den Autor eines Werkes an. Man spricht in diesen Zusammenhang von einem Genitivus auctoris. In (123) bezeichnet die *of*-Konstruktion ein konzeptionelles Ganzes, von dem das Grundwort einen Teil angibt. Man spricht in diesem Fall auch von einem Genitivus partitivus. In (124) drückt die *of*-Konstruktion eine Zugehörigkeit zum Grundwort im weitesten Sinne aus. Man spricht hier manchmal auch von einem Genitivus possessivus.

Bei der Übersetzung der *of*-Konstruktionen in die DRT-E wird nicht versucht, die inhaltliche Leistung dieser Konstruktionen darzustellen, sondern es interessieren ausschliesslich die transitiven Relationen, die aus den binären *of*-Prädikaten abgeleitet

werden können und dann jeweils zwei Individuenreferenten durch eine Bedingung der Form `of(`I_1`,`I_2`)` miteinander in Beziehung setzen. Aufgrund der Information, die durch die *of*-Konstruktionen aus den drei Sätzen (122)-(124) verfügbar wird, kann man schliessen, dass sich die Nummer der Kopie in (124) auf die Kopie des Buches von Hofstadter in (122) bezieht.

Im Unterschied zu Präpositionen in internen Komplementen wird die Präposition bei der Übersetzung von *of*-Konstruktionen nicht mit dem Grundwort konkateniert, da diese Nomen auch nicht-relational verwendet werden können und man eine doppelte Lexikalisierung vermeiden möchte.

Präpositionen in postnominalen Komplementen

Merkmal	Wert	Beispiel
Wortform	`<wform>`	`of`

Komplement
Kategorie
Kasus
	`n2`
	`n0`
	`acc`
		`n2`
		`n0`
		`acc`

Bedingung	`<wform>(`I_1`,`I_2`)`	`of(I1,I2)`

Eine in ACE zulässige Alternative zu gewissen postnominalen *of*-Konstruktionen sind possessive Determinatoren, die Spezifikatoren (`specN`) von Nominalphrasen bilden (z.B. *John's password*). Possessive Determinatoren stehen vor dem nominalen Kopf, den sie ergänzen und bestehen ihrerseits aus einer Nominalphrase (z.B. *John's*). Diese Nominalphrasen haben Eigennamen oder Gattungsbezeichnungen als nominale Köpfe. Die Nomen sind durch das Genitivsuffix *'s* markiert und haben den semantischen Typ `person`. Anstelle der *of*-Konstruktion *the password of John* kann in ACE alternierend die possessive Konstruktion *John's password* verwendet werden. In beiden Fällen führt die Übersetzung in der DRS zu einer binären Bedingung der Form `of(`I_1`,`I_2`)`.

Präpositionen in Adjunkten

Als Adjunkte können Präpositionalphrasen (`p2`) in ACE entweder an verbale `v1`-Konstituenten wie in (125) oder an adjektivische `a1`-Konstituenten wie in (126) angeschlossen werden.

(125) *The staff user [`v2` [`v1` [`v1` [`v0` removes] [`n2` the copy]] [`p2` from the library]]].*

(126) *The password is [`a2` [`a1` [`a1` [`a0` valid]] [`p2` from Monday]]].*

Adjungierte Präpositionalphrasen modifizieren in ACE ausschliesslich Eventualitäten. Da in ACE keine Präpositionalphrasen als Adjunkte von nominalen Köpfen zugelassen sind - sondern nur *of*-Konstruktionen als Komplemente -, kann sich die Präpositionalphrase *from the library* in (125) nicht auf die Nominalphrase *the copy* beziehen. Aufgrund dieser Einschränkung erhält der Satz (125) eine eindeutige Lesart, die dem Anwendungsspezialisten nach der Übersetzung in der Paraphrase (127) durch geschweifte Klammern angezeigt wird:

(127) *The staff user { removes the copy from the library }.*

Die Präposition in (125) bzw. in (127) kennzeichnet ein lokales Verhältnis, das zwischen demjenigen Ereignis besteht, das durch die verbale `v1`-Konstituente *removes the copy* denotiert wird, und derjenigen Entität, die durch die definite Nominalphrase *the library* denotiert wird. Auf ähnliche Weise legt die Präposition in (126) ein temporales Verhältnis zwischen einem Zustand und einer Entität fest. Viele Präpositionen können nicht auf die Kennzeichnung eines einzigen Verhältnisses festgelegt werden, denn die Art bzw. der Typ der inhaltlichen Information ihres nominalen Komplements spielt dabei eine entscheidende Rolle. So kann die Präposition *from* entweder lokal verwendet werden und die Herkunft bezeichnen (z.B. *from the library*) oder temporal gebraucht werden und einen Startpunkt festlegen (z.B. *from Monday (onwards)*). Den Unterschied machen hier die beiden Nomen *library* und *Monday*, die ganz verschiedene Konzepte denotieren. Da eine Präposition offensichtlich zur Kennzeichnung von unterschiedlichen Verhältnissen verwendet werden kann, werden Präpositionen im Lexikon des Attempto Systems als partielle Bedingungen der Form `M(E,<wform>(I))` verzeichnet, wobei `M` den Typ der Modifikation angibt und `<wform>(I)` mit dem Individuenreferenten `I` diejenige Relation ist, die die Eventualität `E` benennt. Der Individuenreferent `I` wird aus der präpositionalen Nominalphrase abgeleitet, die in der Adjunktposition steht.

Präpositionen in Adjunkten

Merkmal	Wert	Beispiel
Wortform	`<wform>`	`from`

Komplement
Kategorie
Kasus
	`n2`
	`n0`
	`acc`
		`n2`
		`n0`
		`acc`

Bedingung	`M(E,<wform>(I))`	`M(E,from(I))`

Der Modifikator `M` wird in ACE aus der Präposition und dem semantischen Typ des Nomens (`person`, `time`, `object`) abgeleitet. Für den Satz (125) ergibt sich aufgrund der Präposition *from* und dem semantischen Typ `object`, der dem Nomen *library* zugeordnet ist, der Modifikator `origin`. Aus dieser Information entsteht die Bedingung `origin(E,from(I))` in der DRT-E. Wenn die Präposition *from* zusammen mit einem Nomen verwendet wird, dem der Anwendungsspezialist den semantischen Typ `time` zugeordnet hat - wie das für *Monday* im Satz (126) der Fall sein kann -, dann führt das zum Modifikator `start`, woraus schliesslich in der DRT-E die Bedingung `start(E,from(I))` resultiert. Die einzelnen Modifikatortypen stehen - wie das bei Adverbien der Fall ist (vgl. 5.4.1.4 Adverb) - in Beziehung zu einem eindeutigen Fragewort. Mit Fragesätzen kann der Systembenutzer die modifizierende Information bei der Validierung der Spezifikation untersuchen. Nach der Übersetzung lässt sich mit dem Frageausdruck *where ... from* die Herkunft erfragen, die über ein Ereignis ausgesagt wird, und mit dem Frageausdruck *from when ...* der Startpunkt eines Zustands.

Adjungierte Präpositionalphrasen, die Eventualitäten modifizieren, unterscheiden sich nur strukturell von Adverbien. Die Modifikatortypen, die in ACE durch Adverbien realisiert werden können, bilden gerade eine Teilmenge derjenigen Modifikatoren, die durch adjungierte Präpositionalphrasen verfügbar sind. Die untenstehende Tabelle führt alle zur Zeit in ACE verfügbaren Modifikatortypen auf, wobei diejenigen auf dunklem Hintergrund nur durch adjungierte Präpositionalphrasen realisiert werden können.

Modifikatortypen in ACE

Modifikator	Fragewort	Beispiel	Bedingung
Ort	where	in the library	`location(E,in(I))`
Herkunft	where ... from	from the borrower	`origin(E,from(I))`
Richtung	where ... to	into the slot	`direction(E,into(I))`
Zeitpunkt	when	in the morning	`time(E,in(I))`
Zeitraum	how long	for a day	`duration(E,for(I))`
Startpunkt	from when	from eight o'clock	`start(E,from(I))`
Endpunkt	until when	until Monday	`end(E,until(I))`
Frequenz	how often	during every cycle	`frequency(E,during(I))`
Art und Weise	how | with what	with the ISBN	`manner(E,with(I))`
Begleiter	with whom	with Yolande	`comitative(E,with(I))`

5.4.2.2 Spezifikator

Die Kategorie *Spezifikator* setzt sich aus den beiden Teilkategorien *Determinator* und *Negationswort* zusammen. Die Determinatoren (z.B. *a, the, no, every*) besetzen die Spezifikatorposition (`specN`) in Nominalphrasen und ergänzen eine nominale `n1`-Konstituente zu einer Nominalphrase (`n2`). Die Determinatoren haben einen viel grösseren Einfluss auf die Konstruktion der DRS als die syntaktische Struktur der Sätze vermuten lässt. Von der Sytax her hat ein Determinator nur ein Argument, und zwar die verbleibende nominale `n1`-Konstituente. Semantisch stellt ein Determinator `D` eine Relation zwischen einer quantifizierten Variablen `x`, einem Restriktor `Rx` und einem nuklearen Skopus `Sx` dar: `D(x,Rx,Sx)`. Die quantifizierte Variable `x` kann als Individuenreferent verstanden werden, der aus der Nominalphrase abgeleitet worden ist. Der Restriktor `Rx` umfasst die Bedingungen aus der verbleibenden nominalen `n1`-Konstituente ohne den Determinator. Die Bedingungen des Restriktors `Rx` schränken die Menge der Entitäten ein, die dem Individuenreferenten aufgrund des Determinators zugeordnet werden können, und beschreiben das Denotat der `n1`-Konstituente. Der nukleare Skopus `Sx` umfasst alle aus der Verbalphrase abgeleiteten Bedingungen, für die angenommen wird, dass sie zutreffen, wenn sich passende Entitäten für den Individuenreferenten finden lassen. Der nukleare Skopus `Sx` beschreibt dann das Denotat der Verbalphrase. Daraus folgt, dass ein Determinator eine spezifische Relation zwischen einem Denotat für eine `n1`-Konstituente und einem Denotat für eine Verbalphrase (`v2`) herstellt. Die beiden Denotate sind Entitätsmengen. Die verschiedenen Determinatoren spezifizieren unterschiedliche Relationen zwischen solchen Entitätsmengen. Der Determinator allein ist kein logischer Quantor, sondern bloss ein Bestandteil, um einen logischen Quantor zu bilden. Die quantifizierenden Ausdrücke der natürlichen Sprache bestehen normalerweise aus einer vollständigen Nominalphrase, die sich aus einem Determinator und einem Restriktor zusammensetzt [vgl. Barwise & Cooper 81:159 ff., Partee et al. 90:373 ff., Kamp & Reyle 93:314 ff., Chierchia 95:11 ff.].

Die Nominalphrasen quantifizieren nicht in Isolation, sondern ihr Verhalten muss so beschrieben werden, dass die richtigen Voraussagen über ihre Rollen im Text gemacht werden können. Es reicht beispielsweise nicht aus, dass eine indefinite Nominalphrase in der DRT-E in einen Restriktor, d.h. eine Reihe von Bedingungen für die Nominalphrase, und einen existentiell quantifizierten Individuenreferenten übersetzt wird, der global in der ganzen Spezifikation zugänglich ist, sondern es müssen die anderen quantifizierenden Nominalphrasen im Satz berücksichtigt werden, die die quantifizierende Kraft der indefiniten Nominalphrase bestimmen.

Gemäss ihrer Funktion werden die in ACE zulässigen Determinatoren in die folgenden Kategorien unterteilt: *indefiniter Determinator, definiter Determinator, possessiver Determinator, universeller Determinator* und *negativer Determinator*.

Spezifikatoren

Spezifikator	Beispiel
Determinator	
Indefiniter Determinator	a/an, there is a,
Definiter Determinator	the
Possessiver Determinator	N's
Universeller Determinator	each, every, for every
Negativer Determinator	no, there is no
Negationswort	does not, not

Die Kategorie *Negationswort* zählt als zweite Teilkategorie zu den Spezifikatoren von ACE. Die Negationswörter besetzen die fakultative Spezifikatorposition (`specV`) in der Verbalphrase und können durch die beiden Negationsausdrücke *does not* und (*is) not* realisiert werden. Die Negationsausdrücke verneinen die aus der Verbalphrase abgeleiteten Bedingungen. Diese Form der Negation wird auch als interne Negation bezeichnet und steht im Kontrast zur externen Negation, die durch eine Nominalphrase mit dem negativen Determinator *no* zustande kommt [vgl. Partee et al. 90:383]. Die externe Negation entspricht in ACE gerade dann der logischen Negation (= Satznegation), wenn die Nominalphrase mit dem negativen Determinator in der Subjektposition steht. Die Oberflächenreihenfolge der quantifizierten Nominalphrasen im Satz bestimmt das Skopusverhältnis in ACE: Links stehende Nominalphrasen haben immer Skopus über weiter rechts stehende Nominalphrasen.

Nominalphrasen mit kardinalen Determinatoren (z.B. *three staff users*) führen zu Sätzen mit Pluralphänomenen, in denen möglicherweise kollektive, distributive und kumulative Lesarten unterschieden werden müssen. Kollektive und distributive Lesarten werden in der klassischen DRT durch die Einführung von Diskursreferenten für Entitätsmengen behandelt [vgl. Kamp & Reyle 93:305 ff., Graham 94:16 ff.]. Kardinale Determinatoren sind in ACE zur Zeit nicht zulässig.

Nominalphrasen mit vagen Determinatoren (z.B. *most staff users*) oder relativen Determinatoren (z.B. *many staff users*) führen ebenfalls zu Pluralphänomenen und ausserdem zu ungenauer Quantifizierung. Die Nomininalphrase *most staff users* kann entweder so verstanden werden, dass nur mehr als die Hälfte der Mitarbeiter gemeint ist,

oder dass deutlich mehr als die Hälfte der Mitarbeiter - so etwa 75% - gemeint ist. Die Berechnung der Kardinalität der Entitätsmenge, auf die die aus *staff users* abgeleiteten Bedingungen zutreffen, verlangt je nach Lesart unterschiedliche Multiplikationsfaktoren. Auch die Verarbeitung der Nominalphrase *many staff users* würde zusätzliche Information verlangen, und zwar einen durch den Kontext festgelegten (und vom Anwender akzeptierten) Standardwert, relativ zu dem sich die Kardinalität der Entitätsmenge für *staff users* berechnen lässt. Vage und relative Determinatoren beschreiben die zu spezifizierenden Sachverhalte ungenau und sind deshalb in ACE nicht zulässig.

Determinator

Die in ACE zulässigen Nominalphrasen mit Determinatoren entsprechen nur zum Teil den Quantoren in der Prädikatenlogik, denn die syntaktische Struktur von Sätzen mit quantifizierten Nominalphrasen ist verschieden von der syntaktischen Struktur quantifizierter Formeln in der Prädikatenlogik. Die Konstituenz wird in der formalen prädikatenlogischen Struktur nicht ausgedrückt, zudem treten gravierende Probleme auf bei der Behandlung von anaphorischen Ausdrücken. Die DRT-E löst diese Probleme durch eine adäquatere Darstellung, und zwar indem quantifizierte Nominalphrasen in Individuenreferenten und Bedingungen übersetzt werden. Zu diesem Zweck werden für eine Teilmenge der Determinatoren Schemata, d.h. uninstantiierte DRSen, im Lexikon bereitgestellt, die erst bei der DRS-Konstruktion vervollständigt werden. Gewisse Nominalphrasen erzwingen darüber hinaus die Formulierung von kontextsensitiven Regeln, um anaphorische Bezüge aufzulösen.

Indefiniter Determinator

Nominalphrasen mit den indefiniten Determinatoren *a* und *there is a* können in ACE dazu verwendet werden, um Individuenreferenten für Entitäten einer bestimmten Kategorie in den Spezifikationstext einzuführen. Die Übersetzung der indefiniten Nominalphrase `Det` *staff user* in den Sätzen

(128) *A staff user owns a password.*

(129) *There is a staff user who every machine identifies.*

führt in der DRT-E zum gleichen Resultat: `[I] staff_user(I) number(I,sing) gender(I,masc/fem) group(I,count)`. Der indefinite Determinator wird im Lexikon nicht speziell repräsentiert. Der DRS-Konstruktionsalgorithmus muss beim Aufbau der DRS bloss dafür sorgen, dass diejenigen Diskursreferenten und Bedingungen, die den Restriktor und den nuklearen Skopus bilden, in der richtigen Reihenfolge und

unter Berücksichtigung des aktuellen Kontexts in der DRS akkumuliert werden. Die temporären Bedingungen (`number(I,sing) gender(I,masc/fem) group(I,count)`) helfen unter anderem bei der Auflösung von anaphorischer Referenz für nachfolgende Nominalphrasen. In der folgenden Diskussion werden diese temporären Bedingungen der Einfachheit halber nur noch dann notiert, wenn sie in der Argumentation eine Rolle spielen.

Die indefinite Nominalphrase *a staff user* im Satz (128) bildet in ACE die Normalform, um einen neuen Individuenreferenten einzuführen. Die indefinite Nominalphrase *there is a staff user* mit dem Determinator *there is a* wird speziell dazu gebraucht, um das Skopusverhältnis im Satz durch Quantorenanhebung explizit zu machen. Sobald in einem natürlich-sprachlichen Satz zwei oder mehr unterschiedlich quantifizierte Nominalphrasen auftauchen, wird die Bestimmung der Skopusverhältnisse relevant. Im Gegensatz zur natürlichen Sprache, wo das Skopusverhältnis nicht allein aufgrund der textuellen Reihenfolge der Nominalphrasen bestimmt werden kann, ist in ACE die Reihenfolge der Nominalphrasen ausschlaggebend. Die im Satz zuerst eingeführte Nominalphrase hat immer weiten Skopus über die nachfolgenden quantifizierten Nominalphrasen.

Im Fall von attribuierten Nominalphrasen wird der abgeleitete Individuenreferent zusätzlich mit dem Individuenreferent aus der Subjekt-Nominalphrase identisch gemacht (= unifiziert). Beispielsweise wird die Nominalphrase *a staff user* im Satz

(130) *John is a staff user.*

dazu verwendet, um eine Person genauer zu beschreiben. Die Verarbeitung dieser attribuierten Nominalphrase führt zu einem neuen Individuenreferenten `I2` und zu der Bedingung `staff_user(I2)`. Diese Information wird mit den Bedingungen `state(S,P(I1)) I2=I1` für das kopulative Verb *be* unifiziert und führt zu der folgenden (bereinigten) DRS: `[I1,S] named(I1,'John') state(S,staff_user(I1))`. Das Beispiel zeigt, dass nicht alle Fälle des Gebrauchs von indefiniten Nominalphrasen notwendigerweise zu selbständigen Individuenreferenten führen müssen.

Definiter Determinator

Nominalphrasen mit dem definiten Determinator *the* können in ACE unter bestimmten Voraussetzungen ebenfalls einen neuen Individuenreferenten für eine bestimmte Entität einführen. Und zwar führt die definite Nominalphrase *the staff user* im Satz

(131) *The staff user creates a catalogue entry.*

genau dann einen neuen Individuenreferenten `I2` in das Diskursuniversum ein, wenn in den (zugänglichen) DRSen noch kein Individuenreferent `I1` existiert, der den Bedingungen `staff_user(I2)` und `I2=I1` genügt.

Ist eine Entität einmal durch eine indefinite oder definite Nominalphrase eingeführt worden und somit im Spezifikationstext bekannt und identifizierbar, dann werden weitere definite Nominalphrasen, die dieselbe nominale Information bzw. Teilinformation enthalten, anaphorisch interpretiert.

(132) *A password consists of an alphanumeric number. The alphanumeric number ...*

(133) *A password consists of an alphanumeric number. The number ...*

Der anaphorische Gebrauch einer definiten Nominalphrase (*the alphanumeric number* oder *the number*) führt nach der Übersetzung zu einem (temporären) Individuenreferenten `I2` für den definiten Ausdruck zusammen mit einer Reihe von (temporären) Bedingungen für den Restriktor. Der Algorithmus für die Referenzresolution stellt den anaphorischen Gebrauch der definiten Nominalphrase fest, indem er die (zulässigen) DRSen nach denjenigen Bedingungen durchsucht, die als erste alle Bedingungen der aktuellen definiten Nominalphrase subsummieren. Wenn die Suche erfolgreich ist, dann wird eine Identitätsbedingung der Form `I2=I1` erzeugt, wobei `I1` den Individuenreferenten des Antezedens bezeichnet. Der DRS-Konstruktionsalgorithmus unifiziert anschliessend die beiden Individuenreferenten miteinander und entfernt diejenigen Bedingungen, die aus der anaphorisch verwendeten Nominalphrase abgeleitet wurden. Wenn bei der Suche keine passenden Bedingungen gefunden werden, dann wird für die definite Nominalphrase ein neuer Individuenreferent eingeführt.

Die anaphorisch gebrauchte definite Nominalphrase im Satz (133) wird im Vergleich zum nominalen Antezedens ohne das Adjektiv *alphanumeric* verwendet. Unvollständige nominale Anaphern werden in der Paraphrase immer durch die pränominale Information des Antezedens ergänzt. Nach der Verarbeitung durch das Attempto System wird dem Benutzer die Substitution in eckigen Klammern angezeigt.

(134) *The password consists of an alphanumeric number. The [alphanumeric] number ...*

Ausser der assertierenden und anaphorischen Gebrauchsweise können definite Nominalphrasen in ACE auch dazu verwendet werden, um Identitätsaussagen zu machen. Die Übersetzung des Satzes

(135) *John is the borrrower.*

führt in der DRT-E zu einer Identitätsbedingung zwischen dem Individuenreferenten `I1` aus der Subjekt-Nominalphrase und dem Individuenreferenten `I2` aus der definiten Nominalphrase. Die Identitätsbedingung `is(I1,I2)` wird durch das kopulative Verb *be* hergestellt. Die definite Nominalphrase trägt ausserdem die Bedingung `borrower(I2)` zur DRS bei. Nach Auflösung der Identitätsbedingung durch den DRS-Konstruktionsalgorithmus verbleibt in der bereinigten DRS die folgende Information: `[I1] named(I1,'John') borrower(I1)`.

Possessiver Determinator

Nominalphrasen mit einem possessiven Determinator der Form *N's* werden in ACE dazu verwendet, um binäre Relationen zwischen einem Eigennamen und einer Gattungsbezeichnung oder zwischen zwei Gattungsbezeichnungen herzustellen. Die Übersetzung des Satzes

(136) *John's password is valid.*

ergibt die folgende DRS: `[I1,I2,S] password(I1) named(I2,'John') of(I1,I2) state(S,valid(I1))`. Für die Gattungsbezeichnung *password* wird der Individuenreferent `I1` und die Bedingung `password(I1)` abgeleitet. Aus der Nominalphrase *John's*, die als possessiver Determinator funktioniert, entstehen der Individuenreferent `I2` und die beiden Bedingungen `named(I2,'John')` und `of(X,I2)`. Das Genitivsuffix *'s* steuert eine uninstantiierte binäre Bedingung der Form `of(X,Y)` bei, die die Relation zwischen dem Eigennamen und der Gattungsbezeichnung anzeigt. Die Übersetzung der Nominalphrase *John's password* und der Nominalphrase *the password of John* führt in der DRT-E zum gleichen Resultat.

Universeller Determinator

Nominalphrasen mit den universellen Determinatoren *each*, *every* und *for every* werden in ACE dazu verwendet, um über alle Entitäten einer bestimmten Kategorie zu sprechen. Die Übersetzung der universell quantifizierten Nominalphrase `Det` *staff user* in den folgenden drei Sätzen

(137) *Each staff user owns a password.*

(138) *Every staff user owns a password.*

(139) *For every staff user there is a password that he owns.*

führt in der DRT-E zum gleichen Ergebnis: `IF [I] staff_user(I) (THEN S)`. Der universelle Determinator `Det` trägt dazu ein Schema einer uninstantiierten DRS der Form `IF R (THEN S)` bei, das im Lexikon abgespeichert ist. Aus der `n1`-Konstituente wird der neue Individuenreferent `I` und die Bedingung `staff_user(I)` abgeleitet, die zusammen den Restriktor `R` bzw. die sub-DRS des Antezedens bilden. Aus dem Restsatz leiten sich weitere Diskursreferenten und Bedingungen ab, die zusammen den nuklearen Skopus `S` bzw. die sub-DRS der Konsequenz ergeben. Die beiden universellen Determinatoren *each* und *every* in den Sätze (137) und (138) werden in ACE nicht unterschiedlich interpretiert und bilden bloss Schreibvarianten. Der universelle Determinator *for every* im Satz (139) dient speziell dazu, um den weiten Skopus der Nominalphrase explizit zu machen.

Negativer Determinator

Nominalphrasen mit den negativen Determinatoren *no* und *there is no* werden in ACE dazu verwendet, um die Existenz von Entitäten für einen Individuenreferenten einer bestimmten Kategorie zu verneinen. Die Übersetzung der Nominalphrase `Det` *staff user* mit dem negativen Determinator `Det` ergibt für die beiden Sätze

(140) *No staff user owns a password.*

(141) *There is no staff user who owns a password.*

in der DRT-E das gleiche Resultat: `IF [I] staff_user(I) (THEN (NOT S))`. Die beiden negativen Determinatoren stellen eine uninstantiierte DRS der Form `IF R (THEN (NOT S))` im Lexikon bereit. Der Restriktor `R` ist eine sub-DRS, die sich aus einem Individuenreferenten und einer Bedingung zusammensetzt, welche aus der `n1`-Konstituente abgeleitet werden. Der nukleare Skopus `S` ist eine sub-DRS, die die Diskursreferenten und Bedingungen aus dem restlichen Satz enthält. Die Nominalphrase mit dem negativen Determinator *there is no* im Satz (141) dient dazu, um den weiten Skopus der Nominalphrase explizit zu machen. Die Bedeutung der Sätze (140) und (141) ist äquivalent zu derjenigen des Satzes:

(142) *Every staff user does not own a password.*

Beim Satz (142) handelt es sich um einen wohlgeformten ACE Satz, doch ist diese Konstruktion für Muttersprachler kaum akzeptabel.

Skopus von quantifizierten Nominalphrasen

Der Skopus von quantifizierten Nominalphrasen wird in ACE aufgrund ihrer textuellen Reihenfolge im Satz bestimmt. Die im Satz zuerst genannte Nominalphrase hat weiten Skopus über die nachfolgenden quantifizierten Nominalphrasen, wobei sich der Skopus bis zum Satzende erstreckt. Steht eine indefinite (oder definite) Nominalphrase eines Satzes nicht unter dem Skopus einer universellen Quantifizierung, einer Negation oder einer Implikation, dann ist der Skopus der Nominalphrase nicht auf den aktuellen Satz beschränkt. Der Individuenreferent dieser Nominalphrase bleibt dann als Antezedens für Anaphern aus den Folgesätzen zugänglich.

Wenn der Anwendungsspezialist mit der Lesart, die sich aus der textuellen Reihenfolge der quantifizierten Nominalphrasen ergibt, nicht zufrieden ist, dann muss er den Satz explizit umformen, um die intendierte Lesart zu erzwingen. Zu diesem Zweck stehen ihm die drei Determinatoren *there is a*, *for every* und *there is no* und die Relativpronomen *who*, *which* und *that* zur Verfügung. Technisch gesprochen, dienen diese Funktionswörter dazu, um eine Quantorenanhebung durch Topikalisierung herbeizuführen.

Unter Topikalisierung wird in ACE der syntaktische Prozess verstanden, der eine quantifizierte Nominalphrase aus einer internen Komplementposition in die satzexterne `top`-Position setzt, so dass der Skopus der Nominalphrase bzw. des Quantors angehoben wird. Dieser Prozess geschieht in ACE vollkommen systematisch: Die anzuhebende Nominalphrase wird durch Linksextraposition und mit Hilfe eines dafür bestimmten Determinators in die satzexterne `top`-Position gesetzt, von wo aus sie den ganzen Restsatz unter ihren Skopus nimmt. Bei diesem Prozess entsteht immer ein Relativsatz, der eine nominale Lücke aufweist. Das Relativpronomen wird in dieser Struktur nicht als Nominalphrase mit Füllerfunktion interpretiert, sondern als Complementizer (`c0`) aufgefasst, dem bloss satzeinleitende Funktion zukommt. Den Füller der nominalen Lücke (t_i) im Relativsatz bildet die topikalisierte Nominalphrase ($n2_i$).

(143) *[s1 [$n2_i$ There is a staff user] [c1 [c0 [rel who]] [s every machine identifies t_i]]].*

Das definierte Skopusverhältnis im Satz (144) kann mit Hilfe des indefiniten Determinators *there is an* und des Relativpronomens *that* umgestellt werden.

(144) *Every staff user gets an extra holiday.*

Daraus resultiert der Satz:

(145) *There is an extra holiday that every staff user gets.*

Durch die Topikalisierung in (145) erhält die indefinite Nominalphrase *there is an extra holiday* weiten Skopus über die universell quantifizierte Nominalphrase *every staff user*. Es entsteht eine Lesart, wo alle Mitarbeiter die gleiche Art Extraferien erhalten (z.B. über Weihnachten/Neujahr).

Das definierte Skopusverhältnis im Satz (146) kann mit Hilfe der beiden Determinatoren *for every*, *there is a* und des Relativpronomens *that* verändert werden.

(146) *The staff user assigns a number to every book.*

Daraus folgt der Satz:

(147) *For every book there is a number that the staff user assigns to the book.*

Durch die Topikalisierung erhält die universell quantifizierte Nominalphrase *for every book* in (147) weiten Skopus über die indefinite Nominalphrase *a number*. Das ergibt für den Satz (147) eine Lesart, bei der es für jedes Buch eine einmalige Nummer gibt, die der Mitarbeiter einem einzelnen Buch zuweist. Linguistisch interessant ist, dass bei der Linksextraposition der universell quantifizierten Nominalphrase anstelle einer zweiten nominalen Lücke eine definite Nominalphrase (*the book*) als Anapher zurückbleibt. Bei eindeutigem Bezug kann anstelle der definiten Nominalphrase auch ein Personalpronomen (*it*) stehen.

Das definierte Skopusverhältnis im Satz (148) kann mit Hilfe des negativen Determinators *there is no* und des Relativpronomens *that* umgestellt werden.

(148) *Every staff user who works at the information desk gets no extra holiday.*

Das ergibt den Satz:

(149) *There is no extra holiday that every staff user who works at the information desk gets.*

Durch die Topikalisierung in (149) erhält die Nominalphrase *there is no extra holiday* mit dem negativen Determinator weiten Skopus über die universell quantifizierte Nominalphrase *every staff user*. Es entsteht eine Lesart, bei der es keine Extraferien für all diejenigen Mitarbeiter gibt, die am Informationsschalter arbeiten.

Da in ACE quantifizierte Nominalphrasen, die in der Subjektposition stehen, immer

weiten Skopus haben, fallen auch koordinierte Verbalphrasen in ihren Bezugsbereich. Der Satz

(150) *Every staff user either enters the name of an author or enters the subject area.*

erhält natürlicherweise eine Lesart, bei der jeder der Mitarbeiter eine der beiden Alternativen auswählen kann. Wenn jedoch eine Lesart intendiert ist, wo alle Mitarbeiter dieselbe Alternative wählen, dann müssen zwei ganze Sätze koordiniert werden, was dazu führt, dass der Koordinator angehoben wird.

(151) *Either every staff user enters the name of an author or every staff user enters the subject area.*

Wenn eine indefinite oder definite Nominalphrase unter dem Skopus einer universellen Quantifizierung, einer Negation oder einer Implikation steht, dann kann sich eine definite Nominalphrase im Folgesatz nicht anaphorisch auf die vorausgehende Nominalphrase beziehen.

(152) *Every staff user owns a password. The password is valid from Monday onwards.*

Dieses Referenzproblem kann durch eine Relativsatzkonstruktion - wie das im Satz (153) dargestellt ist - umgangen werden, und zwar indem die Information aus dem Folgesatz in das konditionale Satzgefüge hineingezogen wird, so dass die Information unter den Skopus der universell quantifizierten Nominalphrase fällt.

(153) *Every staff user owns a password that is valid from Monday onwards.*

Determinatoren im Überblick

Die einzelnen Determinatoren leisten unterschiedliche Beiträge beim Aufbau der DRS. Nominalphrasen mit indefinitem oder definitem Determinator führen einen neuen existentiell quantifizierten Individuenreferenten und eine Reihe von Bedingungen in die DRS ein. Diese beiden Determinatoren werden nicht explizit im Lexikon dargestellt, da sie bei der DRS-Konstruktion absorbiert werden. Beim Aufbau der DRS muss bloss dafür gesorgt werden, dass die Information des Restriktors und des nuklearen Skopus zusammenfinden. Im Fall eines anaphorischen Gebrauchs der definiten Nominalphrase entsteht zusätzlich eine Identitätsbedingung in der DRS. Nominalphrasen mit einem possessiven Determinator tragen eine binäre Bedingung zur DRS bei, für die im Lexikon beim Genitivsuffix *'s* ein uninstantiiertes Schema abgespeichert ist. Für die universellen und negativen Determinatoren ist im Lexikon ebenfalls je ein Schema abgespeichert, das eine uninstantiierte DRS bereitstellt und das die Information für

den Restriktor und den nuklearen Skopus bei der Verarbeitung aufnehmen kann.

Determinatoren

Determinator	Beispiel	Bedingung
Indefiniter Determinator	a/an, there is a	`absorbed`
Definiter Determinator	the	`absorded | anaphoric`
Possessiver Determinator	N's	`of(X,Y)`
Universeller Determinator	each, every, for every	`IF R (THEN S)`
Negativer Determinator	no, there is no	`IF R (THEN (NOT S))`

Negationswort

Unter syntaktischem Aspekt lassen sich in ACE zwei Gruppen von Negationsformen unterscheiden: Negative Determinatoren in Nominalphrasen und Negationswörter in Verbalphrasen. Diese zwei Negationsformen haben aufgrund ihrer möglichen Stellung im Satz unterschiedlichen Skopus. Negative Determinatoren in Nominalphrasen führen zu Quantifizierung und nehmen in ACE alle weiter rechts im Satz stehenden Informationen in ihren Skopus. Es handelt sich dabei um eine sogenannte externe Negation, die gerade dann mit der logischen Negation zusammenfällt, wenn die negative Nominalphrase in der Subjektposition steht. Negationswörter in der Verbalphrase führen zur Verneinung der Prädikation. Man spricht in diesem Fall von einer internen Negation.

Negative Determinatoren in Nominalphrasen

Negation in Nominalphrasen kommt - wie bereits besprochen - durch die negativen Determinatoren *no* und *there is no* zustande, die als Spezifikatoren in den Nominalphrasen funktionieren. Die negativen Determinatoren *no* und *there is no* werden in ACE als Determinatoren kategorisiert. Der negative Determinator verneint die Existenz von Entitäten für denjenigen Individuenreferenten, der aus der `n1`-Konstituente (d.h. dem Restriktor) abgeleitet werden kann. Formal gesprochen, ist ein Satz mit einer negativen Nominalphrase, die in der Subjektposition steht, genau dann wahr, wenn dem Individuenreferenten der Nominalphrase keine Entität zugeordnet werden kann, so dass sowohl der Restriktor als auch der Skopus erfüllbar ist.

Negationsausdrücke in Verbalphrasen

Negation in Verbalphrasen wird entweder durch den vorangestellten Negationsausdruck *does not* bei Vollverben oder durch den nachgestellten Negationsausdruck *not*

beim kopulativen Verb *be* realisiert. Diese Negationswörter funktionieren in ACE als Spezifikatoren von Verbalphrasen und verneinen die Existenz von Entitäten für Diskursreferenten, die die aus der `v1`-Konstituente (d.h. dem nuklearen Skopus) abgeleiteten Bedingungen erfüllen. Diese Form der Negation verneint die im Satz ausgedrückte Prädikation. Formal gesprochen, ist ein Satz mit einem dieser beiden Negationsausdrücke genau dann wahr, wenn es Entitäten gibt, die auf die Bedingungen im Restriktor zutreffen, aber nicht auf die Bedingungen im nuklearen Skopus.

Der Negationsausdruck *not* stellt im Lexikon eine zusammengesetzte negative DRS der Form `NOT S` bereit. `S` ist eine uninstantiierte sub-DRS, die die neuen Diskursreferenten und Bedingungen aus der `v1`-Konstituente aufnimmt.

Negationswort

Negationswort	Beispiel	Bedingung
Negationsausdruck	does not, not	`NOT S`

Skopus der Negation

Nominalphrasen mit den negativen Determinatoren *no* und *there is no* haben in ACE Skopus über alle im Satz weiter rechts stehenden quantifizierten Nominalphrasen. Die Übersetzung des Satzes

(154) *There is no staff user who owns a password.*

führt zu der folgenden DRS: `IF [I1] staff_user(I1) (THEN (NOT [S1,I2] state(S1,own(I1,I2)) password(I2)))`. Hinter dieser Darstellungsform verbirgt sich die logische Äquivalenzbeziehung $\neg \exists x(Rx \wedge Sx) \equiv \forall x(Rx \rightarrow \neg Sx)$.

Im Gegensatz zu den Nominalphrasen mit negativen Determinatoren haben die Negationsausdrücke *does not* und *not* nur Skopus über diejenigen Bedingungen, welche aus der `v1`-Konstituente abgeleitet werden können. Die Übersetzung des Satzes

(155) *John does not own a password.*

ergibt die folgende DRS: `[I1] named(I1,'John') (NOT [S1,I2] state(S1,own(I1,I2)) password(I2))`.

Im Fall von koordinierten Nominalphrasen wie im Satz (156) wird der Negationsausdruck bei der Verarbeitung zusammen mit dem Verb distribuiert. Dabei ist zu

beachten, dass die Negationsausdrücke in ACE stärker binden als die Koordinatoren. Das heisst die Koordination im Satz

(156) *The borrower does not press the key 'A' and the key 'B'.*

wird verstanden (und paraphrasiert) als

(157) *The borrower does not press the key 'A' and [does not press] the key 'B'.*

Dem Satz liegen zwei separate Ereignisse `E1` und `E2` zugrunde, die negiert werden. Als Resultat der Übersetzung für den Satz (157) entsteht eine DRS mit zwei negierten sub-DRSen: `[I1] borrower(I1) (NOT [E1,I2] event(E1,press(II,I2)) key(I2) string(I2,'A')) (NOT [E2,I3] event(E2,press(II,I3)) key(I3) string(I3,'B'))`.

5.4.2.3 Pronomen

In ACE sind drei unterschiedliche Arten von Pronomen verfügbar: Personalpronomen (*he, she, it, him, her*), Relativpronomen (*who, which, that*) und Fragepronomen (z.B. *who, whose, which, what*). Personalpronomen funktionieren als nominale Anaphern und stellen textuelle Beziehungen zu vorausgehenden Nominalphrasen her. Im Gegensatz zu Personalpronomen haben Relativpronomen und Fragepronomen die Fähigkeit aus einfachen Sätzen zusammengesetzte Sätze oder einfache Fragesätze zu bilden. Aufgrund dieser Fähigkeiten zählen Relativpronomen in ACE zu den Subordinatoren und die Fragepronomen zu den Fragewörtern.

Bei den Personalpronomen wird die Kasusmarkierung sichtbar. Personalpronomen haben unterschiedliche Formen, wenn sie im Nominativ (`nom`) oder im Akkusativ (`acc`) verwendet werden. Der binäre Merkmalwert `pro(+)`, der den nominalen Typ kennzeichnet, unterscheidet Personalpronomen von Nomen. Diese Unterscheidung ist notwendig, weil Personalpronomen keine Komplemente selektionieren können.

Personalpronomen

Merkmal	Wert	Beispiel
Wortform	`<wform>`	`she`
Numerus	`sing | plur`	`sing`
Gender	`masc | fem | neut`	`fem`
Kasus	`nom | acc`	`nom`
Typ	`pro(+)`	`pro(+)`

Merkmal	Wert	Beispiel
Bedingung	I_2=I_1	I2=I1

Personalpronomen beziehen sich in ACE auf einen Individuenreferenten (I_1), der durch eine Nominalphrase in einem vorausgehenden Satz oder Teilsatz eingeführt worden ist. Zum Beispiel:

(158) *The password consists of an alphanumeric number. It ...*

Die Referenzresolution geschieht genau so wie bei definiten Nominalphrasen, die anaphorisch gebraucht werden. Um die Auflösung zu kontrollieren, werden bei der Übersetzung in die DRT-E für jeden Individuenreferenten, der aus einer Nominalphrase abgeleitet wird, temporäre Numerus- und Genderbedingungen (*number(I_1, NumVal)*, *gender(I_1,GendVal)*) in die DRS eingeführt. Die Nominalphrase, welche das Personalpronomen enthält, führt ihrerseits einen neuen Individuenreferenten (I_2) mit Numerus- und Genderbedingungen in die DRS ein. Die Lokalisierung des korrekten Antezedenten für diesen neuen Individuenreferenten ist in der vollen natürlichen Sprache oft schwierig, weil der Suchraum gross und die Anschlussmöglichkeiten vielfältig sein können. Aus diesem Grund müssen Personalpronomen in ACE in einer kontrollierten Art und Weise verwendet werden. Ein Personalpronomen kann nur auf das letzte nominale Antezedens bezogen werden. Die Referenzresolution erfolgt aufgrund von kürzester textueller Distanz, Numerus und Gender. Die Beziehung zwischen den beiden Individuenreferenten wird durch eine Identitätsbedingung der Form I_2=I_1 hergestellt, wobei I_1 das erste passende Antezedens sein muss, das die gleichen Werte für die Merkmale Numerus und Gender trägt wie der Individuenreferent I_2, der für das Personalpronomen steht. Nach der Verarbeitung wird das Ergebnis der Referenzresolution für den Anwendungsspezialisten in einer Paraphrase sichtbar:

(159) *The password consists of an alphanumeric number. [The alphanumeric number] ...*

Hier ist das Personalpronomen *it* durch die passende definite Nominalphrase substituiert worden. Die definite Nominalphrase - in eckigen Klammern - nimmt in der Paraphrase die vollständige Information des Antezedens wieder auf.

5.4.2.4 Koordinator

Die Kategorie *Koordinator* umfasst in ACE die drei Bindewörter: Konjunktion, inklusive Disjunktion und exklusive Disjunktion. Diese Koordinatoren können dazu verwendet werden, um Sätze (*s*) und volle phrasale Kategorien (*n2*, *v2*, *p2*, *a2*, *adv2*), die vom gleichen Typ sind, miteinander zu verknüpfen.

Koordinatoren

Koordinator	Beispiel	Bedingung
Konjunktion	and	C_1, …, C_n, C_{n+1}, …, C_m
Inklusive Disjunktion	or	OR K_1 OR K_2
Exklusive Disjunktion	either ... or	EITHER K_1 OR K_2

In ACE sind alle phrasalen Kategorien koordinierbar, ausser Nominalphrasen (*n2*) in der Subjektposition und postnominale *of*-Konstruktionen (*p2*), weil diese Konstruktionen zu Pluralphänomenen führen (können), welche zum gegenwärtigen Zeitpunkt von der Sprache ACE nicht abgedeckt werden. Die nachfolgenden Sätze bilden eine Auswahl von zulässigen Koordinationsmustern, bei denen syntaktisch gleichrangige Konstituenten verbunden werden.

(160) *[s The staff user enters a subject area] or [s the borrower enters a subject area].*

(161) *The staff user [v2 creates a catalogue entry] and [v2 enters the id of the copy].*

(162) *The borrower enters [n2 a name] and [n2 a subject area].*

(163) *The password is either [a2 correct] or [a2 wrong].*

(164) *The book is either [p2 in the library] or [p2 in the bookbindary].*

(165) *The staff user moves the lever [adv2 back] and [adv2 down].*

In (160) werden zwei Hauptsätze (*s*) koordiniert; in (161) zwei Verbalphrasen (*v2*) aneinandergereiht; in (162) zwei Nominalphrasen (*n2*) nebengeordnet; in (163) zwei Adjektivphrasen (*a2*) verbunden; in (164) zwei Präpositionalphrasen (*p2*) verknüpft und in (165) zwei Adverbialphrasen (*adv2*) koordiniert.

Konjunktion

Im Unterschied zur klassischen Logik, wo die Reihenfolge der Konjunkte keine Rolle spielt, hat die textuelle Reihenfolge von koordinierten Konstituenten in der natür-

lichen Sprache oft eine kommunikativ relevante Funktion. Man möchte damit zum Beispiel ausdrücken, dass ein Ereignis auf ein anderes Ereignis folgt. In ACE kann die deterministische Konjunktion *and* beispielsweise dazu verwendet werden, um die zeitliche Anordnung der durch die Konstituenten beschriebenen Eventualitäten festzulegen. Wird eine Konstituente $X2_1$ durch die deterministische Konjunktion *and* mit einer zweiten Konstituente $X2_2$ verbunden, dann führt das in der DRT-E zu einer festen linearen Anordnung von Bedingungen der Form $C_1, ..., C_n, C_{n+1}, ..., C_m$.

Wenn der Anwendungsspezialist beispielsweise einen Sachverhalt beschreiben möchte, wo der Ausleiher zuerst die Taste *'A'* drückt und erst anschliessend die Taste *'B'* drückt, dann hat er in ACE die Möglichkeit, die Reihenfolge der koordinierten Konstituenten durch die deterministische Konjunktion *and* festzulegen.

(166) *The borrower presses the key 'A' and the key 'B'.*

Syntaktisch handelt es sich bei der Koordination im Satz (166) um eine elliptische Struktur (Koordinationsreduktion), bei der vorerwähntes Material *[presses]* ausgespart wird. Durch die strukturgerechte Ergänzung der verbalen Information ergibt sich die Konstituente *presses the key 'A' and [presses] the key 'B'*. Dieser Konstituente liegen zwei zeitlich angeordnete Ereignisse zugrunde. Die Übersetzung in die DRT-E führt zu zwei neuen Ereignisvariablen `E1` und `E2` im Diskursuniversum und zu zwei Bedingungen `event(E1,press(I1,I2))` und `event(E2,press(I1,I3))`. Die lineare Darstellung der Information in der DRS legt implizit fest, dass die Ereignisse `E1` und `E2` im Anwendungsbereich in fester Reihenfolge auftreten.

Wenn der Anwendungsspezialist einen Sachverhalt beschreiben möchte, wo der Ausleiher die Tasten *'A'* und *'B'* gleichzeitig drückt, dann verwendet er neben der Konjunktion *and* das (als Spezialfall) an die Verbalphrase (`v2`) adjungierte Diskurssignal *at the same time*.

(167) *The borrower presses the key 'A' and the key 'B' at the same time.*

Die Übersetzung in die DRT-E führt wiederum zu zwei separaten Ereignisvariablen `E1` und `E2`, ausserdem wird das gleichzeitige Auftreten der Ereignisse im Anwendungsbereich durch die Bedingung `at_same_time(E1,E2)` in der DRS angezeigt.

Wenn der Anwendungsspezialist hingegen einen Sachverhalt beschreiben möchte, wo es keine Rolle spielt, in welcher Reihenfolge der Ausleiher die beiden Tasten *'A'* und *'B'* drückt, dann kann er diesen nichtdeterministischen Sachverhalt explizit machen,

indem er die Verbalphrase (`v2`) durch das Diskurssignal *in any temporal order* modifiziert.

(168) *The borrower presses the key 'A' and the key 'B' in any temporal order.*

Auch hier werden zwei Ereignisvariablen `E1` und `E2` in die DRS eingeführt, ausserdem zeigt die Bedingung `any_temporal_order(E1,E2)` an, dass diese Ereignisse im Anwendungsbereich in beliebiger Reihenfolge auftreten können.

Inklusive Disjunktion

Der Disjunktion `v` in der Logik entspricht die inklusive Disjunktion *or* in ACE. Die Bedeutung der inklusiven Disjunktion ist in der natürlichen Sprache durch den Ausdruck *"A or B or both"* paraphrasierbar. Der Anwendungsspezialist kann die inklusive Disjunktion *or* dazu verwenden, um einen Sachverhalt im Anwendungsbereich zu beschreiben, der wahrheitswertig betrachtet auf drei alternative Arten erfüllbar ist. Die koordinierte Struktur im Satz

(169) *The borrower presses the key 'A' or the key 'B'.*

führt nach der Übersetzung in die DRT-E zu einer inklusiven Bedingung der Form `OR` K_1 `OR` K_2, wobei K_1 und K_2 sub-DRSen sind mit den aus den beiden Disjunkten *presses the key 'A'* und *[presses] the key 'B'* abgeleiteten Bedingungen. Die zusammengesetzte Bedingung `OR` K_1 `OR` K_2 ist dann wahr, wenn entweder K_1 wahr ist, aber nicht K_2; oder wenn K_2 wahr ist, aber nicht K_1; oder wenn K_1 und K_2 beide wahr sind.

Exklusive Disjunktion

Im Gegensatz zur Logik wird der Koordinator *or* in der vollen natürlichen Sprache meistens exklusiv, d.h. ausschliessend, verstanden. Durch die exklusive Disjunktion wird in der Regel ein Sachverhalt beschrieben, wo von zwei oder mehr Fällen, die zur Auswahl stehen, nur einer in Betracht kommt. Diese ausschliessende Lesart kann paraphrasiert werden als *"either A or B, but not both"*. Um eine solche Lesart in ACE zu realisieren, muss der Anwendungsspezialist die exklusive Disjunktion *either ... or* anstelle der inklusiven Disjunktion *or* verwenden.

(170) *The borrower either presses the key 'A' or the key 'B'.*

Die vervollständigte Konstituente *either presses the key 'A' or [presses] the key 'B'* führt zu einer zusammengesetzten DRS der Form `EITHER` K_1 `OR` K_2. Diese DRS ist genau dann wahr, wenn eine der beiden sub-DRSen K_1 und K_2 wahr ist. Um das nachzuweisen, müssen die Bedingungen beider sub-DRSen K_1 und K_2 untersucht werden. Zu

diesem Zweck wird die DRS `EITHER` K_1 `OR` K_2 bei der Ausführung der Spezifikation in eine logisch äquivalente Form überführt und modell- bzw. beweistheoretisch interpretiert. Dabei gilt es zu beachten, dass in der DRT-E Diskursreferenten, die neu in eine sub-DRS eingeführt werden, z.B. in K_1, von einer anderen disjunktiv verknüpften sub-DRS aus, z.B. K_2, zugänglich sind.

Die inklusive und die exklusive Disjunktion können bei der Ausführung der Spezifikation zu Nichtdeterminismus führen. Beispielsweise steht für das konditionale Satzgefüge

(171) *If the light is green then the borrower either presses the key 'A' or the key 'B'.*

nicht fest, welche der beiden Alternativen (*presses the key 'A' oder [presses] the key 'B'*) im Hauptsatz ausgewählt werden muss, wenn die Vorbedingung (*the light is green*) im Nebensatz erfüllt ist. Bei der Ausführung der Spezifikation ist es nötig, eine der beiden Alternativen aufgrund einer fairen Strategie auszuwählen [vgl. Fuchs forthcoming].

Vorrangsregeln für Konjunktion und Disjunktion

Bei komplexeren Verknüpfungen mit mehreren unterschiedlichen Koordinatoren benötigt man Vorrangsregeln, die angeben, welche Konstituenten zusammengehören. In der Logik wird die Zusammengehörigkeit der Ausdrücke durch die Bindungsstärke der Koordinatoren festgelegt und im Zweifelsfall durch Klammern verdeutlicht. In ACE gelten dieselben Konventionen für die Bindungsstärke wie in der Logik: Die Konjunktion bindet stärker als die inklusive Disjunktion und die exklusive Disjunktion. Ferner stehen die Koordinatoren in der Rangfolge zwischen der Negation, die am stärksten bindet und der Implikation, die am schwächsten bindet. Dementsprechend gilt für ACE, dass die deterministische Konjunktion *and* stärker bindet als die inklusive Disjunktion *or* und die exklusive Disjunktion *either … or*. Für den Satz

(172) *The borrower presses the key 'A' or the key 'B' and the key 'C'.*

ergibt sich in abgekürzter Schreibweise die Lesart: *"A or (B and C) or both"*. Wenn der Anwendungsspezialist jedoch eine Lesart im Kopf hat, in der die inklusive Disjunktion stärker binden soll als die deterministische Konjunktion, dann kann er diese Lesart durch Kommasetzung erzwingen.

(173) *The borrower presses the key 'A' or the key 'B', and the key 'C'.*

Für den Satz (173) ergibt sich die Lesart: *"(A or B or both) and C"*. Das Komma wird in der Sprache ACE als Diskurssignal interpretiert und hat die gleiche Funktion wie

Klammern in der Logik.

5.4.2.5 Subordinator

Die Kategorie *Subordinator* besteht in ACE aus den Relativpronomen und der konditionalen Konjunktion. Subordinatoren bilden aus einfachen Sätzen zusammengesetzte Sätze und zeigen das Verhältnis an, das zwischen den Teilsätzen besteht. Bei den zusammengesetzten Sätzen handelt es sich um Satzgefüge, die dadurch gekennzeichnet sind, dass ein Nebensatz einem Hauptsatz grammatisch untergeordnet ist. Syntaktisch funktionieren die Subordinatoren als Nebensatzeinleitungen, die in der sogenannten `specC`-Position am Anfang des eingebetteten Nebensatzes stehen. Relativpronomen leiten in ACE restriktive Nebensätze ein, die eine vorausgehende Nominalphrase aus einem übergeordneten Hauptsatz modifizieren. Konditionale Konjunktionen leiten konditionale Nebensätze ein, die die Vorbedingungen für die nachfolgenden Hauptsätze festlegen.

Subordinatoren

Subordinator	Beispiel	Bedingung
Relativpronomen	who, which, that	I_2=I_1
Konditionale Konjunktion	if ... then	`IF` K_1 `THEN` K_2

Relativpronomen

Relativpronomen (`rel`) müssen in ACE immer streng rechtsassoziativ verwendet werden und dürfen nie weggelassen werden, weil reduzierte Relativsätze zu struktureller Ambiguität (d.h. zu sogenannten "garden path sentences") führen können. Das Relativpronomen bezieht sich immer auf die unmittelbar vorausgehende nominale `n1`-Konstituente, die das Antezedens bildet. In Abhängigkeit vom Typ des nominalen Kopfes, der durch den Relativsatz modifizierten `n1`-Konstituente, können die Relativpronomen unterschiedliche Formen annehmen: Das Relativpronomen *who* bezieht sich ausschliesslich auf nominale Köpfe vom Typ `person`; das Relativpronomen *which* auf nominale Köpfe, die nicht vom Typ `person` sind; und beim Relativpronomen *that* spielt der Typ des nominalen Kopfes keine Rolle.

Relativpronomen

Merkmal	Wert	Beispiel
Wortform	`<wform>`	`who`

Merkmal	Wert	Beispiel
Typ	person | object ∨ time | _	person
Bedingung	I_2=I_1	I2=I1

Von der Syntax her funktionieren Relativpronomen ihrerseits als Nominalphrasen (*n2*), die die *specC*-Position besetzen. Die *specC*-Position geht der Subjektposition eines Relativsatzes unmittelbar voraus. Der verbleibende Relativsatz - ohne das Relativpronomen - ist wie ein einfacher ACE Satz gebaut, ausser dass ihm eine Nominalphrase (*n2*) fehlt, die eine Lücke in der syntaktischen Struktur hinterlässt. Die Nominalphrase mit dem Relativpronomen und der verbleibende Relativsatz werden von der Kategorie *s1* dominiert. Zwischen dem Relativpronomen und der Lücke im verbleibenden Relativsatz besteht eine sogenannte ungebundene Abhängigkeitsbeziehung [vgl. Pereira & Shieber 87:118, Pollard & Sag 94:157 ff., Bennett 95:147 ff.]. Das heisst, das Relativpronomen lizenziert im Relativsatz eine nominale Lücke, die nicht lokal gebunden ist und deshalb (theoretisch) beliebig tief eingebettet sein kann. Die Lücke kann nur dann auftreten, wenn das passende Relativpronomen die *specC*-Position besetzt. Phänomene dieser Art bezeichnet man auch als "filler-gap"-Abhängigkeiten, die hier zwischen einem Relativpronomen (filler) und einer fehlenden Nominalphrase (gap) besteht. Abhängigkeitsbeziehungen dieser Art finden sich auch bei einer Teilmenge von Ergänzungsfragen, wo Frageausdrücke als Füller funktionieren und dort eine Lücke lizenzieren, wo die erfragte Konstituente stehen würde.

Im Gegensatz zu Relativsätzen, die in ACE als Folge einer Topikalisierung entstehen und bei denen dem Relativpronomen bloss verknüpfende Funktion zukommt, übernimmt das Relativpronomen in den folgenden beiden zentral eingebetteten Relativsätzen die Funktion eines nominalen Füllers.

(174) *The [n1 [n1 [n0 staff user]] [s1 [n2 [rel_i who]] [t_i enters the password]]] works in the library.*

(175) *The [n1 [n1 alphanumeric password] [s1 [n2 [rel_j which]] [s the staff user enters t_j]]] is valid.*

Das Relativpronomen (rel_i) wird im Satz (174) als Nominalphrase (*n2*) analysiert und bildet den Füller für die nominale Lücke (t_i) in der Subjektposition des Relativsatzes. Ähnlich ist die Sachlage im Satz (175), wo das Relativpronomen (rel_j) den Füller für die nominale Lücke in der Komplementposition (t_j) des Relativsatzes bildet.

In ACE sind nur restriktive Relativsätze zugelassen, die verwendet werden können, um die Menge der möglichen Entitäten im Anwendungsbereich für einen nominalen Individuenreferenten I_1 einzuschränken. Der Relativsatz liefert zusätzliche Information, aus der eine Reihe von Bedingungen abgeleitet werden, die die vorliegende DRS erweitern. Das Relativpronomen selber führt bei der Übersetzung zu einem (temporären) Individuenreferenten I_2 und einer (temporären) Identitätsbedingung *I2=I1*, die den Individuenreferenten I_2 für das Relativpronomen mit dem Individuenreferenten I_1, der für das Antezedens steht, in Beziehung setzt.

Technisch wird die Abhängigkeit zwischen dem pronominalen Füller und der nominalen Lücke im Attempto System durch einen Term der Form *fgd(rel,I2,n2)* hergestellt und durch einen Liste implementiert [vgl. Pereira & Shieber 87:177 ff., Covington 94:60]. Der Term sorgt zum Beispiel bei der Verarbeitung des Satzes (175) dafür, dass die aus der Verbalphrase des Relativsatzes abgeleitete Bedingung *event(E1,enter(I3,I2))* - trotz der fehlenden Nominalphrase in der Komplementposition - durch Unifikation zu ihrem Individuenreferenten *I2* kommt.

Konditionale Konjunktion

Die konditionale Konjunktion *if* leitet einen Nebensatz ein, der mit einem Hauptsatz verfugt ist. Syntaktisch steht die Konjunktion *if* in der sogenannten *comp*-Position und zeigt das konditionale Verhältnis zwischen den Teilsätzen an. Das Korrelat zu *if* bildet das verbindende Adverb *then*, das ebenfalls in einer satzinitialen Position steht und den nachfolgenden Hauptsatz als bedingt markiert. Ein konditionales Satzgefüge muss in ACE immer durch die Oberflächenmarkierung *if … then* gekennzeichnet werden, damit das Antezedens eindeutig von der Konsequenz abgrenzbar ist. Allgemein bestehen konditionale Verknüpfungen nicht nur zwischen zwei Teilsätzen, sondern sie können mehrere Sätze umfassen. Ein Beispiel dafür ist das folgende konditionale Satzgefüge:

(176) *If a copy is checked out to a borrower and a staff user returns the copy of the book then the copy is available.*

Im einleitenden Nebensatz wird ein komplexer Sachverhalt dargestellt, der vorausgesetzt wird und erfüllt sein muss, damit der im Hauptsatz beschriebene (bedingte) Sachverhalt zutrifft. Die Übersetzung dieses Satzgefüges führt in der DRT-E zu einer implikativen Bedingung der Form *IF* K_1 *THEN* K_2. Diese Bedinungung ist genau dann wahr, wenn jede Einbettung der sub-DRS K_1 in ein Modell auf eine Art und Weise erweitert werden kann, dass auch die sub-DRS K_2 im Modell erfüllt ist. Abgesehen von

Individuenreferenten für Eigennamen sind Individuenreferenten, die neu in eine implikative Bedingung eingeführt werden, von aussen her nicht anaphorisch zugänglich.

5.4.2.6 Fragewort

Die Kategorie *Fragewort* wird aufgrund der syntaktischen Leistung der einzelnen Mitglieder in den Fragesätzen in vier Teilkategorien unterteilt: *Operator, Fragepronomen, Fragedeterminator* und *Frageadverb*. Der Operator *does* spielt sowohl bei der Bildung von Entscheidungs- als auch von Ergänzungsfragen eine zentrale Rolle. Die Mitglieder der restlichen drei Teilkategorien werden nur bei der Bildung von Ergänzungsfragen mitverwendet. Mit den Entscheidungs- und Ergänzungsfragen sind spezielle Frageausdrücke bzw. Satzkonstruktionen verbunden, die helfen eine Anforderungsspezifikation zu validieren. Fragesätze werden im Rahmen der DRT-E in sogenannte Frage-DRSen übersetzt, die durch einen speziellen Operator gekennzeichnet sind. Für die Fragebeantwortung wird die aus der Spezifikation abgeleitete DRS durch Inferenz ausgewertet. Das Ergebnis der Auswertung ist im Fall einer Entscheidungsfrage eine positive oder negative Antwort. Im Fall von Ergänzungsfragen wird zusätzlich versucht, die aus dem Frageausdruck abgeleitete Variable und die womöglich dazugehörige Fragebedingung durch die relevante Information aus der vorliegenden DRS (oder Prolog Wissensbasis) zu ersetzen, so dass ein vollständiger ACE Satz als Antwort erzeugt werden kann, der die erfragte Information beinhaltet.

Fragewörter

Fragewörter	Beispiel	Bedingung
Operator	does	`YNQ K1`
Fragepronomen	who	`WHQ [[Who], …]`
Fragedeterminator	which	`WHQ [[Which], …]`
Frageadverb	where	`WHQ [[Where], …] location(E,Where)`

Frage-DRSen werden beweistheoretisch interpretiert. Eine Frage-DRS ist dann wahr, wenn sie eine logische Konsequenz der spezifizierten Sachverhalte ist, die in einer DRS formal vorliegen. Das heisst, die Wahrheit einer Entscheidungsfrage wird durch logische Inferenz aus einer DRS bewiesen. Bei einer Ergänzungsfrage werden die Variable für den Frageausdruck zusammen mit den Fragebedingungen während des Beweises durch die entsprechenden Bedingungen aus der vorliegenden DRS substituiert. Das Resultat der Berechnung ist die Grundlage für die Fragebeantwortung in ACE.

Frageausdrücke in Entscheidungsfragen

Entscheidungsfragen werden durch Subjekt-Hilfsverb-Inversion gebildet (*Is the password valid?*) oder, wenn die Verbalphrase kein Hilfsverb aufweist, durch *do*-Einsetzung und Subjekt-Operator-Inversion (*Does John own a password?*). Ausser dem Operator *does* verlangen Entscheidungsfragen keine zusätzlichen Frageausdrücke. Mit Entscheidungsfragen kann überprüft werden, ob ein Sachverhalt in einer Anforderungsspezifikation besteht oder nicht. Entscheidungsfragen liefern als Ergebnis eine positive Antwort (*yes*) oder eine negative Antwort (*no*). Die klassische DRT behandelt Fragesätze nicht. Bei der Übersetzung in die DRT-E entsteht aus einer Entscheidungsfrage eine Frage-DRS, die durch den Operator `YNQ` markiert ist. Bei der Fragebeantwortung müssen die einzelnen Bedingungen, die aus dem Fragesatz abgeleitet worden sind, aufgrund der vorliegenden DRS bewiesen werden.

Frageausdrücke in Ergänzungsfragen

Ergänzungsfragen werden in ACE durch Fragepronomen (z.B. *Who owns a password?*), Fragedeterminatoren (z.B. *Which copy carries a number?*) oder Frageadverbien (z.B. *Where does the staff user work?*) gebildet. Die meisten Ergänzungsfragen weisen eine nominale Lücke in der Satzstruktur auf, die mit einem Frageausdruck assoziiert werden kann, der als Füller in der satzexternen `specC`-Position steht. Durch Linksextraposition gebildete Fragesätze können ähnlich wie restriktive Relativsätze analysiert werden. Ergänzungsfragen stellen nicht einen ganzen Sachverhalt in Frage, sondern erkunden mit dem Frageausdruck bloss einen Teilaspekt dieses Sachverhaltes. Die Fragepronomen *who* und *what* fragen nach Information, die zuvor durch ein externes oder internes Komplement in den Spezifikationstext eingeführt worden ist. Die Fragedeterminatoren *which* und *whose* fragen nach Information, die entweder durch ein pränominales Attribut (Adjektivphrase), ein postnominales Attribut (Relativsatz), eine Apposition (Quoted String) oder ein postnominales Komplement (*of*-Konstruktion) in einem Satz der Spezifikation ausgesagt worden ist. Die Frageadverbien *where, when, how* usw. beziehen sich auf Information, die durch verbale Adjunkte (Adverbialphrasen oder Präpositionalphrasen) in den Spezifikationstext gelangt ist.

Ergänzungsfragen werden in Frage-DRSen übersetzt, die durch den Operator `WHQ` markiert sind. Aus den Frageausdrücken wird eine speziell gekennzeichnete Variable der Form `[WH]` abgeleitet und zu den Diskursreferenten hinzugefügt. Zusätzlich zu dieser Information wird für die Frageadverbien im Lexikon eine Bedingung der Form `M(E,WH)` bereitgestellt, um den Modifikatortyp `M` mit der Eventualität `E` bei der Auswertung eindeutig identifizieren zu können. Für die Fragebeantwortung werden

im Fall von Ergänzungsfragen nicht nur die einzelnen Bedingungen bewiesen, sondern die erfragte Information wird ausserdem in einem ACE Satz als Antwort aufbereitet.

5.5 Einfache Sätze in ACE

In diesem Kapitel wird der Aufbau von einfachen ACE Sätze aus einer informellen Perspektive besprochen. Zu diesem Zweck verfolgen wir einen monostralen Ansatz, wo die syntaktische Struktur eines eindeutigen Satzes immer durch ein einziges Baumdiagramm dargestellt werden kann. Es gibt keine Bewegungen von Konstituenten, wie das aus der Tradition der Transformationsgrammatik bekannt ist [vgl. Chomsky 65, Chomsky 86]. Anstelle von Bewegungen sorgen Merkmalstrukturen als Argumente von Phrasenstrukturregeln für den Informationsfluss, der durch Merkmalunifikation zustande kommt [vgl. Gazdar et al. 85, Shieber 86, Covington 94b, Bennett 95]. Auch Unifikationsgrammatiken gehen bei der Definition der syntaktischen Kategorien von einer Version der X-bar-Theorie aus [vgl. Kornai & Pullum 90, Green & Morgan 96:103]. Bei diesen unifikationsbasierten Ansätzen sind die phrasalen Kategorien sensitiv für die Merkmalstrukturen der lexikalischen Köpfe und deren Projektionen. Die Implementation von ACE beruht auf einer unifikationsbasierten Phrasenstrukturgrammatik, in der die linguistischen Objekte durch linearisierte Merkmalstrukturen modelliert werden [vgl. Fuchs & Schwitter 95].

Für die Beschreibung der zulässigen ACE Sätze wird in der Folge von den Merkmalstrukturen und den Implementationsdetails abstrahiert. Es wird gezeigt, wie einfache ACE Sätze gebaut sind, welche obligatorischen oder optionalen Phrasen zur Konstruktion verwendet werden können, welche Wörter sich zu Phrasen gruppieren lassen und welche Wörter welche anderen Wörter modifizieren können. Das Resultat sind einfache wohlgeformte ACE Sätze, für deren Konstruktion sich dann eine Reihe von leicht erlernbaren Schreibprinzipien formulieren lässt.

Den Ausgangspunkt bildet ein abstrakter Satzbauplan, der einen einfachen ACE Satz als eine flache syntaktische Struktur beschreibt, die eine (relativ) feste Ordnung hat und aus den folgenden fünf Funktionen besteht:

Subjekt + [{ Negation } + verbaler Kopf + Komplement { + Adjunkt }]

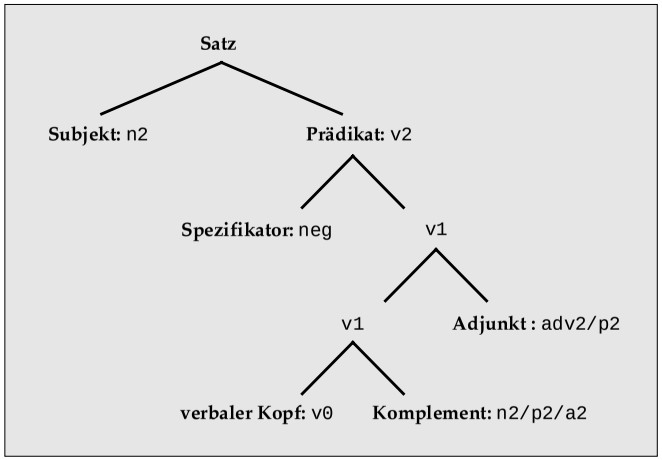
Bei diesem Satzbauplan handelt es sich um ein mnemotechnisches Hilfsmittel, das den Anwendungsspezialisten beim Schreiben der Spezifikation unterstützen soll, wohlgeformte ACE Sätze zu bilden. Der Satzbauplan besteht aus zwei Hauptteilen: dem Subjekt und dem "entfalteten" Prädikat in eckigen Klammern. Diese Darstellung impliziert eine Subjekt-Prädikat-Asymmetrie, die aus den folgenden Gründen motiviert ist: Der verbale Kopf hat aufgrund der Selektionseigenschaften des Verbs die Fähigkeit seine

syntaktische Umgebung zu determinieren. Die Selektionseigenschaften determinieren nicht nur die Wahl der verbalen Komplemente, sondern auch die des Subjekts. Im Gegensatz zu einem verbalen Komplement wird das Subjekt jedoch unmittelbar durch den Satzknoten dominiert und bildet eine externe Ergänzungen. Das Komplement ist dem Subjekt strukturell nicht gleichgestellt. Es bildet ein interne Ergänzungen und wird als Schwester des verbalen Kopfes von einem *v1*-Knoten dominiert.

Neben dem verbalen Kopf und dem Komplement enthält das Prädikat im dargestellten Satzbauplan zwei optionale Funktionen, die durch geschweifte Klammern angezeigt werden. Im einen Fall handelt es sich um ein Adjunkt bzw. um ein Reihe von rekursiv eingeführten Adjunkten, die einen *v1*-Knoten unterschiedlich modifizieren können. Im anderen Fall handelt es sich um Negation bzw. um den Spezifikator des Prädikats, der durch Elemente aus der Kategorie Negationswort (*neg*) realisiert sein kann.

Bevor die zulässigen Realisierungen der einzelnen Funktionen bzw. Positionen in ACE besprochen werden, stellen wir den abstrakten Satzbauplan in Form eines Baumdiagramms dar, um die zugrundeliegende hierarchische Struktur und die Subjekt-Prädikat-Asymmetrie zu verdeutlichen.

Struktur einfacher ACE Sätze



Das Subjekt eines einfachen ACE Satzes kann nur durch eine Nominalphrase (*n2*) realisiert werden. Das Prädikat besteht aus einer Verbalphrase (*v2*) mit einem finiten

Verb als verbalem Kopf (*v0*). Das Verb kann aufgrund seiner Selektionseigenschaften eine oder mehr Nominalphrasen (*n2*), eine Präpositionalphrase (*p2*) oder eine Adjektivphrase (*a2*) als Komplemente selektionieren. Eine Verbalphrase kann eine Reihe von Adjunkten enthalten, die nicht vom Verb selektioniert werden und deshalb weniger stark in die phrasale Struktur eingebunden sind. Die Adjunkte können durch Adverbialphrasen (*adv2*) oder auch durch Präpositionalphrasen (*p2*) realisiert sein. In ACE folgen die Adjunkte dem Komplement im Normalfall unmittelbar. Alternativ kann ein einzelnes Adverb jedoch auch direkt vor dem verbalen Kopf stehen. Bei (interner) Negation ist der Spezifikator durch ein Negationswort (*neg*) der Form *does not* oder *not* besetzt. Ein Sonderfall ist das kopulative Verb *be*, dem der Negationsausdruck *not* folgt. Die untenstehenden Tabellen illustrieren anhand einiger Beispiele die zulässigen Realisierungen des Satzbauplans.

Realisierung des Satzbauplans

Subjekt	Spezifikator	verbaler Kopf	Komplement	Adjunkt
[n2 a staff user]		[v0 [tv removes]]	[n2 the copy]	[p2 from the library]
[n2 [pn John]]	[neg does not]	[v0 [tv enter]]	[n2 the password]	[adv2 [adv0 slowly]]

Alternative Stellung von Adverbien

Subjekt	Spezifikator	Adjunkt	verbaler Kopf	Komplement
[n2 [pn John]]		[adv2 [adv0 slowly]]	[v0 [tv enters]]	[n2 the password]
[n2 [pn John]]	[neg does not]	[adv2 [adv0 slowly]]	[v0 [tv enter]]	[n2 the password]

Stellung des Negationswortes *not*

Subjekt	verbaler Kopf	Spezifikator	Komplement	Adjunkt
[n2 the library]	[v0 [cop is]]	[neg not]	[a2 [a0 open]]	[p2 on Monday]

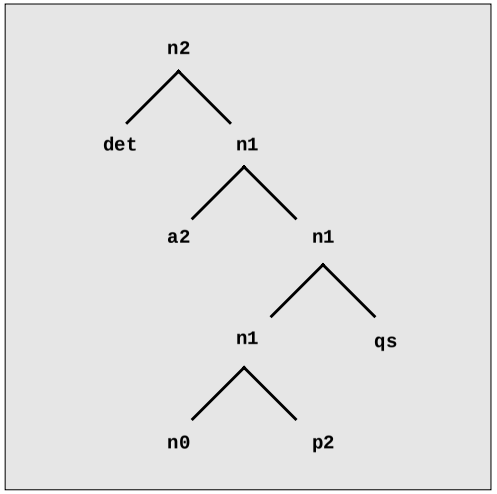
Nach der Strukturbeschreibung einfacher ACE Sätze werden nun die Beziehungen zwischen den lexikalischen und phrasalen Kategorien genauer untersucht. Insbesondere interessiert, welchen strukturellen Beschränkungen die einzelnen Phrasen unterworfen sind.

5.5.1 Nominalphrase

Ausser als Subjekt können Nominalphrasen (*n2*) in ACE auch als interne Komplemente von verbalen Köpfen (*v0*) oder als Komplemente von präpositionalen Köpfen (*p0*) realisiert sein. Eine prototypische Nominalphrase besteht aus einer Gattungsbezeichnung (*cn*) als nomialem Kopf (*n0*) zusammen mit einem Determinator (*det*) als Spezifikator (z.B. *a/the/every/no staff user*).

Bestimmte nominale Köpfe können durch optionale Konstituenten modifiziert werden. Dazu gehören auf der einen Seite Adjektivphrasen (*a2*), die als Attribute funktionieren und vor einem nominalen Kopf stehen. Auf der anderen Seite sind das Elemente der Kategorie Quoted String (*qs*), die als restriktive Appositionen verwendet werden können und einem nominalen Kopf nachgestellt sind. Zu den Komplementen von nominalen Köpfen zählen in ACE die *of*-Konstruktionen, die durch Präpositionalphrasen (*p2*) realisiert werden müssen, wenn die Nomen relational gebrauch werden.

Struktur der Nominalphrase



Der nominale Kopf einer Nominalphrase besteht in ACE entweder aus einer Gattungsbezeichnung (*cn*), einem Massennomen (*mn*), einem Eigennamen (*pn*) oder einem Personalpronomen (*pro*). Wenn der nominale Kopf durch eine Gattungsbezeichnung realisiert ist, dann muss der Spezifikator immer durch einen Determinator (*det*) realisiert sein. Die Gruppe der Determinatoren setzt sich zusammen aus den indefiniten

Determinatoren (*a/an, there is a*), dem definiten Determinator (*the*), dem possessiven Determinator (*N's*), den universellen Determinatoren (*each, every, for every*) und den negativen Determinatoren (*no, there is no*). Massennomen können nicht mit indefiniten Determinatoren verwendet werden. Die obligatorischen Determinatoren stehen immer vor den optionalen Adjektivphrasen (*a2*), die einen nominalen Kopf modifizieren können, wenn es sich beim Kopf, um eine Gattungsbezeichnung oder um ein Massennomen handelt. Das vorangestelle Attribut kann selber eine Attribut-Kopf-Struktur haben, da Adjektive in ACE durch Gradadverbien (*more, most*) modifiziert werden können. Wenn ein Nomen als nominaler Kopf relational verwendet wird, dann selektioniert es in ACE ein Komplement, das durch eine Präpositionalphrase (*p2*) realisiert werden muss. Bei der Präpositionalphrase kann es sich nur um eine *of*-Konstruktion handeln. Im Gegensatz zu Adjektivphrasen dürfen *of*-Konstruktionen in ACE nicht koordiniert werden, da solche Konstruktionen zu Mehrdeutigkeit führen können (vgl. *the father of Alan and of Mel* und *the book of Hofstadter and of Kleist*). Ein Spezialfall ist die Kategorie Quoted String (*qs*), die beliebige Zeichenketten umfasst, welche durch einfache Anführungszeichen gekennzeichnet sind. Solche Zeichenketten sind immer optional und werden dem nominalen Kopf direkt nachgestellt. Quoted Strings funktionieren als restriktive Appositionen und beschreiben das Nomen genauer.

Die folgende Tabelle zeigt exemplarisch den Aufbau von zulässigen Nominalphrasen. Syntaktische Funktionen mit dunklem Hintergrund sind nicht realisierbar, da solche Belegungen zu ungrammatischen Strukturen führen würden. Alle nichtbelegten syntaktischen Funktionen mit hellem Hintergrund sind im Prinzip realisierbar.

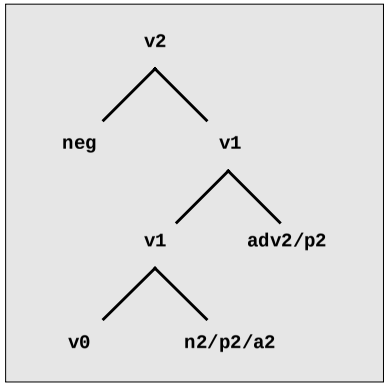
Realisierung der Nominalphrase

Spezifikator	Attribut	Kopf	Apposition	Komplement
[det a]		[n0 [cn password]]		
[det no]	[a2 [a0 correct]]	[n0 [cn password]]		
[det the]		[n0 [cn password]]	[qs '666']	
[det the]		[n0 [cn password]]	[qs '666']	[p2 of John]
[det John's]		[n0 [cn password]]	[qs '666']	
[det the]		[n0 [mn metal]]		
		[n0 [pn LibDB]]		
		[n0 [pro he]]		

5.5.2 Verbalphrase

Das Prädikat des Satzbauplans besteht aus einem verbalen Kopf (*v0*), der das strukturelle Zentrum der Verbalphrase (*v2*) bzw. des ganzen ACE Satzes bildet. Bei den vom verbalen Kopf abhängigen Elementen kann es sich um Komplemente, Adjunkte und Spezifikatoren handeln. Die Komplemente werden aufgrund der Selektionseigenschaften des Verbs ausgewählt. Bei den Komplementen kann es sich entweder um Nominalphrasen (*n2*), eine Präpositionalphrase (*p2*) oder eine Adjektivphrasen (*a2*) handeln. Andere Realisierungen wie beispielsweise Partizipialkonstruktionen, Infinitivkonstruktionen oder Satzkomplemente sind in ACE nicht zulässig. Im Gegensatz zu den Komplementen hängt die Wahl der verbalen Adjunkte nicht von den Selektionseigenschaften des Verbs ab, d.h. Adjunkte können aus strukturellen Gesichtspunkten weggelassen werden. Bei den verbalen Adjunkten kann es sich um eine Adverbialphrase (*adv2*) oder auch um eine Präpositionalphrase (*p2*) handeln. Die Elemente der Kategorie Negationswort (*neg*) funktionieren als Spezifikatoren von Verbalphrasen und verneinen die Prädikation. Zur Verneinung wird der Negationsausdruck *does not* zusammen mit einem Vollverb verwendet oder der Negationsausdruck *not* zusammen mit dem kopulativen Verb *be*.

Struktur der Verbalphrase



Der verbale Kopf (*v0*) kann in ACE aus einem intransitiven Verb (*iv*), aus einem transitiven Verb (*tv*), aus einem ditransitiven Verb (*dv*) oder aus einem kopulativen Verb (*cop*) bestehen. Die zulässigen finiten Verben sind durch die folgenden Merkmale gekennzeichnet: 3. Person, Singular, Indikativ, Aktiv und *simple present tense*. Modale Verben (*might, may, could, can, should, ought to, would, will, must*) und intensionale

Verben (z.B. *believe, know, hope*, usw.) sind in ACE nicht erlaubt, da ihre Verwendung nicht zu intersubjektiv überprüfbaren Aussagen führt.

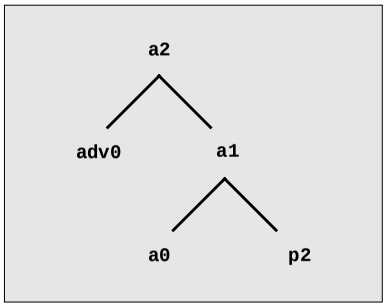
Realisierung der Verbalphrase

Kopf	Komplement(e)	Adjunkt
[v0 [cop is]]	[a2 [a0 valid]]	[p2 on Monday]
[v0 [cop is]]	[n2 the staff user]	
[v0 [cop is]]	[p2 in the slot]	
[v0 [cop is]]	[a2 smaller than the actual amount]	
[v0 [iv works]]		[p2 from Monday]
[v0 [tv enters]]	[n2 the password]	[adv2 [adv0 slowly]]
[v0 [dv adds]]	[n2 a copy] [p2 to the libraray]	[p2 on Monday]

5.5.3 Adjektivphrase

Adjektivphrasen (*a2*) funktionieren entweder als pränominale Attribute in Nominalphrasen (attributive Stellung) oder als Komplemente in kopulativen Konstruktionen (prädikative Stellung). Prädikative Adjektivphrasen sind Teil der Verbalphrase und durch das kopulative Verb *be* mit dem Subjekt verbunden. Adjektivphrasen haben eine eigene interne Struktur, die aber weniger komplex ist, als das beispielsweise bei Verbalphrasen der Fall ist. Adjektive können zur Bezeichnung der Ungleichheit zweier Entitäten oder zur Bezeichnung eines absoluten Grades als adjektivische Köpfe (*a0*) von Adjektivphrasen durch Flexionsmorpheme oder Gradadverbien (*adv0*) modifiziert werden. Komparative Adjektive und zweistellige Adjektive selektionieren Komplemente, die formal durch Präpositionalphrasen (*p2*) realisiert sind.

Struktur der Adjektivphrase



Von einer kleinen Menge von Adjektiven abgesehen (z.B. *first, next, last*) sind prototypische Adjektive gradierbar. ACE lässt nur einfache Vergleichsformen zum Ausdruck von komparativer und superlativer Gradangabe zu. Gradmodifikation kann entweder durch die Flexionsmorpheme *-er* und *-est* oder durch die Gradadverbien *more* und *most* angegeben werden. Andere Formen von Modifikation, die von der natürlichen Sprache her vertraut sind, wie etwa Gradadverbien (z.B. *so, only, too*), die als Spezifikatoren funktionieren, oder Intensitätsadverbien (z.B. *very, absolute, utterly*), die als Attribute funktionieren, sind in ACE nicht zulässig. Auch Modfikatoren, die einem adjektivischen Kopf nachgestellten sind (z.B. *in some way*), dürfen in ACE nicht verwendet werden.

Realisierung des attributiven Adjektivs

Attribut	adjektivischer Kopf
	[a0 correct]
	[a1 correcter]
	[a1 correctest]
	[a0 expensive]
[adv0 more]	[a0 expensive]
[adv0 most]	[a0 expensive]

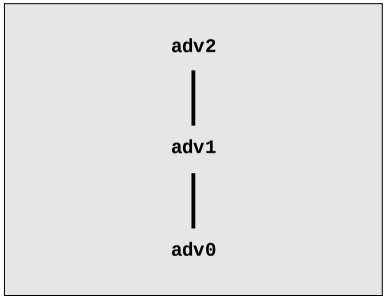
Realisierung des prädikativen Adjektivs

Attribut	adjektivischer Kopf	Komplement
	[a0 high]	
	[a1 higher]	[p2 than the maximum amount]
	[a0 expensive]	
[adv0 more]	[a0 expensive]	[p2 than a copy]
[adv0 more]	[a0 expensive]	
	[a0 close]	[p2 to the desk]

5.5.4 Adverbialphrase

Adverbialphrasen (*adv2*) haben in ACE keine interne Struktur, da nur einzelne Adverbien als Adjunkte von *v1*-Konstituenten zulässig sind. Die adverbialen Köpfe (*a0*) sind also direkt phrasal verwendungsfähig. Die Vorteile, die durch eine phrasale Kategorisierung der Adverbien entstehen, werden bei der Definition der phrasalen Koordination ersichtlich.

Struktur der Adverbialphrase

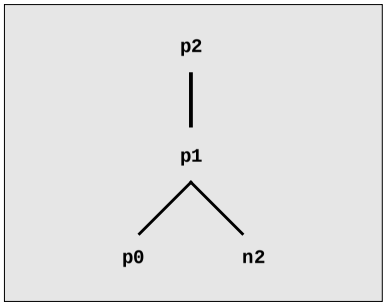


Adverbien stehen in ACE im Normalfall unmittelbar hinter den verbalen Komplementen (z.B. *enters the password slowly*). Alternativ kann ein einzelnes Adverb auch vor dem verbalen Kopf stehen (z.B. *slowly enters the password*). Im Gegensatz zur englischen Standardsprache sind satzmodifizierende Adverbien in ACE nicht zulässig. Ausserdem dürfen Adverbien in ACE nicht durch andere Gradadverbien oder Intensitätsadverbien modifiziert werden.

5.5.5 Präpositionalphrase

Präpositionalphrasen (*p2*) können in ACE als postnominale Komplemente, als Komplemente von komparativen und zweistelligen Adjektiven in Adjektivphrasen, als Komplemente von Verben in Verbalphrasen oder als Adjunkte von *v1*-Konstituenten realisiert werden. Die Präpositionen (*p0*) funktionieren als lexikalische Köpfe der Präpositionalphrasen und selektionieren Nominalphrasen (*n2*) als Komplemente.

Struktur der Präpositionalphrase



Im Gegensatz zur englischen Standardsprache, wo die präpositionalen Köpfe durch Gradadverbien und Intensitätsadverbien modifiziert werden können (z.B. *so completely beyond the limit*), ist diese Art von Modifikation in ACE nicht zulässig.

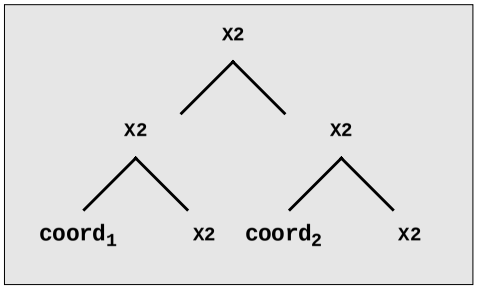
Realisierung der Präpositionalphrase

Kopf	Komplement
[p0 of]	[n2 the book]
[p0 into]	[n2 the slot]
[p0 than]	[n2 the maximum amount]

5.5.6 Phrasale Koordination

Phrasale Koordination ist in ACE nur zwischen vollständigen phrasalen Konstituenten (*X2*) zulässig, die vom gleichen syntaktischen Status sind (z.B. *v2* und *v2*). In ACE sind drei unterschiedliche Koordinatoren (*coord*) verfügbar: Konjunktion (*cj*), inklusive Disjunktion (*dj*) und exklusive Disjunktion bzw. Alternation (*al*). Scheinbar führt die Verwendung der exklusiven Disjunktion *either … or* zu einer anderen syntaktischen Struktur, als das bei der Konjunktion *and* und bei der inklusiven Disjunktion *or* der Fall ist. Der Koordinator *either* steht vor der ersten phrasalen Konstituente (*X2*) und der Koordinator *or* vor allen weiteren phrasalen Konstituenten (*X2*) (z.B. *either presses the key 'A' or the key 'B' or the key 'C'*). Um phrasale Koordination unter einem einheitlichen strukturellen Schema behandeln zu können, wird davon ausgegangen, dass der erste Koordinator ($coord_1$) auch leer sein kann. Der zweite Koordinator ($coord_2$) und alle nachfolgenden Koordinatoren werden dann iterativ eingeführt.

Struktur der phrasalen Koordination



Koordination ist in ACE nur durch syndetische Konstruktionen realisierbar, wo die einzelnen Konstituenten durch explizite Koordinatoren untereinander verbunden sind. Asyndetische Konstruktionen, bei denen die einzelnen koordinierten Konstituenten nur durch Kommas abgetrennt sind und aufgezählt werden, sind in ACE nicht zulässig. Das Komma hat in ACE eine Sonderfunktion. Es dient als Diskurssignal und kann im Fall von komplexen Verknüpfungen verwendet werden, um die vorliegende Bindungshierarchie der Koordinatoren aufzulösen (z.B. *A or B, and C* anstelle von *A or B and C*, wenn die Vorrangsregel *A or (B and C)* gilt).

Koordination von Nominalphrasen in der Subjektposition und von postnominalen Präpositionalphrasen (*of*-Konstruktionen) ist in ACE zur Zeit nicht zulässig, da diese Art von Koordination zu Pluralphänomenen führen kann.

Bei Koordination von Nominalphrasen in der Komplementposition wird das Verb und allenfalls der Negationsausdruck während der Verarbeitung in die koordinierten Konstituenten ausdistribuiert und das Resultat in der Paraphrase angezeigt (z.B. *The borrower presses the key 'A' and the key 'B'* ⇒ *The borrower presses the key 'A' and [presses] the key 'B'*). Daraus folgt, dass Koordination von Nominalphrasen in ACE immer auf Koordination von vollständigen Verbalphrasen zurückgeführt wird.

Natürlich-sprachliche Koordination unterscheidet sich von Koordination in der Logik, wo die Konjunkte kommutativ sind (z.B. $A \wedge B \equiv B \wedge A$). Die textuelle Reihenfolge der Konstituenten bei Konjunktion ist in ACE entscheidend, weil aufgrund dieser Reihenfolge die zeitliche Anordnung der zugrundeliegenden Ereignisse abgeleitet wird. Die zeitliche Anordnung der Ereignisse kann in ACE durch Diskurssignale modifiziert werden. Um Gleichzeitigkeit der Ereignisse auszudrücken, muss das an die Verbalphrase (*v2*) adjungierte Diskurssignal *at the same time* verwendet werden. Um Nichtdeterminismus der Ereignisse auszudrücken, muss das Diskurssignal *in any temporal order* an die Verbalphrase (*v2*) adjungiert werden.

Realisierung der phrasalen Koordination

Koordinator	Konstituente	Koordinator	Konstituente
	[n2 the key 'A']	[coord [cj and]]	[n2 the key 'B']
	[v2 enters a password]	[coord [dj or]]	[v2 enters a code]
[coord [al either]]	[v2 enters a password]	[coord [al or]]	[v2 enters a code]

Koordinator	Konstituente	Koordinator	Konstituente
	[a2 [a0 correct]]	[coord [cj and]]	[a2 [a0 valid]]
	[adv2 [adv0 slowly]]	[coord [dj or]]	[adv2 [adv0 manually]]
[coord [al either]]	[adv2 [adv0 slowly]]	[coord [al or]]	[adv2 [adv0 manually]]
	[p2 on the paper]	[coord [cj and]]	[p2 on the screen]

5.6 Zusammengesetzte Sätze in ACE

Ausgehend von einfachen ACE Sätzen können mit Hilfe von Koordinatoren oder auch Subordinatoren zusammengesetzte ACE Sätze gebildet werden. Koordinierte ACE Sätze setzen sich aus einfachen ACE Sätzen zusammen, die durch einen oder mehr Koordinatoren verbunden sind. Der Koordinator steht jeweils in einer satzexternen Position direkt vor dem nachfolgenden Teilsatz. Subordinierte Sätze sind aus einfachen ACE Sätzen abgeleitete Nebensätze, die aus einem einleitenden Subordinator und einem abhängigen Satzkomplement bestehen. Das Satzkomplement ist entweder ein einfacher ACE Satz, ein koordinierter ACE Satz oder ein unvollständiger ACE Satz.

Der mnemotechnische Satzbauplan für einfache ACE Sätze kann um zwei satzexterne Funktionen bzw. Positionen erweitert und verallgemeinert werden, so dass sich die Struktur von koordinierten und subordinierten ACE Sätzen (und später auch von Fragesätzen) durch ein einheitliches Schema beschreiben lässt. In einem ersten Schritt wollen wir die beiden satzexternen Positionen informell durch die beiden Buchstaben *W* und *C* kennzeichnen. Die *W*-Position kann durch eine *wh*-Phrase (Relativpronomen oder Frageausdruck) besetzt sein, die eine phrasale Lücke im verbleibenden Satzkomplement lizenziert. Die *C*-Position kann entweder durch einen Complementizer (Subordinator oder Operator) realisiert sein, von dem ein Satzkomplement abhängt, oder aufgrund von struktureller Analogie kann an dieser Position auch ein Koordinator stehen. Diesen Überlegungen zufolge sieht die Revision des Satzbauplans in seiner abstraktesten Form folgendermassen aus:

W + C + Satzkomplement

Theoretisch verbirgt sich hinter diesem flachen Satzbauplan das X-bar-Schema für eine Complementizer-Phrase (`s1`). Der Buchstabe *W* steht für die `specC`-Position, der Buchstabe *C* für die `comp`-Position und der Ausdruck *Satzkomplement* für die Satzexpansion `s`. Es werden nun die in ACE zulässigen Belegungen für diesen Satzbauplan diskutiert.

5.6.1 Subordinierte Sätze

ACE kennt zwei unterschiedliche Formen von subordinierten Sätzen: Konditionalsätze und restriktive Relativsätze. Konditionalsätze bilden zusammen mit einem übergeordneten Hauptsatz ein konditionales Satzgefüge. Restriktive Relativsätze sind in ACE mit nominalen Ausdrücken aus einem übergeordneten Satz verfugt. Gemeinsam ist diesen beiden Nebensatzarten, dass sie in ACE explizit durch einen (konditionalen

oder relativen) Subordinator eingeleitet sein müssen. Von ihrer internen Struktur her und aufgrund ihrer inhaltlichen Funktion unterscheiden sich die beiden Nebensatzarten jedoch wesentlich.

Konditionalsätze (*s1*) können in ACE nur durch die konditionale Konjunktion *if* eingeleitet werden. Diese subordinierende Konjunktion steht in der *comp*-Position und nimmt einen Satz (*s*) als Komplement, der entweder die Oberflächenstruktur eines einfachen ACE Satzes oder eines koordinierten ACE Satzes hat. (Der Spezifikator *specC* ist im Fall von Konditionalsätzen nicht besetzt.) Inhaltlich legt ein konditionaler Nebensatz, die Bedingungen fest, unter denen der im übergeordneten Hauptsatz beschriebene Sachverhalt zutrifft.

Restriktive Relativsätze (*s1*) werden in ACE durch die Relativpronomen *who, which* oder *that* eingeleitet. Diese Relativpronomen funktionieren normalerweise als Nominalphrasen (*n2*) und stehen aufgrund ihrer syntaktischen Eigenschaften in der *specC*-Position. Beim verbleibenden Relativsatz (*s*) handelt es sich um einen unvollständigen ACE Satz, der das Satzkomplement bildet. (Die *comp*-Position ist im Fall von Relativsätzen nicht besetzt). Das Relativpronomen lizenziert als Füller eine nominal Lücke in der Struktur des verbleibenden Relativsatzes. Die nominale Lücke kann im Prinzip beliebig tief eingebettet sein, da es sich hier um eine Form von ungebundener Abhängigkeit handelt. Restriktive Relativsätze funktionieren in ACE ausschliesslich als postnominale Modifikatoren in Nominalphrasen von einfachen ACE Sätzen. Inhaltlich modifizieren Relativsätze eine *n1*-Konstituente, indem sie den Bezugsbereich desjenigen Individuenreferenten durch zusätzliche Bedingungen einschränken, der aus dem nominal Ausdruck abgeleitet werden kann.

Realisierung von subordinierten Sätzen

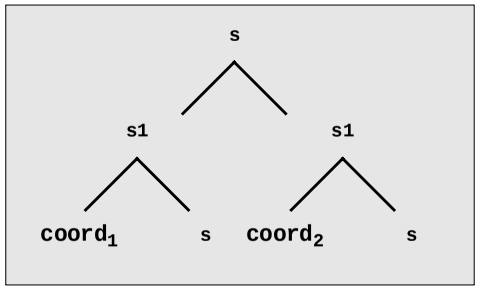
specC	comp	Satzkomplement
	[kcj if]	[s the staff user enters the password]
[n2 [rel which]]		[s the staff user enters [t]]

Einen Sonderfall bilden Relativsätze, die als Resultat einer Quantorenanhebung durch Topikalisierung entstehen, und zwar deshalb, weil es sich bei der zurückbleibenden nominalen Lücke im entstehenden Relativsatz um eine Form von lokaler Abhängigkeit handelt. Das sieht man daran, dass bei der Topikalisierung von universell quantifizierten Nominalphrasen eine nominale Anapher im Relativsatz zurückbleibt (z.B. *For every book there is a number that the staff user assigns to the book).* Das Relativpronomen wird in solchen Fällen nicht als Nominalphrase interpretiert, die in der *specC*-Position steht und Füllerfunktion hat, sondern als syntaktisches Bindeglied verstanden, das in der *comp*-Position steht. Der eigentliche Füller ist die topikalisierte Nominalphrase bzw. sind die topikalisierten Nominalphrasen.

5.6.2 Koordinierte Sätze

In ACE können einfache Sätze koordiniert werden. Zu diesem Zweck stehen die drei Koordinatoren *and, or* und *either … or* zur Verfügung. Strukturell unterscheidet sich Satzkoordination nicht wesentlich von phrasaler Koordination in ACE. Die Koordinatoren stehen in der satzexternen *comp*-Position und bilden zusammen mit einem nachfolgenden Teilsatz *s* einen komplexeren Teilsatz der Kategorie *s1*. Zwei Teilsätze der Kategorien *s1* werden jeweils von einer übergeordneten Satzkategorie *s* dominiert. Wie im Fall von phrasaler Koordination ist bei Konjunktion oder bei inklusiver Disjunktion der erste Koordinator $coord_1$ leer und nur der zweite Koordinator $coord_2$ realisiert. Bei exklusiver Disjunktion ist sowohl der erste als auch der zweite Koordinator realisiert.

Struktur von koordinierten Sätzen



Im Gegensatz zu phrasaler Koordination - wo beispielsweise ausgesparte Verben distribuiert werden - ist Koordinationsreduktion bei koordinierten Sätzen in ACE nicht zulässig (z.B. **John works in the library and Rona at the information desk*).

5.7 Fragesätze in ACE

Nicht nur subordinierte ACE Sätze können als eine strukturelle Variante von einfachen ACE Sätzen aufgefasst werden, sondern auch Fragesätze. Im Fall von Entscheidungsfragen werden einzelne Wörter im Vergleich zu einfachen ACE Sätzen lokal umgestellt, d.h. in einer satzexternen Position realisiert oder erst dort eingesetzt. Ergänzungsfragen werden in struktureller Analogie zu restriktiven Relativsätzen gebildet, und zwar steht ein Frageausdruck, dem in bestimmten Fällen ein Operator folgt, in einer satzexternen Position und lizenziert eine nicht lokal gebundene Lücke im verbleibenden Satz.

5.7.1 Entscheidungsfragen

Entscheidungsfragen werden durch "Subjekt-Hilfsverb-Inversion" gebildet oder - dort wo keine Form von *be* verwendet wird - durch *do*-Einsetzung. Genaugenommen ist das Bild das reine Inversion suggeriert falsch, denn bei Entscheidungsfragen handelt es sich wiederum um eine Satzkategorie vom Typ `s1`, die eine satzexterne Position enthält. Das Hilfsverb *be* bzw. der *do*-Operator steht in der `comp`-Position und der verbleibende Satz (`s`) funktioniert als Satzkomplement. Bei Entscheidungsfragen, die mit dem Hilfsverb *be* gebildet werden, entsteht im Satzkomplement eine lokale Lücke an der Stelle, wo in einfachen ACE Sätzen das Verb *be* stehen würde. Bei Entscheidungsfragen, die mit dem *do*-Operator gebildet werden, bleibt eine Grundform des Vollverbs im Satzkomplement stehen.

Realisierung von Entscheidungsfragen

specC	comp	Satzkomplement
	[op does]	[s the staff user enter the password]
	[cop is]	[s the password [t] valid]

5.7.2 Ergänzungsfragen

Ergänzungsfragen werden durch Frageausdrücke gebildet, die mit einer syntaktischen Funktion in einfachen ACE Sätzen in Beziehung stehen. Bei den Frageausdrücken handelt es sich entweder um Fragepronomen (`qpro`), Fragedeterminatoren (`qdet`) oder Frageadverbien (`qadv`). Fragepronomen (*who, what*) beziehen sich auf Information, die in einfachen ACE Sätzen durch interne oder externe Komplemente ausgesagt worden ist. Fragedeterminatoren (*which, whose*) erfragen Information, die in einer nominalen Konstituente von ACE Sätzen durch Attribute, Appositionen, Relativsätze oder *of*-

Konstruktionen mitgeteilt worden ist. Frageadverbien (z.B. *where, when, how*) erheben Information, die in einfachen ACE Sätzen durch verbale Adjunkte ausgesagt worden ist. Gemeinsam ist diesen Frageausdrücken im Unterschied zu Entscheidungsfragen, dass sie nicht einen ganzen Sachverhalt erfragen, der zuvor in einem ACE Satz ausgesagt worden ist, sondern sich immer nur auf Teilaspekte dieses Sachverhaltes beziehen. Im Fall von Ergänzungsfragen stehen die Frageausdrücke syntaktisch immer in der `specC`-Position und weisen einen *do*-Operator in der `comp`-Position auf, ausser wenn sich der Frageausdruck auf die Subjektposition im Satzkomplement bezieht. Die Frageausdrücke sind mit einer phrasalen Lücke im Satzkomplement assoziiert, die nicht lokal gebunden ist.

Realisierung von Ergänzungsfragen

specC	comp	Satzkomplement
[n2 [qpro who]]		[s [t] enters John's password into the computer]
[n2 [qpro what]]	[op does]	[s the staff user enter [t] into the computer]
[n2 [qdet whose] [n1 password]]	[op does]	[s the staff user enter [t] into the computer]
[n2 [qadv where]]	[op does]	[s the staff user enter John's password into [t]]

5.8 Satzbauplan und Schreibprinzipien für ACE

ACE ist eine kontrollierte natürliche Sprache, die aus einer wohldefinierten Teilmenge der englischen Standardsprache besteht. Um diese Teilmenge für den Anwendungsspezialisten leicht erlernbar zu machen und die kontrollierte Verwendungsweise der zulässigen Konstruktionen im Schreibprozess optimal zu unterstützen, stehen neben dem Satzbauplan eine überschaubare Anzahl von 30 Schreibprinzipien zur Verfügung. Die Schreibprinzipien orientieren die Anwendungsspezialisten darüber, wie das Vokabular organisiert ist, welche lexikalischen und strukturellen Einschränkungen beim Ausbau des Satzbauplans gelten und wie die zulässigen Konstruktionen interpretiert werden. Die Schreibprinzipien machen zu diesem Zweck (hauptsächlich) normative Konstruktionsaussagen, die die Auswahl und die Organisation der sprachlich zulässigen Mittel steuern und unter Kontrolle bringen. Um dem praktischen Gebrauch zu genügen, sind die normativen Konstruktionsaussagen durch explikative Erläuterungen ergänzt. Beispielsweise teilen die Schreibprinzipien durch graphische Mittel in einer Paraphrase mit, wenn aus einer Konstruktionsaussage aufgrund der deterministischen Parsingstrategie des Attempto Systems eine syntaktische Substitution oder eine bestimmte strukturelle Interpretation folgt. Ausserdem zeigen Reformulierungsvorschläge an, wie alternative Interpretationen erschlossen werden können. Die Schreibprinzipien basieren auf dem Satzbauplan und setzen eine Reihe von Grundkenntnissen über die Ausbaumöglichkeiten des Satzbauplans voraus.

5.8.1 Ausbau des einfachen Satzbauplans

Die nachfolgende Aufstellung illustriert am Beispiel von konkreten ACE Sätzen, wie die einzelnen syntaktischen Funktionen des Satzbauplans formal ausgebaut sein können und macht die Anwendungsspezialisten mit den Grundformen der Sprache ACE vertraut. Die Schreibprinzipien formulieren dann zusätzliche Einschränkungen für den vertikalen und den horizontalen Ausbau des Satzbauplans.

Subjekt + [{ Negation } + verbaler Kopf + Komplement { + Adjunkt }]

Das Subjekt als externes Komplement wird immer durch eine Nominalphrase realisiert. Interne Negation kommt durch einen der beiden Negationsausdrücke *does not* und *not* zustande. Der verbale Kopf bildet das strukturelle Zentrum des Satzbauplans und wird entweder durch ein intransitives, transitives oder ditransitives Verb realisiert oder durch das kopulative Verb *be*. Bei den internen Komplementen handelt es sich entweder um Nominalphrasen, Präpositionalphrasen oder Adjektivphrasen, die durch das Verb selektioniert werden. Das Adjunkt liefert optionale Information in

Form einer Adverbialphrase oder einer Präpositionalphrase.

Subjekt: Nominalphrase

A staff user creates a catalogue entry.
The staff user creates a catalogue entry.
Every staff user creates a catalogue entry.
No staff user creates a catalogue entry.
The staff user 'no. 3' creates a catalogue entry.
The responsible staff user creates a catalogue entry.
The staff user who works at the check-out counter creates a catalogue entry.
John creates a catalogue entry.
He creates a catalogue entry.
John's secretary creates a catalogue entry.
The secretary of John creates a catalogue entry.

Negation: Negationswort

The staff user does not create a catalogue entry.
The copy is not in the library.

Verbaler Kopf: Verb

The staff user works.
The staff user returns the copy.
The borrower brings the copy back.
The staff user gives the copy to the borrower.
The copy is in the library.

Komplement: Nominalphrase/Präpositionalphrase/Adjektivphrase

The staff user returns a copy to the library.
The staff user returns Mary's copy to the library.
The staff user returns the old copy to the library.
The staff user returns the copy 'no. 185' to the library.
The staff user returns the copy of a book to the library.
The staff user returns the copy that has a damaged cover to the library.
The borrower asks for a registration form.
The identity card is valid.
The book amount is smaller than the book limit.

Adjunkt: Adverbialphrase/Präpositionalphrase

The staff user enters the identity card slowly.
The staff user enters the identity card into the bar code reader.
The staff user enters the identity card slowly into the bar code reader.

5.8.2 Ausbau des erweiterten Satzbauplans

Die nachfolgende Aufstellung zeigt, wie die beiden satzexternen Positionen *W* und *C* des erweiterten Satzbauplans besetzt sein können, so dass sich auf der Grundlage von einfachen ACE Sätzen zusammengesetzte Sätze und Fragesätze in ACE bilden lassen. Dabei ist zu beachten, dass es sich beim Satzkomplement auch um einen unvollständigen ACE Satz handeln kann.

W + C + Satzkomplement

Die *W*-Position des erweiterten Satzbauplans wird ausschliesslich durch *wh*-Ausdrücke realisiert, die eine phrasale Lücke im Satzkomplement legitimieren. Wenn eine Ergänzungsfrage nicht nach dem Subjekt fragt, dann steht neben dem *wh*-Ausdruck in der *W*-Position zusätzlich ein Complementizer (*do*-Operator) in der *C*-Position. Alle übrigen Complementizer und Koordinatoren stehen in der *C*-Position und leiten ein Satzkomplement ein oder verbinden einen Teilsatz.

W-Position: *wh*-Ausdruck
… which the staff user enters.
Who enters John's identity card into the bar code reader?

W-C-Position: *wh*-Ausdruck und Complementizer
What does the staff user enter into the bar code reader?
Whose identity card does the staff user enter into the bar code reader?
Where does the staff user enter John's identity card into?

C-Position: Complementizer oder Koordinator
Does the staff user enter the password?
Is the password valid?
If a copy of a book is checked out to a borrower …
… and the copy is available …
… or the copy is available …
either the copy is available …

Zu den Complementizern zählen in ACE der Operator *do*, das kopulative Verb *be* und die konditionale Konjunktion *if*. Zulässige Koordinatoren in ACE sind die Konjunktion *and*, die inklusive Disjunktion *or* und die exklusive Disjunktion *either … or*.

5.8.3 Schreibprinzipien

Die Schreibprinzipien sind für die Wortebene, Konstituentenebene, Satzebene und Textebene definiert. Ein Schreibprinzip besteht aus einer normativen und einer explikativen Komponente. Die normative Komponente macht eine Konstruktionsaussage und legt fest, ob eine lexikalische Kategorie oder eine syntaktische Konstruktion zum Sprachumfang von ACE gehört und wie die sprachlichen Elemente interpretiert werden. Die Konstruktionsaussagen sind bewusst knapp formuliert, damit sie gut lernbar und erinnerbar sind. Die explikative Komponente verdeutlicht, die in der normativen Komponente gemachte Aussage durch Beispielsätze. Hier ist ein solches Schreibprinzip:

[10] Relativsätze modifizieren die unmittelbar vorausgehende nominale Konstituente.

(E) *The staff user returns a copy of a book that has a damaged cover.*
(P) *The staff user returns a copy of { a book that has a damaged cover }.*
(R) *The staff user returns a copy of a book. The copy has a damaged cover. [→ 29].*

Die Konstruktionsaussage in der normativen Komponente trägt eine Nummer und ist fett gedruckt. Die Beispielsätze in der explikativen Komponente gehören drei unterschiedlichen Gruppen an: *(E)* bezeichnet einen korrekten (oder inkorrekten) ACE Satz als Eingabesatz. *(P)* bezeichnet eine Paraphrase eines korrekten ACE Satzes und *(R)* einen Reformulierungsvorschlag für einen ACE Satz. Korrekte ACE Sätze illustrieren die Umsetzung einer normativen Konstruktionsaussage. Inkorrekte ACE Sätze zeigen, die Verletzung einer Konstruktionsaussage an und sind durch einen Stern (*) gekennzeichnet. Die zur Diskussion stehende Konstruktion ist bei diesen Eingabesätzen jeweils durch Unterstreichung graphisch markiert. Wo nötig informiert eine Paraphrase (*P*), wie ein korrekter ACE Satz aufgrund der deterministischen Parsingstrategie interpretiert wird. Geschweifte Klammern (*{}*) in der Paraphrase zeigen die gewählte Analyse bei struktureller Mehrdeutigkeit an. Die Anbindung von Relativsätzen an eine nominale Konstituente oder die Anbindung von Adverbial- und Präpositionalphrasen an eine verbale Konstituente wird durch dieses graphische Mittel klar gemacht. Mögliche eckige Klammern (*[]*) in der Paraphrase zeigen eine textuelle Substitution an und verdeutlichen eine bestimmte Interpretation. Ersetzung von Synonymen durch das Leitwort, strukturelle Rekonstruktion bei Koordinationsreduktion oder Auflösung von anaphorischer Referenz wird durch dieses Mittel angezeigt. Reformulierungsvorschläge geben einen Hinweis darauf, wie eine alternative Lesart in ACE aussehen kann und zeigen auf das betreffende Schreibprinzip (*[→ 29]*), das diesen Vorschlag erklärt.

Wortebene

[1] Inhaltswörter müssen definiert werden und sind veränderbar.

(E) *The staff user enters the new identity card slowly into the bar code reader.*

[2] Funktionswörter sind strukturbildend und vordefiniert.

(E) *If no catalogue entry of the book exists then he creates a catalogue entry that contains the author name of the book and the title of the book.*

[3] Inhaltswörter mit modalartigem oder intensionalem Charakter sind nicht zulässig.

(E) **The copy may be available.*

(E) **The staff user knows the name of the borrower.*

(E) **The result is possible.*

(E) **The staff user probably adds a copy of a book to the library.*

[4] Synonyme und Abkürzungen für Inhaltswörter sind zulässig und werden durch das Leitwort ersetzt.

(E) *The employee enters the id of a copy.*
(P) *The [staff user] enters the [identification] of a copy.*

[5] Verben stehen in der 3. Person Singular, im Indikativ, im Aktiv und im *simple present tense*.

(E) *The copy is available.*

(E) *The staff user returns a copy of a book to the library.*

(E) *Is the copy in the library?*

(E) *What does the staff user return to the library?*

[6] Verben bezeichnen entweder Ereignisse oder Zustände.

(E) *The staff user adds a copy of a book to the library.*

(E) *The book has an ISBN number.*

[7] Partizip Perfekt Formen von Verben sind als Adjektive zulässig.

(E) *The copy that has a damaged cover is checked out to a borrower.*

Konstituentenebene

[8] Gattungsbezeichnungen sind immer mit einem Determinator zu verwenden.

(E) *The staff user creates a catalogue entry.*

(E) *There is no staff user that owns a password.*

[9] *of*-Konstruktionen sind die einzigen zulässigen postnominalen Präpositionalphrasen.

(E) *The staff user checks out a copy of a book.*

(E) **The staff user checks out a copy with a damaged cover.*

[10] Relativsätze modifizieren die unmittelbar vorausgehende nominale Konstituente.

(E) *The staff user returns a copy of a book that has a damaged cover.*
(P) *The staff user returns a copy of { a book that has a damaged cover }.*
(R) *The staff user returns a copy of a book. The copy has a damaged cover. [→ 29].*

[11] Adverbial- und Präpositionalphrasen modifizieren die verbale Konstituente.

(E) *The staff user enters the identity card slowly into the bar code reader.*
(P) *The staff user { enters the identity card slowly into the bar code reader }.*
(R) *The staff user slowly enters the identity card into the bar code reader. [→ 12].*

(E) *The staff user returns a copy of a book with a damaged cover.*
(P) *The staff user { returns a copy of a book with a damaged cover }.*
(R) *The staff user returns a copy of a book that has a damaged cover. [→ 10].*

[12] Adverbien stehen unmittelbar vor dem Verb oder folgen einer verbalen Konstituente.

(E) *The staff user slowly enters the identity card.*

(E) *The staff user enters the identity card slowly.*

[13] Reihenfolge der Verbalphrasen bestimmt die zeitliche Anordnung der Ereignisse.

(E) *The borrower presses the key 'A' and presses the key 'B'.*
(R) *The borrower presses the key 'A' and presses the key 'B' at the same time. [→ 14].*

[14] Vordefinierte Diskurssignale heben die Reihenfolge von koordinierten Verbalphrasen auf.

(E) *The borrower presses the key 'A' and presses the key 'B' at the same time.*

(E) *The borrower presses the key 'A' and the key 'B' in any temporal order.*

[15] Koordination ist nur zwischen gleichartigen Phrasen (und Sätzen) zulässig.

(E) *The borrower presses the key 'A' and presses the key 'B'.*

(E) *The password is correct and valid.*

(E) *The staff user enters the password correctly and manually.*

(E) *John works at the information desk and at the check-out counter.*

(E) *John works in the library and Rona works at the information desk.*

[16] Koordinationsreduktion in Verbalphrasen ist zulässig und führt zu einer Rekonstruktion.

(E) *The borrower presses the key 'A' and the key 'B'.*
(P) *The borrower presses the key 'A' and [presses] the key 'B'.*

(E) *LibDB does not list the book amount and the book limit.*
(P) *LibDB does not list the book amount and [does not list] the book limit.*

(E) *The password is not correct and valid.*
(P) *The password is not correct and [is not] valid.*

[17] Koordination zwischen Subjekt-Nominalphasen und *of*-Konstruktionen ist nicht zulässig.

(E) **John and Mary enter a password.*
(R) *John enters a password and Mary enters a password. [→ 15].*

(E) **The staff user checks out the book of Hofstadter and of Kleist.*
(R) *The staff user checks out a book of Hofstadter and checks out a book of Kleist. [→ 15].*

[18] Koordination unterliegt der Vorrangsregel: *Konjunktion bindet stärker als Disjunktion.*

(E) *The borrower presses the key 'A' or the key 'B' and the key 'C'.*
(R) *The borrower presses the key 'A' or the key 'B', and the key 'C'. [→ 19].*
(R) *The borrower presses the key 'A' or the key 'B'. The borrower presses the key 'C'. [→ 13].*

[19] Komma ist ein Diskurssignal und invertiert die Vorrangsregel bei Koordination.

(E) *The borrower presses the key 'A' or the key 'B', and the key 'C'.*

[20] Bei inklusiver Disjunktion schreibt man *or* und bei exklusiver Disjunktion *either … or*.

(E) *The borrower presses the key 'A' or the key 'B'.*

(E) *The borrower either presses the key 'A' or the key 'B'.*

Satzebene

[21] (Eigenname | definite Nominalphrase) + kopulatives Verb + (Eigenname | definite Nominalphrase) bezeichnen eine Identität.

(E) *Mary is the staff user.*
(P) *Mary is [identical to] the staff user.*
(R) *Mary is a staff user. [→ 22].*

[22] Nominalphrase + kopulatives Verb + indefinite Nominalphrase bezeichnen einen Zustand.

(E) *Mary is a staff user.*
(R) *Mary is the staff user. [→ 21].*

[23] Skopus einer Nominalphrase reicht immer bis zum Statzende.

(E) *Every staff user who works at the information desk gets no extra holiday.*
(R) *There is no extra holiday that every staff user who works at the information desk gets. [→ 24].*

[24] Skopus einer Nominalphrase ist durch Topikalisierung veränderbar.

(E) *There is no extra holiday that every staff user who works at the information desk gets.*

(E) *For every book there is a number that the staff user assigns to the book.*

(E) *There is a staff user who every machine identifies.*

[25] Skopus bei externer Negation (= Negation einer Nominalphrase) reicht bis zum Satzende.

(E) *No staff user owns a password.*

(E) *The staff user enters no password.*
(R) *There is no password that the staff user enters. [→ 24].*

[26] Skopus bei interner Negation (= Negation der Verbalphrase) reicht über die Verbalphrase.

(E) *John does not own a password that is valid.*

(E) *John does not own a password and an identification card.*
(P) *John does not own a password and [does not own] an identification card.*

[27] Skopus eines Konditionalsatzes reicht immer bis zum Statzende.

(E) *If the staff user returns a copy then it is available.*

Textebene

[28] Eine Spezifikation ist ein ACE Text, der aus einzelnen Paragraphen besteht.

(E) *A staff user enters the id of a copy.*
If the copy is checked out to a borrower
then LibDB displays the name of the borrower.
If the copy is not checked out to a borrower
then LibDB displays the message 'No borrower'.

[29] Personalpronomen werden und Eigennamen oder definite Nominalphrasen können anaphorisch verwendet werden. Die Auflösung erfolgt aufgrund von Gender, Numerus und Distanz.

(E) *The password consists of an alphanumeric number. It ...*
(P) *The password consists of an alphanumeric number. [The alphanumeric number] ...*

(E) *John adds a copy of a book to the library. John ...*
(P) *John adds a copy of a book to the library. [John] ...*

(E) *The staff user returns a copy of a book. The copy has a damaged cover.*
(P) *The staff user returns a copy of a book. [The copy] has a damaged cover.*

(E) *The staff user hands over a green identity card to a borrower. The identity card ...*
(P) *The staff user hands over a green identity card to a borrower. [The green identity card] ...*

[30] Anaphorische Referenz ist nicht möglich, wenn eine indefinite oder definite Nominalphrase im Skopus eines Negationsausdrucks, einer universell quantifizierten Nominalphrase oder einer Implikation steht.

(E) *John does not own a password. *It is valid.*
(R) *John does not own a password that is valid. [→ 26].*

(E) *Every staff user owns a password. *He enters it.*
(P) *Every staff user owns a password that he enters. [→ 23].*

(E) *If no catalogue entry of the book exists then the staff user creates a catalogue entry for the book. *A copy of the book is available.*
(R) *If no catalogue entry of the book exists then the staff user creates a catalogue entry for the book and a copy of the book is available. [→ 27].*

6. Eine Evaluation von ACE

Das Bibliotheksproblem liegt in einer sehr knappen natürlich-sprachlichen Problembeschreibung vor, die über Sachverhalte aus einem allgemein vertrauten Anwendungsbereich einer Bibliothek informiert [vgl. Seite 30]. Möglicherweise hält der Leser dieser Arbeit ein Buch in den Händen, das er sich aus einer Bibliothek ausgeliehen hat. Beim Bibliotheksproblem kann man auf die unmittelbare Erfahrung der Leser im Verkehr mit Bibliotheken zählen, wenn die Problembeschreibung Aussagen über Transaktionen wie *Check out a copy of a book* oder *Return a copy of a book* macht. Doch gerade diese scheinbare Gewissheit über die Art der vorliegenden Sachverhalte legt Fallstricke aus, wenn es darum geht, die informelle Problembeschreibung in eine exaktere Software-Anforderungsspezifikation (SAS) zu überführen. Ein implizites Fürwahrhalten der konzeptionellen Voraussetzungen in der Problembeschreibung kann keine intersubjektiv gesicherte Geltung beanspruchen. Die konzeptionellen Voraussetzungen, denen die sprachlichen Ausdrücke im Anwendungsbereich zugrundeliegen, müssen zuerst ausgehandelt und möglichst vollständig geklärt werden.

6.1 Konzeptionelle Voraussetzungen

Die Problembeschreibung ist an verschiedenen Stellen mehrdeutig und unvollständig. Bevor ein Vorschlag einer SAS in ACE für das Bibliotheksproblem vorgestellt wird, sollen diese kritischen Stellen erörtert werden. Es wird diskutiert, welche Mehrdeutigkeiten auftauchen und welche konzeptionellen Entscheidungen bei der Umsetzung der SAS in ACE getroffen worden sind. Da es sich bei ACE um eine sachverhaltsorientierte Spezifikationssprache handelt, die durch einen Computer ausführbar ist, können diese konzeptionellen Entscheidungen entweder unmittelbar auf der textuellen Ebene nachvollzogen oder bei der Ausführung der Spezifikation in ACE beobachtet werden.

In ihrer komparativen Studie von 12 unterschiedlichen Spezifikationsvorschlägen für das Bibliotheksproblem stellt [Wing 88] fünf semantische Mehrdeutigkeiten in der Problembeschreibung fest. Sie formuliert für diese Mehrdeutigkeiten Fragen und untersucht, ob und wie die verschiedenen Beiträge dazu Stellung nehmen. In der Folge sollen diese Fragen auch für die zu entwickelnde SAS in ACE beantwortet werden. Das trägt zur Klärung der Anforderungen bei und simuliert zugleich eine Anforderungsanalyse. Die Anworten auf die Fragen bilden dann den Ausgangspunkt für den Spezifikationsvorschlag in ACE und helfen die konsistente Verwendungsweise der lexikalischen Ausdrücke für die SAS zu klären. Aus den Antworten können informelle Erkennungsregeln abgeleitet werden, die die benutzerdefinierten Inhaltswörter im Lexikon des Attempto Systems ergänzen und die am Software-Entwicklungsprozess Beteiligten anleiten, wie die sprachlichen Ausdücke in der Anwendungsumgebung verwendet werden.

1. Was ist eine Bibliothek?

Die Bibliothek (*library*) ist die Anwendungsumgebung, in der die Anforderungen an die zu entwickelnde Bibliotheksdatenbank (*LibDB*) erhoben werden und in der die Bibliotheksdatenbank ihre Dienstleistungen anbieten wird. Die Anwendungsumgebung besteht aus zwei Benutzergruppen (*user*), die mit unterschiedlicher Berechtigung Transaktionen auf der Bibliotheksdatenbank ausführen.

2. Was ist ein Benutzer?

In der Anwendungsumgebung werden zwei Gruppen von Benutzern (*user*) unterschieden: Mitarbeiter (*staff user*) und Ausleiher (*borrower*). Ob ein Mitarbeiter zugleich ein Ausleiher sein kann, darüber gibt die Problembeschreibung keine Auskunft. Deshalb wird dieser Sachverhalt auch nicht in der SAS berücksichtigt. Für die Mitarbeiter sind alle fünf Transaktionen aus der Problembeschreibung zulässig. Die Ausleiher hingegen können nur die 3. Transaktion (*Get the list of books by a particular author or in a particular subject area*) vollständig ausführen. Die 4. Transaktion (*Find out the list of books currently checked out by a particular borrower*) kann von einem Ausleiher ausgeführt werden, wenn sie auf persönliche Information beschränkt bleibt. Die Ausleiher können gemäss der dritten Restriktion (*A borrower may not have more than a predefined number of books checked out at one time*) nur dann ein Buch beziehen, wenn die Anzahl der bereits bezogenen Bücher (*book amount*) kleiner ist als eine festgelegte Limite (*book limit*).

3. Was ist ein Buch?

Ein Buch ist eine konzeptionelle Entität, mit der ein Autorname (*author name*), ein Buchtitel (*title*) und ein Sachgebiet (*subject area*) assoziiert wird. In ACE werden diese Teilbeziehungen durch *of*-Konstruktionen hergestellt: *the author name of the book*, *the title of the book* oder *the subject area of the book*. Für jedes neue Buch, das in die Bibliothek aufgenommen wird, erzeugt ein Mitarbeiter einen Katalogeintrag (*catalogue entry*). Im Unterschied zu einem Buch ist eine Kopie (*copy*) eine physikalische Entität, die durch eine eindeutige Identifikation (*an id of the copy*) gekennzeichnet ist. Über die Art der Identifikation wird in der ACE Spezifikation nichts ausgesagt. Kopien mit dem gleichen Autornamen und dem gleichen Titel gehören zu derselben konzeptionellen Enti-

tät *book*. Der Bezug zwischen einer Kopie und einem Buch wird in ACE wiederum durch eine *of*-Konstruktion hergestellt, z.B. *a copy of the book is available*.

4. Was bedeutet "available"?
Eine Buchkopie ist entweder in der Bibliothek verfügbar oder ausgeliehen. Beides kann nicht zur gleichen Zeit der Fall sein. Die ersten beiden Restriktionen in der Problembeschreibung (*All copies in the library must be available for checkout or be checked out* und *No copy of the book may be both available and checked out at the same time*) legen fest, dass alle Buchkopien entweder verfügbar oder ausgeliehen sind und dass keine Buchkopie gleichzeitig verfügbar und ausgeliehen ist. In der ACE Spezifikation werden diese beiden Restriktionen zusammen berücksichtigt. Neben den beiden angesprochenen Zuständen *available* und *checked out* sind in einer realen Bibliothek normalerweise weitere Zustände für eine Buchkopie denkbar. Eine Buchkopie kann entweder als gestohlen oder verloren gelten oder aber in Reparatur sein. Über diese Möglichkeiten spricht die Problembeschreibung nicht. Aus diesem Grund werden solche Fälle auch nicht in der ACE Spezifikation behandelt.

5. Was bedeutet "last checked out"?
Schwierig ist die Frage zu beantworten, wie der Ausdruck *last checked out* der 5. Transaktion (*Find out what borrower last checked out a particular copy of a book*) in der Problembeschreibung zu interpretieren ist und ob dieser Ausdruck im Kontrast zum Ausdruck *currently checked out* in der 4. Transaktion (*Find out the list of books currently checked out by a particular borrower*) steht. Wenn *currently checked out* und *last checked out* als Synonyme aufgefasst werden, dann ist der Name des gegenwärtigen Ausleihers der Buchkopie ein korrektes Resultat für die 5. Transaktion. Wenn *currently checked out* und *last checked out* einen Unterschied implizieren, dann sind zwei intendierte Interpretationen denkbar. Bei der ersten Interpretation sind zwei Fälle zu unterscheiden: Wenn die Buchkopie gerade ausgeliehen ist, dann ist der Name des gegenwärtigen Ausleihers ein korrektes Resultat für die 5. Transaktion. Wenn die Buchkopie nicht ausgeliehen ist, dann ist der Name desjenigen Ausleihers ein korrektes Resultat, der die Buchkopie zuletzt ausgeliehen hat. Bei der zweiten Interpretation ist nur der Name desjenigen Ausleihers ein korrektes Resultat, der die Buchkopie zuletzt ausgeliehen hat. Es zeigt sich, dass die Frage, was *last checked out* bedeutet, nicht abschliessend beantwortet werden kann, weil nicht klar ist, was die Autoren bei der Problembeschreibung mit dem Ausdruck intendiert haben. Für den Spezifikationsvorschlag in ACE werden die beiden Ausdrücke in der Problembeschreibung als Synonyme interpretiert. Die zutreffende Interpretation kann erst bei der Ausführung der Spezifikation

durch die Systembenutzer validiert werden und ist kaum durch blosses Lesen der ACE Spezifikation zu entdecken.

Die natürlich-sprachliche Problembeschreibung ist (notwendigerweise) unvollständig. Sie macht keine Aussagen darüber, welche Eigenschaften zu Beginn als Initialzustand für die Bibliotheksdatenbank gelten. Ausserdem fehlt zumindest die Transaktion *Add a new copy of a book to the library*, wenn man sich bei der Konzeptualisierung für eine Unterscheidung zwischen einem Buch und einer Buchkopie entschlossen hat. Auch macht die Problembeschreibung keinerlei Aussagen darüber, ob bei einem Fehlschlag einer Transaktion eine Fehlerbehandlung durchgeführt werden soll und ob bestimmte nichtfunktionale Anforderungen zu berücksichtigen sind.

Die Problembeschreibung basiert auf einer funktionsorientierten Systemsichtweise, bei der die einzelnen Transaktionen im Mittelpunkt stehen. Die Transaktionen können (weitgehend) unabhängig voneinander ausgeführt werden. Eine Ausführung in textueller Reihenfolge ist zwar möglich, macht aber wenig Sinn, da dies einer unrealistischen Vorgehensweise entsprechen würde, die so im Anwendungsbereich nicht vorzufinden ist. Die einzelnen Transaktionen können als Minispezifikationen betrachtet werden und sind in ACE adäquat als unabhängige Paragraphen darstellbar. Die Transaktionen hängen von einer Reihe von Vorbedingungen ab, die festlegen, wann die Transaktion zulässig ist. Die Konsequenzen beschreiben dann, was sich als Resultat der Transaktion verändert. Da ACE Spezifikationen zur Zeit über keinen ausgearbeiteten Mechanismus verfügen, um Restriktionen in einer allgemeinen Weise, d.h. nicht-lokal, zu definieren, sind die Restriktionen in die einzelnen Minispezifikationen eingearbeitet.

Um eine solche Minispezifikation in ACE auszuführen, sind zwei Vorgehensweisen vorstellbar: Der Anwendungsspezialist kann entweder während der Ausführung vom Attempto System als Orakel befragt werden und die einzelnen Vorbedingungen einer Transaktion sanktionieren oder er muss vor der Ausführung einen Testfall in ACE schreiben, aus dem die notwendigen Informationen für die Vorbedingungen und der automatische Aufruf der Transaktion abgeleitet wird. Wenn der Testfall unvollständig ist und nicht alle Vorbedingungen der Transaktion erfüllbar sind, dann kann die fehlende Information vom Anwendungsspezialisten erfragt werden.

6.2 Eine ACE Spezifikation für das Bibliotheksproblem

% ==
% [1 a] Check out a copy of a book.
% ==

If
    a borrower asks for a copy of a book
    and the copy is available
    and LibDB calculates the book amount of the borrower
    and the book amount is smaller than the book limit
    and a staff user checks out the copy to the borrower
then
    the copy is checked out to the borrower.

% ——————————————————————————————————————
% [1 a] Diskursrepräsentationsstruktur
% ——————————————————————————————————————

```
[F]
named(F,'LibDB')

IF
   [A,B,C,D,E,G,H,I,J,K,L]
   borrower(A)
   copy(B)
   book(C)
   of(B,C)
   event(D,ask_for(A,B))
   state(E,available(B))
   book_amount(G)
   of(G,A)
   event(H,calculate(F,G))
   book_limit(I)
   state(J,smaller_than(G,I))
   staff_user(K)
   event(L,check_out_to(K,B,A))
   THEN
      [M]
      state(M,checked_out_to(B,A))
```


% ==
% [1 b] Return a copy of a book.
% ==

If
    a copy of a book is checked out to a borrower
    and a staff user returns the copy
then
    the copy is available.

% ——————————————————————————————————————
% [1 b] Diskursrepräsentationsstruktur
% ——————————————————————————————————————

```
IF
   [A,B,C,D,E,F]
   copy(A)
   book(B)
   of(A,B)
   borrower(C)
   state(D,checked_out_to(A,C))
   staff_user(E)
   event(F,return(E,A))
   THEN
      [G]
      state(G,available(A))
```

% ==
% [2 a] Add a copy of a new book to the library.
% ==

If
    a staff user adds a copy of a book to the library
    and no catalogue entry of the book exists
then
    the staff user creates a catalogue entry
        that contains the author name of the book
            and the title of the book and the subject area of the book
    and he enters the id of the copy
    and the copy is available.

```
% ------------------------------------------------------------
% [2 a]  Diskursrepräsentationsstruktur
% ------------------------------------------------------------

IF
   [A,B,C,D,E]
   staff_user(A)
   copy(B)
   book(C)
   of(B,C)
   library(D)
   event(E,add_to(A,B,D))
      NOT
         [F,G]
         catalogue_entry(F)
         of(F,C)
         state(G,exist(F))
   THEN
      [H,I,J,K,L,M,N,O,P,Q,R]
      catalogue_entry(H)
      event(I,create(A,H))
      author_name(J)
      of(J,C)
      state(K,contain(H,J))
      title(L)
      of(L,C)
      state(M,contain(H,L))
      subject_area(N)
      of(N,C)
      state(O,contain(H,N))
      identification(P)
      of(P,B)
      event(Q,enter(A,P))
      state(R,available(B))
```

```
% ============================================================
% [2 b]  Add a copy of a book to the library.
% ============================================================
```

If
    a staff user adds a copy of a book to the library
    and a catalogue entry of the book exists
then
    the staff user enters the id of the copy and the copy is available.


```
% ------------------------------------------------------------
% [2 b]  Diskursrepräsentationsstruktur
% ------------------------------------------------------------

IF
   [A,B,C,D,E,F,G]
   staff_user(A)
   copy(B)
   book(C)
   of(B,C)
   library(D)
   event(E,add_to(A,B,D))
   catalogue_entry(F)
   of(F,C)
   state(G,exist(F))
   THEN
      [H,I,J]
      identification(H)
      of(H,B)
      event(I,enter(A,H))
      state(J,available(B))
```

```
% ============================================================
% [2 c]  Remove a copy of a book from the library.
% ============================================================
```

If
    the copy is available
    and the staff user removes the copy from the library
then
    LibDB deletes the id of the copy
    and the copy is not available.

```
% ------------------------------------------------------------
% [2 c]  Diskursrepräsentationsstruktur
% ------------------------------------------------------------

[F]
named(F,'LibDB')

IF
   [A,B,C,D,E]
   copy(A)
   state(B,available(A))
```

```
   staff_user(C)
   event(D,remove(C,A))
   origin(D,from(E))
   library(E)
   THEN
      [G,H]
      identification(G)
      of(G,A)
      event(H,delete(F,G))
      NOT
         [I]
         state(I,available(A))
```

% ==
% [3] Get the list of books by a particular author or in a particular subject area.
% ==

If
    a user enters an author name
    and the user is a staff user or a borrower
then
    for every catalogue entry that contains the author name
        LibDB lists the author name and the title.

% ——
% [3 a] Diskursrepräsentationsstruktur
% ——

```
[H]
named(H,'LibDB')

IF
   [A,B,C]
   user(A)
   author_name(B)
   event(C,enter(A,B))
      OR
         [D]
         state(D,staff_user(A))
      OR
         [E]
         state(E,borrower(A))
   THEN
      IF
```



```
         [F,G]
         catalogue_entry(F)
         state(G,contain(F,B))
         THEN
            [I,J,K]
            event(I,list(H,B))
            title(J)
            event(K,list(H,J))
```

% ==

If
    a user enters a subject area
    and the user is a staff user or a borrower
then
    for every catalogue entry that contains the subject area
        LibDB lists the author name and the title.

% ——
% [3 b] Diskursrepräsentationsstruktur
% ——

```
[H]
named(H,'LibDB')

IF
   [A,B,C]
   user(A)
   subject_area(B)
   event(C,enter(A,B))
      OR
         [D]
         state(D,staff_user(A))
      OR
         [E]
         state(E,borrower(A))
   THEN
      IF
         [F,G]
         catalogue_entry(F)
         state(G,contain(F,B))
         THEN
            [I,J,K,L]
            author_name(I)
```

```
          event(J,list(H,I))
          title(K)
          event(L,list(H,K))


% ======================================================
% [4]     Find out the list of books currently checked out by a particular borrower.
% ======================================================
```

If
    a user enters a name of a borrower
    and the user is a staff user
then
    for every copy that is checked out to the borrower
        LibDB lists the author name and the title.

```
% ------------------------------------------------------
%  [4 a]  Diskursrepräsentationsstruktur
% ------------------------------------------------------

[H]
named(H,'LibDB')

IF
   [A,B,C,D,E]
   user(A)
   name(B)
   borrower(C)
   of(B,C)
   event(D,enter(A,B))
   state(E,staff_user(A))
   THEN
     IF
       [F,G]
       copy(F)
       state(G,checked_out_to(F,C))
       THEN
         [I,J,K,L]
         author_name(I)
         event(J,list(H,I))
         title(K)
         event(L,list(H,K))
```


If
    a user enters a name of a borrower
    and the user is the borrower
then
    for every copy that is checked out to the borrower
        LibDB lists the author name and the title.

```
% ------------------------------------------------------
% [4 b]   Diskursrepräsentationsstruktur
% ------------------------------------------------------

[F]
named(F,'LibDB')

IF
   [A,B,C]
   user(A)
   name(B)
   borrower(A)
   of(B,A)
   event(C,enter(A,B))
   THEN
     IF
       [D,E]
       copy(D)
       state(E,checked_out_to(D,A))
       THEN
         [G,H,I,J]
         author_name(G)
         event(H,list(F,G))
         title(I)
         event(J,list(F,I))


% ======================================================
% [5]     Find out what borrower last checked out a particular copy of a book.
% ======================================================
```

If
    a staff user enters an id of a copy
    and the copy is checked out to a borrower
then
    LibDB displays the name of the borrower.

```
% ------------------------------------------------------------
% [5]    Diskursrepräsentationsstruktur
% ------------------------------------------------------------

[E]
named(E,'LibDB')

IF
   [A,B,C,D]
   staff_user(A)
   identification(B)
   copy(C)
   of(B,C)
   event(D,enter(A,B))
   THEN
      [F,G,H]
      name(F)
      borrower(G)
      of(F,G)
      event(H,display(E,F))
```


6.3 Nachprüfung

Im Unterschied zu bestehenden Subsprachen und kontrollierten natürlichen Sprachen, die für die Spezifikation eingesetzt werden, bildet eine sachverhaltsorientierte SAS in ACE eine textuelle Sicht auf eine formale Spezifikation in Logik. Die eindeutige Übersetzbarkeit der vorliegenden SAS weist der textuellen Sicht die Bedeutung der zugrundeliegenden formalen Logiksprache (d.i. eine strukturierte Form der Prädikatenlogik) zu. Die Korrespondenz zwischen der SAS in ACE und der formalen Sprache wird dadurch erreicht, dass der Benutzer beim Schreiben der SAS einem einzigen abstrakten Satzbauplan und einer kleinen Anzahl von Schreibprinzipien folgt, die auf die Logiksprache abgestimmt sind. Alle syntaktischen Konstruktionen haben eine genau definierte Verwendungsweise, so dass sich strukturelle Mehrdeutigkeiten immer auf ein Schreibprinzip zurückführen lassen. Die mit der Sprache ACE vertrauten Benutzer wissen immer, was sie mit den syntaktischen Konstruktionen tun, wenn sie den Satzbauplan korrekt verwenden und die Schreibprinzipien kennen.

Inwieweit erfüllt nun eine SAS in ACE auch die Qualitätsattribute (korrekt, eindeutig, vollständig, verifizierbar, konsistent und verständlich), die wir für eine ideale Spezifikation postuliert haben? Im Unterschied zu einer formalen Spezifikation ist eine SAS in ACE sicher gut lesbar und verständlich. Die Validierung der Anforderungen kann von allen am Software-Entwicklungsprozess beteiligten Personen problemlos durch Inspektion geleistet werden. Entdeckte Fehler können unmittelbar in der Benutzersprache ACE behoben werden. Die Korrekturen müssen nicht in einer formalen Sprache kodiert werden (was wiederum zu Fehlern bei der Umsetzung führen kann). Dank der Schreibprinzipien von ACE und der deterministischen Parsingstrategie resultiert eine Übersetzung in Logik, die eindeutig und formal ist. Die übersetzte SAS ist nicht nur ein konzeptionelles Modell des zu entwickelnden Softwareproduktes, sondern auch ein Verhaltensmodell dessen Korrektheit, Konsistenz und Vollständigkeit überprüft werden kann. Neben der Inspektion kann der Benutzer die übersetzte Spezifikation ausführen und unmittelbare Erfahrungen mit den spezifizierten Sachverhalten in anwendungsspezifischen Konzepten machen. Der Benutzer kann bei der Ausführung das Verhalten der einzelnen Transaktionen in ACE beobachten. Da es sich bei einer SAS in ACE um eine formale Spezifikation in Logik handelt, ist die spätere Verifikation des Softwareproduktes gegenüber den faktischen Anforderungen zumindest theoretisch nicht ausgeschlossen.

Im nächsten Kapitel wird die Arbeitsweise des Attempto Spezifikationssystems an einem kleinen Ausschnitt aus der Bibliotheksspezifikation verdeutlicht.

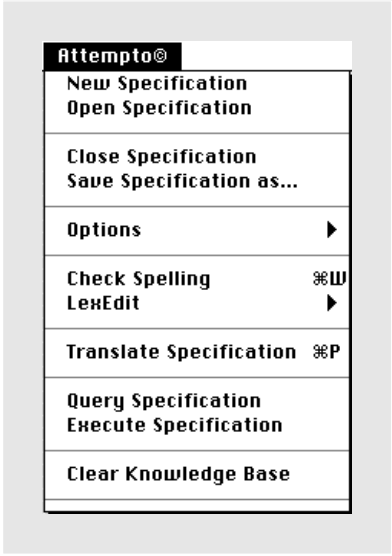
7. Arbeiten mit dem Attempto Spezifikationssystem

Das Attempto Spezifikationssystem zeichnet sich durch eine einfache Handhabung aus. Die verschiedenen Funktionen, die dem Anwendungsspezialisten beim Schreiben und Validieren einer ACE Spezifikation zur Auswahl stehen, werden hier kurz vorgestellt. Ausserdem wird die Arbeitsweise des Systems schrittweise an einem Ausschnitt der Bibliotheksspezifikation verdeutlicht.

7.1 Die Funktionalität des Attempto Systems

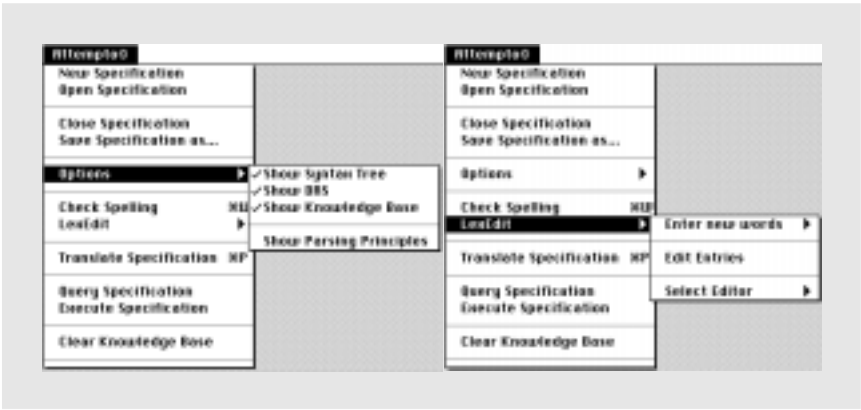
Die vorliegende prototypische Version des Attempto Systems ist in LPA MacProlog32 [Johns 94] implementiert. MacProlog32 verfügt über eine gut ausgebaute Programmierumgebung für die Gestaltung von benutzerfreundlichen Schnittstellen auf dem Apple Macintosh Computer.

Nach dem Starten von Attempto steht dem Anwendungsspezialisten die gesamte Systemfunktionalität auf einen Blick im **Attempto** Menüfenster zur Verfügung.



Mit dem Befehl **New Specification** wird ein leeres '**Untitled Specification Window**' erzeugt, das auf den gewünschten Dokumentnamen umbenannt und anschliessend mit dem Befehl **Save Specification as**... gesichert werden kann. Der Befehl **Open Specification** aktiviert ein Dialogfenster, aus welchem der Anwendungsspezialist eine bereits existierende ACE Spezifikation auswählen und laden kann. Zum Schliessen einer aktiven Spezifikation wird der Befehl **Close Specification** verwendet. Dieser Befehl schliesst nicht nur das Spezifikationsfenster, sondern löscht auch die aktuelle Wissensbasis. Mit dem Befehl **Check Spelling** kann der ACE Spezifikationstext im aktiven Spezifikationsfenster vor der Übersetzung auf unbekannte Wörter überprüft werden. Der Anwendungsspezialist lässt die ACE Spezifikation mit dem Befehl **Translate Specification** übersetzen. Um die übersetzte Spezifikation durch Fragen zu validieren, wählt der Anwendungsspezialist den Befehl **Query Specification** aus, welcher ein Dialogfenster für ACE Fragesätze erzeugt. Um die übersetzte Spezifikation durch Ausführung zu validieren, ruft der Anwendungsspezialist den Befehl **Execute Specification** auf, wodurch die übersetzte Spezifikation durch einen Metainterpreter interaktiv ausgeführt wird. Der Befehl **Clear Knowledge Base** löscht die Wissensbasis bei Bedarf.

Das **Attempto** Menüfenster enthält zwei Befehle, die in Untermenüs verzweigen. Der Menübefehl **Options** verzweigt zu Befehlen, die optional den aktuellen Syntaxbaum, die Diskursrepräsentationsstruktur und den Inhalt der Prolog Wissensbasis anzeigen, oder die Parsingprinzipien in den paraphrasierten Text einblenden.

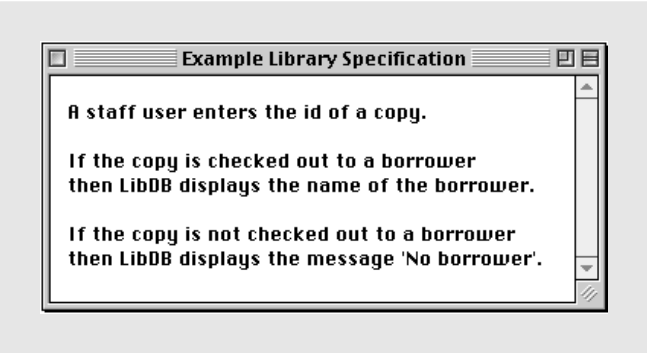


Der Menübefehl **LexEdit** verschafft Zugang zum lexikalischen Editor. Der Anwendungsspezialist hat die Möglichkeit, neue Inhaltswörter während des Spezifikationsprozesses ins Lexikon einzugeben und existierende lexikalische Einträge von Inhaltswörtern zu editieren. Gegenwärtig stehen zwei Typen von lexiklischen Editoren zur Auswahl: einer für den Anwendungsspezialisten und einer für den linguistischen Experten.

7.2 Verarbeitung der Bibliotheksspezifikation in ACE

Zur Illustration der Arbeitsweise des Attempto Systems folgt ein Ausschnitt aus der Bibliotheksspezifikation in ACE. Es wird gezeigt, wie der Anwendungsspezialist die ACE Spezifikation für die 5. Transaktion der Problembeschreibung entwickelt und durch das Attempto System verarbeiten lässt. In Ergänzung zur Problembeschreibung wird zusätzlich spezifiziert, wie sich das System verhält, wenn die Vorbedingungen der Transaktion nicht erfüllt sind. Der Systembenutzer erhält dann die Information, dass kein Ausleiher *'No borrower'* zum gegenwärtigen Zeitpunkt eine bestimmte Kopie ausleiht.

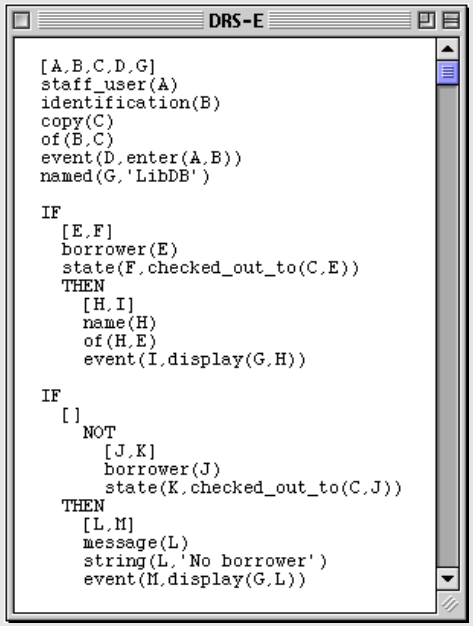
Der Anwendungsspezialist erzeugt zuerst mit dem Befehl **New Specification** ein neues Spezifikationsfenster. Anschliessend verfasst er den besagten Spezifikationstext, wobei er den ACE Schreibprinzipien folgt, die den Gebrauch der Sprache regeln.



Der Spezifikationsausschnitt besteht aus den folgenden ACE Konstruktionen:

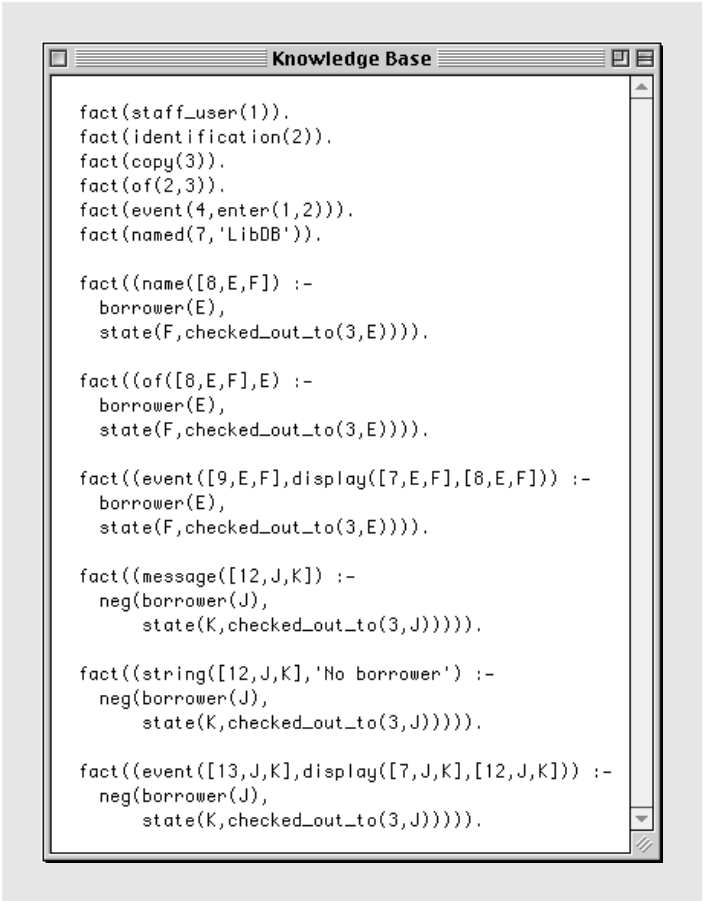
- zusammengesetzte Wörter, z.B. *staff user*
- Abkürzungen, z.B. *id* steht hier für *identification*
- *of*-Konstruktionen, z.B. *the id of a copy*
- Eigennamen, z.B. *LibDB*
- interne Negation, z.B. *is not checked out to a borrower*
- Quoted Strings, z.B. *'No borrower'*
- einfache Sätze, z.B. *a staff user enters the id of a copy*
- zusammengesetzte Sätze, z.B. das konditionale Satzgefüge *If … then …*
- anaphorische Bezüge von definiten Nominalphrasen auf indefinite Nominalphrasen, z.B. *the copy* ⇒ *a copy* und *the borrower* ⇒ *a borrower*.

Der Anwendungsspezialist startet den Übersetzungsprozess, indem er den Befehl **Translate Specification** aufruft. Der Chart-Parser nimmt den Spezifikationstext als Liste von Worten entgegen und erzeugt Syntaxbäume als grammatische Repräsentation der einzelnen Sätze, eine um Ereignisse und Zustände erweiterte Diskursrepräsentationsstruktur (DRS-E) als formale Repräsentation der spezifizierten Sachverhalte und eine Paraphrase in ACE als Rückmeldung für den Benutzer.



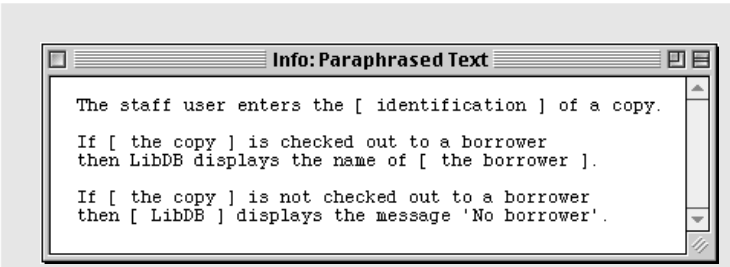
Die DRS-E besteht aus einer Menge von Diskursreferenten (*[A, B, C, ...]*) und einer Anzahl von Bedingungen für die Diskursreferenten in fester Reihenfolge. Die anaphorischen Referenzen und die Abkürzung, die im Spezifikationstext vorkommen, sind in der abgebildeten DRS-E bereits durch den DRS-Konstruktionsalgorithmus aufgelöst worden. Der Spezifikationstext ist genau dann wahr, wenn die Diskursreferenten - gemäss den Regeln der Diskursrepräsentationstheorie - auf Entitäten im intendierten Modell abgebildet werden können, so dass die in der DRS-E beschriebenen Bedingungen im Modell wahr sind.

Die DRS-E bildet als formale Repräsentation der ACE Spezifikation den Ausgangspunkt für die Validierung und die automatische Programmentwicklung. Optional kann die DRS-E in eine Prolog Knowledge Base übersetzt werden. Dazu müssen diejenigen Diskursreferenten, die als existentiell quantifizierte Variablen dargestellt sind, durch Skolemkonstanten ersetzt werden. Wenn existentiell quantifizierte Variablen im Skopus eines Allquantors stehen, führt das zu Skolemfunktionen. Die Skolemfunktionen werden als Listen (z.B. *[8|[E,F]]* bzw. *[8,E,F]*) dargestellt, mit einer Konstanten für den existentiellen Quantor als Kopf und den universell quantifizierten Variablen, die den existentiellen Quantor im Skopus haben, als Schwanz [vgl. Covington et al. 88].



Ein syntaktisches Problem entsteht, wenn eine *IF…THEN* DRS nach Prolog übersetzt wird. Eine Prolog Regel bzw. Hornklausel kann nicht mehr als ein Prädikat in der Konsequenz haben. Aus diesem Grund muss der Konsequenzteil einer implikativen DRS (*THEN …*) mit mehreren Bedingungen immer aufgespalten werden und auf eine Menge von Prolog Regeln verteilt werden. Die Prolog Klauseln werden in einem einheitlichen Format als Fakten assertiert, so dass sie durch einen Metainterpreter verarbeitet werden können. Negation wird durch das metalogische Prädikat *neg* angezeigt und ebenfalls durch den Metainterpreter verarbeitet [vgl. Fuchs forthcoming].

Zusätzlich zu diesen internen Repräsentationen, welche normalerweise für den Anwendungsspezialisten nicht sichtbar sind, erzeugt der Chart-Parser eine Paraphrase des Spezifikationstextes. Die Paraphrase zeigt dem Anwendungsspezialisten an, welche Interpretation das Attempto System aufgrund der Schreibprinzipien und der deterministischen Parsingstrategie gewählt hat. Eckige Klammern (*[]*) verdeutlichen alle Ersetzungen und Interpretationen, die der Parser vorgenommen hat. Entfernt man die eckigen Klammern, dann entsteht ein gültiger ACE Text, der vom System wiederum verarbeitet werden kann.



Aufgrund der Paraphrase entscheidet der Anwendungsspezialist, ob er die vom System vorgelegte Interpretation akzeptieren will, oder ob er bestimmte Textstellen umformulieren muss, um zu der intendierten Interpretation zu gelangen. Wenn die Eingabe mehrdeutig ist, dann schlägt das System seiner deterministischen Parsingstrategie folgend immer eine eindeutige Interpretation vor, die durch die Schreibprinzipien festgelegt sind.

Relativsätze werden in ACE beispielsweise immer an das letzterwähnte Nomen angebunden. Mit dem Befehl **Show Parsing Principles** kann sich der Anwendungsspezialist

dieses Parsingprinzip anzeigen lassen. Geschweifte Klammern (*{}*) markieren die Anbindung. Nehmen wir an, das Nomen *borrower* im Satz

LibDB displays the name of the borrower.

wird durch den Relativsatz *who has the copy* modifiziert, dann zeigt die Einblendung der Parsingprinzipien die folgende Darstellung in der Paraphrase:

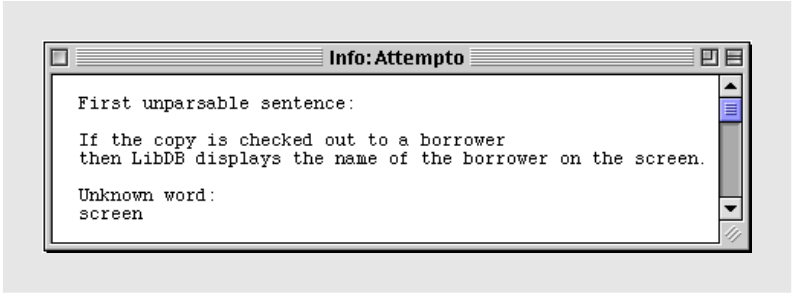
```
LibDB displays the name of {the borrower who has the copy}.
```

Ist der Anwendungsspezialist mit dem Ergebnis nicht zufrieden, dann muss er die Eingabe gemäss den Schreibprinzipien umformulieren und den Text mit dem Befehl **Translate Specification** neu übersetzen.

Wenn der Chart-Parser einen Satz nicht analysieren kann, dann erzeugt das Attempto System eine Fehlernachricht. Wenn das Inhaltswort *screen* zum Zeitpunkt der Analyse im folgenden um die Präpositionalphrase *on the screen* erweiterten konditionalen Satz

If the copy is checked out to a borrower
then LibDB displays the name of the borrower on the screen.

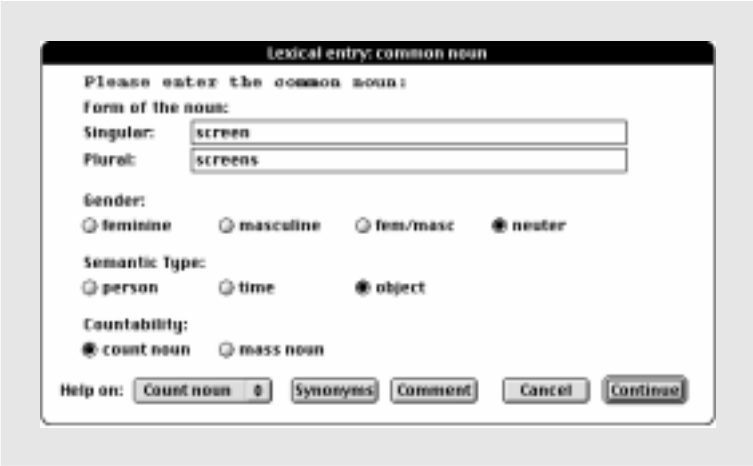
noch nicht im Lexikon verfügbar ist, dann schlägt die Analyse des Satzes fehl. Das Attempto System ruft automatisch den Spelling Checker auf, der auf das unbekannte Wort *screen* stösst und eine Fehlernachricht erzeugt:



Da es sich beim Wort *screen* um ein unbekanntes Inhaltswort handelt, das korrekt geschrieben ist, ruft der Anwendungsspezialist den lexikalischen Editor mit dem Befehl **Enter new word** auf und fügt das Wort zum Lexikon hinzu. Dazu braucht der

Anwendungsspezialist nur schulgrammatisches Wissen und kein linguistisches Expertenwissen.

Das untenstehende Dialogfenster zeigt, wie der Anwendungsspezialist das Nomen *screen* in das linguistische Lexikon einträgt. Er muss die Wortformen im Singular und Plural eingeben und das Geschlecht des Nomen bestimmen. Weiter ist zu entscheiden, ob das Nomen zur Klasse der zählbaren Nomen oder Massennomen gehört. Zusätzlich erhält das Nomen einen semantischen Typ zugewiesen. Optional kann der Benutzer jedes Inhaltswort mit einer Anzahl von Synonymen und Abkürzungen assoziieren und ausserdem informelle Erkennungsregeln für den Begriff in einem Kommentarfeld formulieren. Aus dieser Information wird der komplette Lexikoneintrag abgeleitet.

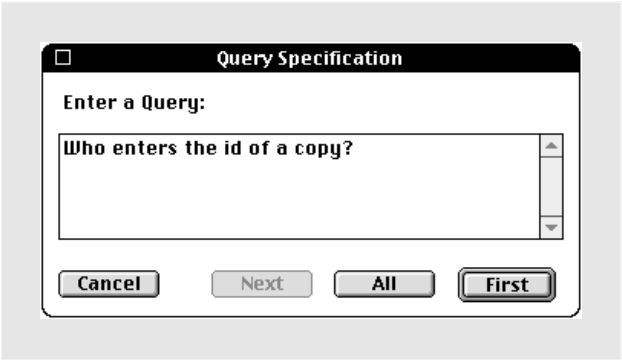


Neben der Schnittstelle für die Anwendungsspezialisten gibt es eine Expertenschnittstelle, die nicht nur das Eintragen und Verändern von Inhaltswörtern zulässt, sondern dem linguistischen Experten erlaubt, alle lexikalischen Einträge auf der Ebene der Merkmalstrukturen zu modifizieren. Für beide Benutzergruppen stehen Hilfefunktionen und Hilfetexte bereit, die den Aufbau des Lexikons unterstützen [vgl. Hamberger 95].

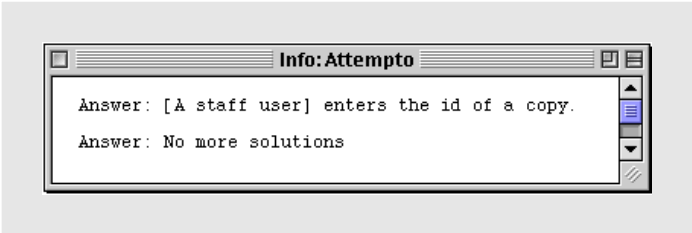
Die Verantwortung für lexikalische Ambiguität liegt bei den Anwendungsspezialisten; sie entscheiden, wie ein Wort in einer Spezifikation verwendet werden muss.

7.3 Fragebeantwortung

Der Anwendungsspezialist kann ACE Fragesätze dazu verwenden, um sich über die spezifizierten Sachverhalte zu orientieren. Mit Entscheidungsfragen lässt sich feststellen, ob ein beschriebener Sachverhalt wahr oder falsch ist. Mit Ergänzungsfragen lassen sich bestimmte Aspekte eines Sachverhaltes überprüfen. Der Befehl **Query Specification** erzeugt ein separates Dialogfenster, in welches der Anwendungsspezialist die Frage eingibt. Hier ist ein Beispiel für eine Ergänzungsfrage:



Technisch werden Fragen in Frage-DRSen übersetzt und durch logische Inferenz beantwortet. Wenn es auf eine Frage mehr als eine Antwort gibt, dann können die einzelnen Antworten schrittweise oder alle Antworten auf einmal generiert und angezeigt werden. Bei den Antworten handelt es sich wiederum um Sätze oder Konstruktionen in ACE.

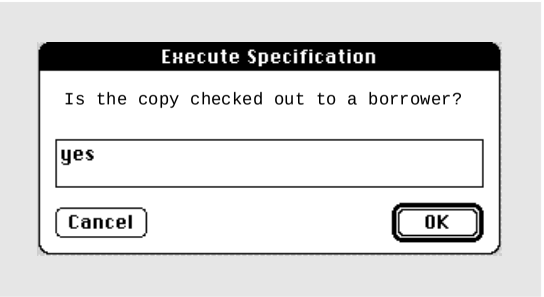


7.4 Ausführung

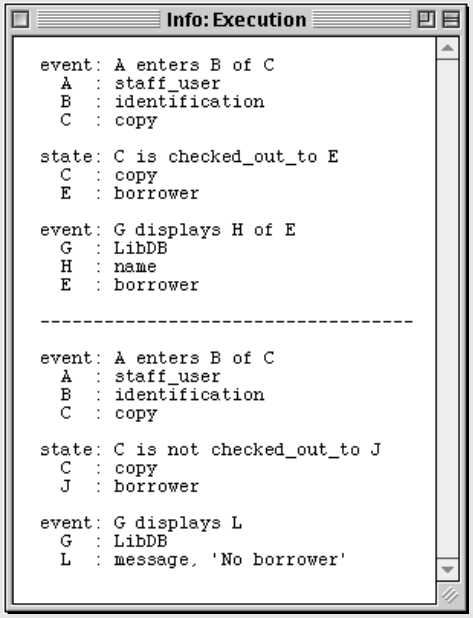
Neben der Fragebeantwortung bildet die DRS-E den Ausgangspunkt für die Validierung durch Ausführung. Mit dem Befehl **Execute Specification** kann der Anwendungsspezialist die formalisierte Spezifikation durch einen Metainterpreter symbolisch ausführen lassen. Das Verhalten der Spezifikation wird dadurch in den vertrauten Begriffen aus dem Anwendungsgebiet überprüfbar. Die Ausführung der Spezifikation verlangt jedoch zusätzliche Informationen über die Simulationsumgebung und über die aktuellen Sachverhalte, welche zur reinen ACE Spezifikation hinzugefügt werden müssen.

In der DRS-E gibt es Bedingungen, die nicht nur wahrheitsfunktional sind, sondern die bei der Ausführung der formalen Spezifikation Seiteneffekte (in der wirklichen Welt) verursachen. Die Seiteneffekte müssen durch Schnittstellenprädikate definiert werden, deren Komplexität von der Simulationsumgebung abhängig ist und die möglicherweise in einer Bibliothek zur Verfügung stehen.

Ausserdem verlangt die Ausführung Information über Sachverhalte, die entweder durch den Anwendungsspezialisten oder durch Fakten aus einer Wissensbasis zur Verfügung gestellt werden muss. Beispielsweise wird dem Anwendungsspezialisten bei der Ausführung des Spezifikationsausschnittes die untenstehende Entscheidungsfrage gestellt. Je nach Beantwortung der Frage führt das zu einem anderen Pfad durch die Spezifikation.



Die Ergebnisse der schrittweisen Ausführung der Spezifikation werden in das Fenster **Info: Execution** ausgegeben. Das untenstehende Fenster zeigt das Gesamtergebnis nach zweimaliger Ausführung der Spezifikation an. Beim ersten Mal ist die Frage *Is the copy checked out to a borrower?* mit *yes* beanwortet worden und beim zweiten Mal mit *no*.



Der Anwendungsspezialist kann bei der Ausführung die chronologische Reihenfolge der Ereignisse und Zustände beobachten und sich - in den vertrauten Begriffen aus dem Anwendungsgebiet - davon überzeugen, ob die Spezifikation unter dem operationalen Modell korrekt und vollständig ist.

8. Schlussfolgerungen und offene Probleme

> Somewhere between ridiculous pedantry and erroneous formulation there exists a reasonably precise way of specifying a problem in Attempto Controlled English (ACE).

Die vorgestellte kontrollierte natürliche Sprache ACE und das prototypische Spezifikationssystem Attempto beweisen, dass eindeutige und präzise Software-Anforderungsspezifikationen in einer Teilmenge der englischen Standardsprache geschrieben werden können, die maschinell gut durch einen Computer verarbeitbar ist. Die Sprache ACE besteht aus einer eingeschränkten Grammatik und einem Lexikon mit einem definierbaren und einem vordefinierten Teil. Die zulässigen lexikalischen Elemente und grammatischen Konstruktionen sind durch einen abstrakten Satzbauplan und 30 Schreibprinzipien greifbar gemacht. Der definierbare Teil des Lexikons umfasst die anwendungsspezifischen Inhaltswörter und der vordefinierte Teil die anwendungsunabhängigen Funktionswörter. Das interaktive Spezifikationssystem Attempto übersetzt die Paragraphen der ACE Spezifikation eindeutig in Diskursrepräsentationsstrukturen (und optional in Hornklauseln) und erzeugt während der Übersetzung eine Paraphrase, die über die automatisch vorgenommenen Substitutionen und strukturellen Interpretationen orientiert.

Die Sprache ACE kombiniert die Vorteile von informellen Spezifikationssprachen mit den Vorteilen von formalen Spezifikationssprachen und weist eine Reihe von herausragenden Eigenschaften auf. Erstens bietet ACE eine textuelle Sicht auf eine formale Sprache, so dass die Anwendungsspezialisten die zu spezifizierenden Sachverhalte unmittelbar in einer vertrauten und scheinbar informellen Notation auf der Sichtebene beschreiben können, ohne dass sie die natürlich-sprachlichen Ausdrücke zuerst in eine formale Sprache kodieren müssen. Zweitens führt ACE zu Spezifikationen, die für alle am Software-Entwicklungsprozess beteiligten Personen gut lesbar und besser verständlich sind, so dass die Korrektheit und Vollständigkeit der spezifizierten Sachverhalte durch Inspektion intersubjektiv überprüft werden kann. Drittens weist die eindeutige Übersetzbarkeit der textuellen Sicht in ACE in eine strukturierte Form der Prädikatenlogik - aufgrund einer kleinen Anzahl von Schreibprinzipien - der Sicht die Semantik der zugrundeliegenden Logiksprache zu. Die Korrespondenz zwischen textueller Sicht und formaler Sprache gleicht die Spannung zwischen Informalität und Formalität aus und bringt die Bedürfnisse von Anforderungsspezialisten und Softwareentwicklern in Einklang. Viertens informiert eine Paraphrase in ACE - neben den Schreibprinzipien - wie ein Satz vom Attempto System verarbeitet worden ist. Die Verarbeitung ist weitgehend syntaktisch und fügt kein aussersprachliches Weltwissen

zu einer Spezifikation hinzu. Es besteht eine strenge Arbeitsteilung: Das Attempto System verarbeitet die Sprache ACE syntaktisch durch logische Beweisführung und der Benutzer interpretiert die sprachlichen Elemente semantisch durch Modellbildung. Fünftens kann sich der Anwendungsspezialist über die formalisierten Sachverhalte oder Teilaspekte dieser Sachverhalte orientieren, indem er Fragen in ACE an das Spezifikationssystem richtet, die durch logische Inferenz in ACE beantwortet werden. Durch diese Vorgehensweise kann der Anwendungsspezialist Aufschluss hinsichtlich Korrektheit, Vollständigkeit und syntaktischer Widersprüche in der Spezifikation erhalten. Sechstens ist die übersetzte Spezifikation ausführbar, so dass ihr Verhalten in anwendungsspezifischen Konzepten - auf der Sichtebene - beobachtet und überprüft werden kann. Das hat den grossen Vorteil, dass die faktischen Anforderungen - zusätzlich zur Inspektion durch kritisches Lesen - in einer frühen Phase des Software-Entwicklungsprozesses durch Experimentieren validiert werden können.

Um einen kurzen abschliessenden Vergleich zu machen, stellen wir das Attempto/ACE System dem CLARE/CLE Spezifikationssystem [Macias & Pulman 95] gegenüber. Im Gegensatz zum CLARE/CLE System, bei dem einfache Sätze und Phrasen durch eine menügesteuerte Metagrammatik zusammengefügt werden, aber keine Aussagen vorliegen, welche Einschränkungen beim Schreiben der Spezifikation für die phrasale und die lexikalische Ebene gelten, sind diese Restriktionen für das Attempto/ACE System präzis in den Schreibprinzipien festgelegt. Im CLARE/CLE System lösen die Systembenutzer, die nach der Verarbeitung verbleibenden Ambiguitäten auf, indem sie eine passende Paraphrase aus einer Menge von Paraphrasen auswählen. Das System erzeugt also immer mit grossem Aufwand alle Lesarten für einen Satz. Nicht so das Attempto/ACE System, bei dem aufgrund der deterministischen Parsingstrategie und der Schreibprinzipien für jeden Satz genau eine Lesart berechnet wird, die der geübte Systembenutzer schon beim Schreiben der Spezifikation erzwingen kann. Das CLARE/CLE System verlangt für jedes neue Anwendungsgebiet die Bereitstellung von fachspezifischem Wissen. Das ist beim Attempto/ACE System nicht der Fall, da die Spezifikation die einzige Informationsquelle ist und die Verarbeitung vollkommen syntaktisch erfolgt.

Um die Sprache ACE in industriellen Projekten nutzbar zu machen, sind zusätzliche Anstrengungen nötig. Es muss sorgfältig geprüft werden, welche Konstruktionen die Expressivität der Sprache weiter erhöhen können, so dass ACE zugleich leicht handhabbar und effizient maschinell verarbeitbar bleibt. Ferner sind Strukturierungsmechanismen für grosse Spezifikationen zu erarbeiten, die es erlauben, eine Spezifi-

kation in lose gekoppelte Paragraphen zu unterteilen, die nach der Übersetzung als Module oder in einer abstrakteren Form als logische Schemata zur Verfügung stehen. In diesem Zusammenhang ergeben sich folgende interessante Forschungsschwerpunkte:

- *Behandlung von Pluralphänomen*
  Die Verarbeitung von natürlich-sprachlichen Pluralphänomenen ist sehr komplex, da Pluralsätze mehrere mögliche Lesarten haben. Für ACE ist die Einführung von spezialisierten Diskurssignalen (*both, each, together, separately, as a group*) denkbar, die explizit eine kollektive, distributive oder kumulative Lesart auf der Textoberfläche erzwingen (z.B. *Two staff users together work at the check-out counter*). Für Nominalphrasen mit kardinalen Determinatoren (*one, two, three*) und Massausdrücken (z.B. *three litre of water*) sind adäquate logische Darstellungsformen zu bestimmen.

- *Einführung von temporalen Nebensätzen*
  In ACE bestimmt die textuelle Reihenfolge der Verbalphrasen die zeitliche Anordnung der zugrundeliegenden Ereignisse. Durch vordefinierte Diskurssignale (*at the same time* und *in any temporal order*) kann diese Reihenfolge im Fall von koordinierten Verbalphrasen explizit aufgehoben werden, um Gleichzeitigkeit oder Nichtdeterminismus zu erzwingen. Es ist zu untersuchen, inwieweit die Verwendung von temporalen Nebensätzen, die eine vorzeitige, gleichzeitige oder nachzeitige Relation zwischen den Ereignissen im Neben- und Hauptsatz durch einleitende Konjunktionen (*after, when, before*) anzeigen, die Expressivität von ACE erhöhen können.

- *Behandlung von Nebenläufigkeit*
  In der Sprache ACE gibt es gegenwärtig keinen Mechanismus bzw. keine vordefinierten Diskurssignale, wodurch sich Nebenläufigkeit explizit beschreiben lässt. Das heisst, es können keine ineinander verzahnten Abläufe spezifiziert werden. Zwei Lösungen sind denkbar, die auf unterschiedlichen Ebenen ansetzen: Entweder muss die Nebenläufigkeit von zwei oder mehr Paragraphen auf einer Metaebene indizierbar sein oder die Nebenläufigkeit muss den betreffenden ACE Sätzen in einem Paragraphen durch ein Diskurssignal (*when*) aufgeprägt werden können (z.B. *When a staff user enters the password then LibDB checks it. When a staff user presses the key 'Stop' then LibDB terminates the transaction.*).

- *Strukturierung von Spezifikationen*
  Eine Spezifikation wird als ein ACE Text betrachtet, der aus einem oder mehr Paragraphen besteht. Ein Paragraph ist eine unabhängige (modulare) Einheit, die als Ganzes in eine Diskursrepräsentationsstruktur übersetzt wird. Für grössere Spezifikationen führt dieser Ansatz zu Problemen, da man einerseits Bezüge zwischen den einzelnen Paragraphen herstellen will und andererseits mit modularen (wiederverwendbaren) Komponenten arbeiten möchte. Strukturierungsmechanismen sind nötig, um kohäsive Paragraphen zu koppeln.

- *Definition von Constraints (Restriktionen)*
  Das Attempto System verfügt zur Zeit über keinen Mechanismus zur separaten Definition und Verwaltung von Constraints. Constraints könnten dazu benutzt werden, um Rand- oder Nebenbedingungen für eine Spezifikation festzulegen. Durch die lokale Verwaltung der Constaints, die über Constraint-Variablen mit der Spezifikation interagieren, würde sich die Ausdruckskraft der Spezifikation erhöhen. Damit Constraints generell einsetzbar sind, müssten in ACE verschiedene Möglichkeiten angeboten werden, um Constraints zu definieren. Zu ihnen sollten extensionale Beschreibungsmöglichkeiten (Aufzählung) gehören, aber auch Mittel mit denen Constraints intensional (durch Ausdrücke) definiert werden können.

Obwohl ACE gut geeignet ist, um präzise Spezifikationen zu schreiben, muss kontrollierte natürliche Sprache nicht immer die erste Wahl sein. Gewisse Teile einer Spezifikation, z.B. die Bestimmung der Benutzerschnittstelle oder die Beschreibung von Algorithmen, verlangen nach anderen adäquateren Notationen. Deshalb ist es wichtig, die Sprache ACE durch graphische und algebraische Notationen zu ergänzen.

Um die Adäquatheit und Akzeptanz von ACE zu testen, ist die Sprache an weiteren Spezifikationsbeispielen und in industriellen Projekten zu erproben - und wo nötig zu verbessern. Gegenwärtig wird das Attempto System nach SICStus Prolog portiert, um die Sprache ACE über das Internet einem breiteren Publikum verfügbar zu machen. Dadurch erhoffen wir uns konstruktive Rückmeldungen, die schlussendlich wieder der Weiterentwicklung der Sprache ACE zugute kommen.

Literaturverzeichnis

[Adriaens & Schreurs 92]
G. Adriaens, D. Schreurs, From Cogram to Alcogram: Towards a Controlled English Grammar Checker, Proceedings COLING 92, pp. 595-601, 1992.

[Adriaens 94]
G. Adriaens, The LRE SECC Project: Simplified English Grammar and Style Correction in an MT Framework, Proceedings of the 1st Language Engineering Convention (Paris), pp. 1-8, 1994.

[Adriaens & Macken 95]
G. Adriaens, L. Macken, Technological evaluation of a controlled language application: precision, recall, and convergence tests for SECC, Proceedings of the Sixth International Conference on Theoretical and Methodological Issues in Machine Translation (TMI-95), Vol. 1, pp. 123-141, 1995.

[Adriaens 96a]
G. Adriaens (chairman), Proceedings of the First International Workshop on Controlled Language Applications, CLAW 96, University of Leuven, Belgium, March 1996.

[Adriaens 96b]
G. Adriaens, SECC: Using Text Structure Information to Improve Checker Quality and Coverage, Proceedings of the First International Workshop on Controlled Language Applications, CLAW 96, University of Leuven, Belgium, pp. 226-232, March 1996.

[AECMA 95]
The European Association of Aerospace Industries (AECMA), AECMA Simplified English, AECMA Document: PSC-85-16598, A Guide for the Preparation of Aircraft Maintenance Documentation in the International Aerospace Maintenance Language, Issue 1, September 1995.

[Alexejew et al. 73]
P. M. Alexejew, W. M. Kalinin, R. G. Piotrowski, Sprachstatistik, Wilhelm Fink Verlag, München, 1973.

[Allen 91]
J. Allen, Natural Language, Knowledge Representation, and Logical Form, Technical Report 367, Computer Science Department, University of Rochester, 1991.

[Allen 95]
J. Allen, Natural Language Understanding, Second Edition, The Benjamin/Cummings Publishing Company, Redwood City, California, 1995.

[Almqvist & Sågvall Hein 96]
I. Almqvist, A. Sågvall Hein, Defining ScaniaSwedish - a Controlled Language for Truck Maintenance, Proceedings of the First International Workshop on Controlled Language Applications, CLAW 96, University of Leuven, Belgium, pp. 159-165, March 1996.

[Alshawi 92]
H. Alshawi (ed.), The Core Language Engine, MIT Press, Cambridge, 1992.

[Alshawi et al. 92]
H. Alshawi, D. Carter, R. Crouch, S. Pulman, M. Rayner, A. Smith, CLARE, a Contextual Reasoning and Cooperative Response Framework for the Core Language Engine, Final Report, SRI International, December 1992.

[Arnold et al. 95]
D. Arnold, L. Balkan, S. Meijer, R. L. Humphreys, L. Sadler, Machine Translation: An Introductory Guide. Available over the Internet from <http://clwww.essex.ac.uk/~doug/book/book.html>, 21.12. 1995.

[Asher 93]
N. Asher, Reference to Abstract Objects in Discourse, Studies in Linguistics and Philosophy, Vol. 50, Kluwer Academic Publishers, Dordrecht, 1993.

[Baker et al. 94]
K. L. Baker, A. M. Franz, P. W. Jordan, T. Mitamura, E. H. Nyberg, 3rd, Coping With Ambiguity in a Large-Scale Machine Translation System, Proceedings of COLING-94, 1994. Available over the Internet from <http://www.lti.cs.cmu.edu/Research/Kant/>, 14.4.1997.

[Balzer et al. 78]
R. Balzer, N. Goldman, D. Wile, Informality in Program Specifications, IEEE Transactions on Software Engineering, Vol. 4, No. 2, pp. 94-102, Mach 1978.

[Balzer 85]
R. M. Balzer, A 15 Year Perspective on Automatic Programming. IEEE Transactions on Software Engineering, Vol. 11, No. 11, pp. 1257-1268, November 1985.

[Barwise & Cooper 81]
J. Barwise, R. Cooper, Generalized quantifiers and natural language, Linguistics and Philosophy 4, pp. 159-219, 1981.

[Bennett 95]
P. Bennett, A Course in Generalized Phrase Structure Grammar, UCL Press, London, 1995.

[Bonzi 90]
S. Bonzi, Syntactic Patterns in Scientific Sublanguages: A Study of Four Disciplines, Journal of the American Society for Information Science, Vol. 41, No. 2, pp. 121-131, 1990.

[Börger & Rosenzweig 95]
E. Börger, D. Rosenzweig, A Mathematical Definition of Full Prolog, Science of Computer Programming, Vol. 24, No. 3, pp. 249-286, June 1995.

[Borsley 91]
R. D. Borsley, Syntactic Theory, Edward Arnold, London, 1991.

[Bowen & Hinchey 95]
J. P. Bowen, M. G. Hinchey, Seven More Myths of Formal Methods, IEEE Software, Vol. 12, No. 3, pp. 34-40, July 1995.

[Braun et al. 96]
H. v. Braun, W. Hesse, H. B. Kittlaus, G. Scheschonk, Ist die Welt objektorientiert? Von der natürlichsprachlichen Weltsicht zum OO-Modell, in: E. Ortner, B. Schienmann, H. Thoma (Hrsg.): Natürlichsprachlicher Entwurf von Informationssystemen - Grundlagen, Methoden, Werkzeuge, Anwendungen, GI-Workshop, Tutzing, 28.-30. Mai, pp. 280-295, 1996.

[Briscoe et al. 87]
T. Briscoe, C. Grover, B. Boguraev, J. Carroll, The ALVEY Natural Language Tools Project Grammar: A Large Computational Grammar, Technical Report, ALVEY Documents, Cambridge University, Computer Laboratory, UK, 1987.

[Capindale & Crawford 89]
R. A. Capindale, R. G. Crawford, Using a natural language interface with casual users, International Journal Man-Machine Studies, Vol. 32, pp. 341-362, 1989.

[Carbonell et al. 92]
J. G. Carbonell, T. Mitamura, E. H. Nyberg, 3rd, The KANT Perspective: A Critique of Pure Transfer (and Pure Interlingua, Pure Statistics, ...), Center for Machine Translation, Carnegie Mellon University, Pittsburgh, 1992. Available over the Internet from <http://www.lti.cs.cmu.edu/Research/Kant/>, 14.4.1997.

[Castell & Hernández 95]
N. Castell, A. Hernández, Filtering Software Specifications Written in Natural Language, in: C. Pinto-Ferreira (ed.), N. J. Mamede (coed.), Progress in artificial intelligence, EPIA 95, Springer, Berlin, pp. 447-455, 1995.

[Chevalier et al. 78]
L. Chevalier, J. Dansereau, G. Poulin, TAUM-METEO: Description du Système. Université de Montréal, Canada, 1978.

[Chierchia & McConnell-Ginet 90]
G. Chierchia, S. McConnell-Ginet, Meaning and Grammar, MIT Press, Cambridge, Mass., 1990.

[Chierchia 95]
G. Chierchia, Dynamics of Meaning, Anaphora, Presupposition, and the Theory of Grammar, The University of Chicago Press, Chicago, 1995.

[Chomsky 57]
N. Chomsky, Syntactic Structures, Mouton, The Hague, 1957.

[Chomsky 70]
N. Chomsky, Remarks on nominalization, in: R. A. Jacobs, P. S. Rosenbaum (eds.): Readings in English transformational grammar, Ginn & Co., Waltham, Mass., pp. 184-221, 1970.

[Chomsky 86]
N. Chomsky, Barriers, MIT Press, Cambridge, Mass., 1986.

[Clarke & Wing 96]
E. Clarke, J. Wing, Formal Methods: State of the Art and Future Directions, CMU Computer Science, Technical Report CMU-CS-96-178, August 1996.

[Collins & Loftus 75]
A. M. Collins, E. F. Loftus, A spreading-activation theory of semantic processing, Psychological review, 82, pp. 407-428, November 1975.

[Covington et al. 88]
M. A. Covington, D. Nute, N. Schmitz, D. Goodman, From English to Prolog via Discourse Representation Theory, Research Report 01-0024, Artificial Intelligence Programs, University of Georgia, 1988.

[Covington 94a]
M. A. Covington, Natural Language Processing for Prolog Programmers, Prentice Hall, New Jersey, 1994.

[Covington 94b]
M. A. Covington, GULP 3.1: An Extension of Prolog for Unification-Based Grammar, Research Report AI-1994-06, Artificial Intelligence Center, University of Georgia, 1994.

[Covington 96]
M. A. Covington, Natural Language Plurals in Logic Programming Queries, Research Report AI-1996-01, Artificial Intelligence Center, University of Georgia, 1996.

[Crystal 87]
D. Crystal, The Cambridge Encyclopedia of Language, Cambridge University Press, Cambridge, 1987.

[Dankel et al. 94]
D. D. Dankel, M. S. Schmalz, K. S. Nielsen, Understanding Natural Language Software Specifications, Proc. AI-94, Paris, France, 1994.

[Davis 90]
A. M. Davis, Software Requirements, Analysis and Specification, Prentice Hall, New Jersey, 1990.

[Davis et al. 93]
A. M. Davis, S. Overmyer, K. Jordan, J. Caruso, F. Dandashi, A. Dinh, G. Kincaid, G. Ledeboer, P. Reynolds, P. Sitaram, A. Ta, M. Theofanos, Identifying and Measuring Quality in a Software Requirements Specification, in: Proc. Software Metrics Symposium, IEEE CS Press, Los Alamitos, California, pp. 141-153, 1993.

[Davis 94]
A. M. Davis, Requirements Engineering, in: J. J. Marciniak (ed.), Encyclopedia of Software Engineering, Vol. 2, Wiley & Sons, New York, pp. 1043-1054, 1994.

[Dodd 90]
T. Dodd, Prolog: A Logical Approach, Oxford University Press, Oxford, 1990.

[Douglas & Hurst 96]
S. Douglas, M. Hurst, Controlled Language Support for Perkins Approved Clear English (PACE), Proceedings of the First International Workshop on Controlled Language Applications, CLAW 96, University of Leuven, Belgium, pp. 93-105, March 1996.

[Fanselow & Felix 87]
G. Fanselow, S. W. Felix, Sprachtheorie, Eine Einführung in die Generative Grammatik, Band 2: Rektions- und Bindungstheorie, UTB 1442, Francke, Tübingen, 1987.

[Farke 94]
H. Farke, Grammatik und Sprachverarbeitung: zur Verarbeitung syntaktischer Ambiguität, Westdeutscher Verlag, Opladen, 1994.

[Farrington 96]
G. Farrington, AECMA Simplified English, An Overview of the International Aerospace Maintenance Language, Proceedings of the First International Workshop on Controlled Language Applications, CLAW 96, University of Leuven, Belgium, pp. 1-21, March 1996.

[Fromherz 93]
M. P. J. Fromherz, A Methodology for Executable Specifications - Combining Logic Programming, Object-Orientation and Multiple Views, PhD thesis, Department of Computer Science, University of Zurich, 1993.

[Fuchs 89]
N. E. Fuchs, SimpleMat (Customer´s View), Working Paper, Department of Computer Science, University of Zurich, November 1989.

[Fuchs 90]
N. E. Fuchs, Kurs in Logischer Programmierung, Springer, Wien, 1990.

[Fuchs 92]
N. E. Fuchs, Specifications Are (Preferably) Executable, Software Engineering Journal, Vol. 7, No. 5, pp. 323-334, September 1992.

[Fuchs et al. 94]
N. E. Fuchs, H. F. Hofmann, R. Schwitter, Specifying Logic Programms in Controlled Natural Language, Report 94.17, Department of Computer Science, University of Zurich, 1994.

[Fuchs & Schwitter 95]
N. E. Fuchs, R. Schwitter, Specifying Logic Programs in Controlled Natural Language, Proceedings CLNLP 95, ELSNET/COMPULOG-NET/EAGLES Workshop on Computational Logic for Natural Language Processing, University of Edinburgh, April 1995.

[Fuchs & Schwitter 96]
N. E. Fuchs, R. Schwitter, Attempto Controlled English (ACE), Proceedings of the First International Workshop on Controlled Language Applications, CLAW 96, University of Leuven, Belgium, pp. 124-136, March 1996.

[Fuchs & Robertson 96]
N. E. Fuchs, D. Robertson, Declarative specifications, The Knowledge Engineering Review, Vol. 11, No. 4, pp. 317-331, 1996.

[Fuchs forthcoming]
N. E. Fuchs, Declarative Executable Specifications, Habilitation, Department of Computer Science, University Zurich, forthcoming.

[Gal et al. 91]
A. Gal, G. Lapalme, P. Saint-Dizier, H. Somers, Prolog for Natural Language Processing, John Wiley & Sons, Chichester, 1991.

[Gazdar et al. 85]
G. Gazdar, E. Klein, G. Pullum, I. Sag, Generalized Phrase Structure Grammar, Harvard University Press, Cambridge, Mass., 1985.

[Gazdar & Mellish 89]
G. Gazdar, C. Mellish, Natural Language Processing in PROLOG, An Introduction to Computational Linguistics, Addison-Wesley, Wokingham, 1989.

[Goyvaerts 96]
P. Goyvaerts, Controlled English, Curse or Blessing? - A User´s Perspective, Proceedings of the First International Workshop on Controlled Language Applications, CLAW 96, University of Leuven, Belgium, pp. 137-142, March 1996.

[Graham 94]
L. K. Graham, An Implementation of Plurality in Discourse Representation Theory, Research Report AI-1994-04, Artificial Intelligence Center, University of Georgia, 1994.

[Gravell & Henderson 96]
A. Gravell, P. Henderson, Executing formal specifications need not be harmful, Software Engineering Journal, Vol. 11, No. 2, pp. 104-110, March 1996.

[Green & Morgan 96]
G. M. Green, J. L. Morgan, Practical guide to syntactic analysis, CSLI Lecture Notes, No. 67, Center for the Study of Language and Information, Stanford University, California, 1996.

[Greenbaum 96]
S. Greenbaum, The Oxford English Grammar, Oxford University Press, New York, 1996.

[Grice 75]
H. P. Grice, Logic and conversation, in: P. Cole, J. Morgan (eds.), Syntax and Semantics, Vol. 3: Speech Acts, Academic Press, New York, pp. 41-58, 1975.

[Grishman & Kittredge 86]
R. Grishman, R. Kittredge (eds.), Analyzing Language in Restricted Domains: Sublanguage Description and Processing, Lawrence Erlbaum Associates, Inc., Publishers, Hillsdale, New Jersey, 1986.

[Grumberg & Kurshan 94]
O. Grumberg, R. P. Kurshan, How Linear can branching-time be?, in: D. M. Gabbay (ed.), H. J. Ohlbach (coed.), Temporal Logic, First International Conference, ICTL '94, Bonn, Germany, July 1994, Proceedings, pp. 180-194, 1994.

[Guenthner et al. 86]
F. Guenthner, H. Lehmann, W. Schönfeld, A theory for the representation of knowledge, IBM Journal of Research and Development, Vol. 30, No. 1, pp. 39-56, January 1986.

[Guttag & Horning 93]
J. V. Guttag, J. J. Horning, Larch, Languages and tools for formal specification, Springer, New York, 1993.

[Haas 97]
S. W. Haas, Disciplinary Variation in Automatic Sublanguage Term Identification, Journal of the American Society for Information Science, Vol. 48, No. 1, pp. 67-79, 1997.

[Hall 90]
A. Hall, Seven Myths of Formal Methods, IEEE Software, Vol. 7, No. 5, pp. 11-19, September 1990.

[Hamberger 95]
B. Hamberger, LexEdit, Ein lexikalischer Editor für das natürlich-sprachliche Spezifikationssystem 'Attempto', Diplomarbeit in Informatik, Institut für Informatik, Universität Zürich, 1995.

[Harel 87]
D. Harel, Statecharts: a visual formalism for complex systems, Science of Computer Programming, Vol. 8, No. 3, pp. 231-274, 1987.

[Harris 68]
Z. S. Harris, Mathematical structures of language, Wiley, New York, 1968.

[Harris et al. 89]
Z. Harris, Result: Formulas of Information, in: Z. Harris, M. Gottfried, T. Ryckman, P. Mattick, Jr., A. Daladier, T. N. Harris, S. Harris, The Form of Information in Science, Analysis of an Immunology Sublanguage, Kluwer, Dordrecht, pp. 25-63, 1989.

[Hayes et al. 96]
P. Hayes, S. Maxwell, L. Schmandt, Controlled English Advantages for Translated and Original English Documents, Proceedings of the First International Workshop on Controlled Language Applications, CLAW 96, University of Leuven, Belgium, pp. 84-92, March 1996.

[Hess 91]
M. Hess, Recent Developments in Discourse Representation Theory, in: M. King (ed.), Communication with Men and Machines, Geneva, 1991. Available over the Internet from <http://www.ifi.unizh.ch/groups/CL/hess/oldpublications.html>, 12.10.1996.

[Hindle & Rooth 93]
D. Hindle, M. Rooth, Structural Ambiguity and Lexical Relations, Computational Linguistics, Special Issue on Using Large Corpora: I, Vol. 19, No. 1, pp. 103-120, March 1993.

[Hirst 87]
G. Hirst, Semantic interpretation and the resolution of ambiguity, Studies in Natural Language Processing, Cambridge University Press, Cambridge, 1987.

[Hoare 87]
C. A. R. Hoare, An overview of some formal methods for program design, in: C. A. R. Hoare, C. B. Jones, Essays in Computing Science, Prentice Hall, pp. 371-387, 1987.

[Hogger 90]
C. J. Hogger, Essentials of Logic Programming, Clarendon Press, Oxford, 1990.

[Hsia et al. 93]
P. Hsia, A. Davis, D. Kung, Status Report: Requirements Engineering, IEEE Software, Vol. 10, No. 6, pp. 75-79, November 1993.

[Huddleston 88]
R. Huddleston, English grammar: an outline, Cambridge University Press, Cambridge, 1988.

[IEEE Std 610.12 90]
IEEE Std 610.12, IEEE Standard Glossary of Software Engineering Terminology, The Institute of Electrical and Electronics Engineers, New York, 1990.

[IEEE Std 830 93]
IEEE Std 830, IEEE Recommended Practice for Software Requirements Specifications, The Institute of Electrical and Electronics Engineers, New York, 1994.

[Ishihara et al. 92]
Y. Ishihara, H. Seki, T. Kasami, A Translation Method from Natural Language Specifications into Formal Specifications Using Contextual Dependencies, in: Proceedings of IEEE International Symposium on Requirements Engineering, Jan. 4-6, 1993, San Diego, IEEE Computer Society Press, pp. 232-239, 1992.

[Jackendoff 77]
R. Jackendoff, X-Syntax: A Study of Phrase Structure, MIT, Cambridge, Mass., 1977.

[Jackson 95]
M. Jackson, Software Requirements & Specifications, a lexicon of practice, principles and prejudices, Addison-Wesley, Wokingham, 1995.

[Johns 94]
N. Johns, MacProlog32 User Guide, Logic Programming Associates Ltd, London, 1994.

[Jones 86]
C. B. Jones, Systematic Software Development Using VDM, Prentice-Hall, London, 1986.

[Kamp 81]
H. Kamp, A theory of truth and semantic representation, in: J. Groenendijk, T. M. V. Janssen, M. B. J. Stokhof (eds.), Formal Methods in the Study of Language, University of Amsterdam, pp. 277-322, 1981.

[Kamp & Reyle 93]
H. Kamp, U. Reyle, From Discourse to Logic, Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory, Kluwer Academic Publishers, Dordrecht, 1993.

[Kemmerer 85]
R. A. Kemmerer, Testing Formal Specifications to Detect Design Errors, IEEE Transactions on Software Engineering, Vol. 11, No. 1, pp. 32-43, January 1985.

[Kittredge 82]
R. I. Kittredge, 5. Sublanguages, American Journal of Computational Linguistics, Vol. 8, No. 2, pp. 79-84, April-June 1982.

[Kittredge 87]
R. I. Kittredge, The significance of sublanguage for automatic translation, in: S. Nirenburg (ed.), Machine translation, theoretical and methodological issues, Cambridge University Press, Cambridge, pp. 59-67, 1987.

[Kornai & Pullum 90]
A. Kornai, G. K. Pullum, The X-bar Theory of Phrase Structure, Language 66, pp. 24-50, 1990.

[Kowalski 85]
R. A. Kowalski, The relation between logic programming and logic specification, in: C. A. R. Hoare, J. C. Shepherdson, Mathematical Logic and Programming Languages, Prentice-Hall, Inc., New Jersey, pp. 11-24, 1985.

[Kowalski 94]
R. A. Kowalski, Logic without Model Theory, in: D. M. Gabbay (ed.), What is a Logical System?, Studies in Logic and Computation, Vol. 4, Clarendon Press, Oxford, pp. 35-71, 1994.

[Larsen et al. 96]
P. G. Larsen, J. Fitzgerald, T. Brookes, Applying Formal Specification in Industry, IEEE Software, Vol. 13, No. 3, pp. 48-56, May 1996.

[Lee & Sluizer 87]
S. Lee, S. Sluizer, SXL: An Executable Specification Language, Proceedings, Fourth International Workshop on Software Specification and Design, April 3-4, 1987, Monterey, California, Los Angeles, pp. 231-235, 1987.

[Lehrberger 86]
J. Lehrberger, Sublanguage Analysis, in: R. Grishman, R. Kittredge, Analyzing Language in Restricted Domains: Sublanguage Description and Processing, LEA, Hillsdale, New Jersey, pp. 19-38, 1986.

[Lenat & Guha 90]
D. B. Lenat, R. V. Guha, Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project, Addison-Wesley, Reading, Mass., 1990.

[Lewandowski 94]
Th. Lewandowski, Linguistisches Wörterbuch 2, UTB 1518, Quelle & Meyer, Heidelberg, 1994.

[Liskov et al. 81]
B. Liskov, R. Atkinson, T. Bloom, E. Moss, J. C. Schaffert, R. Scheifler, A. Snyder, CLU Reference Manual, Lecture Notes in Computer Science, Vol. 114, Springer, Berlin, 1981.

[Lloyd 87]
J. W. Lloyd, Foundations of Logic Programming, 2nd Edition, Springer, New York, 1987.

[Lloyd 94]
J. W. Lloyd, Practical Advantages of Declarative Programming, Invited Lecture, GULP-PRODE '94, Peñiscola (Spain), September 1994.

[Lonsdale et al. 94]
D. W. Lonsdale, A. M. Franz, J. R. R. Leavitt, Large-scale Machine Translation: An Interlingua Approach,

Center for Machine Translation, Carnegie Mellon University, Pittsburgh, 1994. Available over the Internet from <http://www.lti.cs.cmu.edu/Research/Kant/>, 14.4.1997.

[Losee & Haas 95]
R. M. Losee, S. W. Haas, Sublanguage Terms: Dictionaries, Usage, and Automatic Classification, Journal of the American Society for Information Science, Vol. 46, No. 7, pp. 519-529, 1995.

[Lux & Dauphin 96]
V. Lux, E. Dauphin, Corpus Studies: A Contribution to the Definition of a Controlled Language, Proceedings of the First International Workshop on Controlled Language Applications, CLAW 96, University of Leuven, Belgium, pp. 193-204, March 1996.

[Lyons 95]
J. Lyons, Linguistic Semantics, An Introduction, Cambridge University Press, Cambridge, 1995.

[Macias & Pulman 93]
B. Macias, S. G. Pulman, Natural language processing for requirements specifications, in: F. Redmill, T. Anderson (eds.), Safety-Critical Systems, Chapman & Hall, London, pp. 67-89, 1993.

[Macias & Pulman 95]
B. Macias, S. G. Pulman, A Method for Controlling the Production of Specifications in Natural Language, The Computer Journal, Vol. 38, No. 4, pp. 310-318, 1995.

[Marca 88]
D. A. Marca (Workshop Chairman), Fourth International Workshop on Software Specification and Design, Proceedings, April 3-4, 1987, Monterey, California, IEEE Computer Society Order Number 769, Los Angeles, 1987.

[Meyer 85]
B. Meyer, On Formalism in Specifications, IEEE Software, Vol. 2, No. 1, pp. 6-26, January 1985.

[Mitamura et al. 93]
T. Mitamura, E. H. Nyberg, 3rd, J. G. Carbonell, Automated Corpus Analysis and the Acquisition of Large, Multi-Lingual Knowledge Bases for MT, Proceedings of TMI-93, 1993. Available over the Internet from <http://www.lti.cs.cmu.edu/Research/Kant/>, 14.4.1997.

[Mitamura & Nyberg 95]
T. Mitamura, E. H. Nyberg, 3rd, Controlled English for Knowledge-Based MT: Experience with the KANT System, Proceedings of TMI-95, 1995. Available over the Internet from <http://www.lti.cs.cmu.edu/Research/Kant/>, 14.4.1997.

[Nelken & Francez 95]
R. Nelken, N. Francez, Automatic Translation of Natural Language System Specifications into Temporal Logic, Technical Report, Computer Science Department, The Technion, Haifa, Israel, 1995.

[Newton 92]
J. Newton, The Perkins experience, in: J. Newton (ed.), Computers in Translation: A Practical Appraisal, Routledge, London, pp. 46-57, 1992.

[Nonnenmann & Eddy 93]
U. Nonnenmann, J. K. Eddy, Software Testing with Kitss, IEEE Expert, Vol. 8, No. 4, pp. 25-30, August 1993.

[Nyberg & Mitamura 96]
E. H. Nyberg, 3rd, T. Mitamura, Controlled Language and Knowledge-Based Machine Translation: Principles and Practice, Proceedings of the First International Workshop on Controlled Language Applications, CLAW 96, University of Leuven, Belgium, pp. 74-83, March 1996. Available over the Internet from <http://www.lti.cs.cmu.edu/Research/Kant/>, 14.4.1997.

[Ogden 32]
C. K. Ogden, The Basic dictionary, Kegan Paul, Trench, Trubner, London, 1932.

[Ogden 38]
C. K. Ogden, Basic English, Kegan Paul, Trench, Trubner, London, 1938.

[Ortner & Schienmann 96]
E. Ortner, B. Schienmann, Normsprachlicher Entwurf von Informationssystemen - Vorstellung einer Methode, in: E. Ortner, B. Schienmann, H. Thoma (Hrsg.): Natürlich-sprachlicher Entwurf von Informationssystemen - Grundlagen, Methoden, Werkzeuge, Anwendungen, GI-Workshop, Tutzing, 28.-30. Mai, pp. 109-129, 1996.

[Parsons 90]
T. Parsons, Events in the Semantics of English: A Study in Subatomic Semantics, Current Studies in Linguistics, MIT Press, Cambridge, Mass., 1990.

[Partee et al. 90]
B. H. Partee, A. ter Meulen, R. E. Wall, Mathematical Methods in Linguistics, Studies in Linguistics and Philosophy, Vol. 30, Kluwer Academic Publishers, Dordrecht, 1990.

[Partsch 91]
H. Partsch, Requirements Engineering, Handbuch der Informatik, Band 5.5, R. Oldenbourg Verlag, 1991.

[Pereira & Shieber 87]
F. C. N. Pereira, S. M. Shieber, Prolog and Natural-Language Analysis, CSLI Lecture Notes, No. 10, Center for the Study of Language and Information, Stanford University, California, 1987.

[Peterson 81]
J. L. Peterson, Petri Net Theory and the Modeling of Systems, McGraw-Hill, New York, 1981.

[Pinkal 95]
M. Pinkal, Logic and Lexicon, The Semantics of the Indefinite, Studies in Linguistics and Philosophy, Vol. 56, Kluwer, Dordrecht, 1995.

[Pohl 93]
K. Pohl, The Three Dimensions of Requirements Engineering, in: C. Rolland, F. Bodart, C. Cauvet (eds.): Advanced Information Systems Engineering, CAiSE'93, Paris, Lecture Notes in Computer Science, Vol. 685, Springer, Berlin, pp. 275-292, 1993.

[Pollard & Sag 94]
C. Pollard, I. Sag, Head-Driven Phrase Structure Grammar, The University of Chicago Press, Chicago, 1994.

[Pulman et al. 93]
S. G. Pulman, H. Alshawi, D. M. Carter, R. S. Crouch, M. Rayner, A. G. Smith, CLARE: A Combined Language and Reasoning Engine, JFIT conference, 1993. Available over the Internet from <http://www.cam.sri.com/tr/ABSTRACTS.html>, 26.8.1996.

[Pulman 94]
S. G. Pulman, Natural Language Processing and Requirements Specification, Presentation at the Prolog Forum, Department of Computer Science, University of Zurich, February 1994.

[Pulman & Rayner 94]
S. Pulman, M. Rayner, Computer Processable Controlled Language, SRI International Cambridge Computer Science Research Centre, 1994.

[Pym 90]
P. J. Pym, Pre-editing and the use of simplified writing for MT: an engineer's experience of operating an MT system, in: P. Mayorcas (ed.), Translating and the Computer 10, Aslib, London, pp. 80-96, 1990.

[Pym 93]
P. J. Pym, Perkins Engines and Publications, Symposium Proceedings, Technology & Language in Europe 2000, The UK Perspective, London, pp. 1-7, January 1993.

[Rackow 92]
U. Rackow, On the Treatment of Compounds in Machine Translation, A Study, IBM Germany, Scientific Center, IWBS Report 221, Stuttgart, June 1992.

[Radford 88]
A. Radford, Transformational Grammar, Cambridge University Press, Cambridge, 1988.

[Reubenstein & Waters 91]
H. B. Reubenstein, R. C. Waters, The Requirements Apprentice: Automated Assistance for Requirements Acquisition, IEEE Transactions on Software Engineering, Vol. 17, No. 3, pp. 226-240, March 1991.

[Rich et al. 87]
C. Rich, R. C. Waters, H. B. Reubenstein, Proceedings, Fourth International Workshop on Software Specification and Design, April 3-4, 1987, Monterey, California, Los Angeles, pp. 79-86, 1987.

[Rueher 87]
M. Rueher, From Specifications to Design: An Approach Based on Rapid Prototyping, Proceedings, Fourth International Workshop on Software Specification and Design, April 3-4, 1987, Monterey, California, Los Angeles, pp. 126-133, 1987.

[Saeki et al. 89]
M. Saeki, H. Horai, H. Enomoto, Software Development Process from Natural Language Specifications, Proceedings of 11th International Conference on Software Engineering, IEEE Computer Society Press, pp. 64-73, 1989.

[Sager 90]
N. Sager, Computer Analysis of Sublanguage Information Structures, Annals of NY Academy of Sciences, The New York Academy of Sciences, Vol. 683, New York, pp. 161-179, 1990.

[Schach 96]
S. R. Schach, Classical and Object-Oriented Software-Engineering, 3rd ed., Irwin, Chicago, 1996.

[Schachtl 96]
S. Schachtl, Requirements for Controlled German in Industrial Applications, Proceedings of the First International Workshop on Controlled Language Applications, CLAW 96, University of Leuven, Belgium, pp. 143-149, March 1996.

[Schwitter & Fuchs 96]
R. Schwitter, N. E. Fuchs, Attempto – From Specifications in Controlled Natural Language towards Executable Specifications, in: E. Ortner, B. Schienmann, H. Thoma (Hrsg.): Natürlich-sprachlicher Entwurf von Informationssystemen - Grundlagen, Methoden, Werkzeuge, Anwendungen, GI-Workshop, Tutzing, 28.-30. May, pp. 163-177, 1996.

[Searle 69]
J. R. Searle, Speech Acts: An Essay in the Philosophy of Language, Cambridge University Press, Cambridge, 1969.

[Shieber 86]
S. M. Shieber, An Introduction to Unification-Based Approaches to Grammar, CSLI Lecture Notes, No. 4, Center for the Study of Language and Information, Stanford University, California, 1986.

[Slator et al. 86]
B. Slator, M. Anderson, W. Conley, Pygmalion at the Interface, Communications of the ACM, Vol. 29, No. 7, pp. 599-604, 1986.

[Smith 91]
G. W. Smith, Computers and Human Language, Oxford University Press, Oxford, 1991.

[Somers 87]
H. L. Somers, Valency and Case in Computational Linguistics, Edinburgh University Press, Edinburgh, 1987.

[Sommerville 96]
I. Sommerville, Software Engineering, Fifth Edition, Addison-Wesley, Wokingham, 1996.

[Spivey 89]
J. M. Spivey, The Z Notation: A Reference Manual, Prentice-Hall, London, 1989.

[Sterling & Shapiro 94]
L. Sterling, E. Shapiro, The Art of Prolog, Advanced Programming Techniques, Second Edition, MIT Press, Cambridge, 1994.

[Stokes 91]
D. A. Stokes, Requirements analysis, in: J. A. McDermid (ed.), Software Engineer´s Reference Book, Butterworth-Heinemann, Oxford, 1991.

[Tucker 87]
A. B. Tucker, Current strategies in machine translation research and development, in: S. Nirenburg (ed.), Machine translation, theoretical and methodological issues, Cambridge University Press, Cambridge, pp. 22-41, 1987.

[van der Eijk et al. 96]
P. van der Eijk, M. de Koning, G. van der Steen, Controlled language correction and translation, Proceedings of the First International Workshop on Controlled Language Applications, CLAW 96, University of Leuven, Belgium, pp. 64-73, March 1996.

[van Vliet 93]
H. van Vliet, Software Engineering, Principles and Practice, Wiley & Sons, Chichester 1993.

[Véronis 91]
J. Véronis, Error in natural language dialogue between man and machine, International Journal of Man-Machine Studies, Vol. 35, pp. 187-217, 1991.

[Wing 87]
J. M. Wing, A Larch Specification of the Library Problem, Proceedings, Fourth International Workshop on Software Specification and Design, April 3-4, 1987, Monterey, California, Los Angeles, pp. 34-41, 1987.

[Wing 88]
J. M. Wing, A Study of 12 Specifications of the Library Problem, IEEE Software, Vol. 5, No. 4, pp. 66-76, July 1988.

[Wing 94]
J. M. Wing, Formal Methods, in: J. J. Marciniak (ed.), Encyclopedia of Software Engineering, Vol. 1, Wiley & Sons, New York, pp. 504-517, 1994.

[Wojcik et al. 90]
R. H. Wojcik, J. E. Hoard, K. C. Holzhauser, The Boeing Simplified English Checker, Proc. Internatl. Conf. Human Machine Interaction and Artificial Intelligence in Aeronautics and Space, Centre d'Etude et de Recherche de Toulouse, pp. 43-57, 1990.

[Wojcik et al. 93]
R. H. Wojcik, P. Harrison, J. Bremer, Using Bracketed Parses to Evaluate a Grammar Checking Application, 31st Annual Meeting of the Association for Computational Linguistics, ACL, Columbus, Ohio, pp. 38-45, 1993.

[Wojcik & Hoard 96]
R. H. Wojcik, J. E. Hoard, Controlled Languages in Industry, in: R. A. Cole, J. Mariani, H. Uszkoreit, A. Zaenen, V. Zue, Survey of the State of the Art in Human Language Technology. Available over the Internet from <http://www.cse.ogi.edu/CSLU/HLTsurvey/accesswatch/index.html>, 18.6.1996.

[Wojcik & Holmback 96]
R. H. Wojcik, H. Holmback, Getting a Controlled Language Off the Ground at Boeing, Proceedings of the First International Workshop on Controlled Language Applications, CLAW 96, University of Leuven, Belgium, pp. 22-31, March 1996.

[Zoltan-Ford 91]
E. Zoltan-Ford, How to get people to say and type what computers can understand, International Journal of Man-Machine Studies, Vol. 34, pp. 527-547, 1991.

Index

K

L

M

N

O

P

Q

R

Lebenslauf

Ich bin am 28. Juli 1959 in Zürich geboren worden. Die Primarschule (1966-1972) und die Sekundarschule (1972-1975) besuchte ich in Zürich. Anschliessend an meine Schulzeit absolvierte ich eine Lehre als kaufmännischer Angestellter (1975-1978) bei der Firma Angst + Pfister AG in Zürich. Nach dem Lehrabschluss arbeitete ich bei dieser Firma drei weitere Jahre (1978-1981) als käufmännischer Sachbearbeiter in der Verkaufsabteilung *Kunststofftechnik*. Während dieser Zeit fasste ich den Entschluss, die Matura auf dem zweiten Bildungsweg nachzuholen. Ich besuchte zu diesem Zweck die Kantonale Maturitätsschule für Erwachsene in Zürich (1980-1983), die ich mit der Matura, Typus B, abschloss. In den folgenden zehn Jahren (1983-1993) arbeitete ich bei der Schweizerischen Bibliothek für Blinde und Sehbehinderte in Zürich in der Blindenschriftproduktion. Gleichzeitig studierte ich an der Universität Zürich Deutsche Sprach- und Literaturwissenschaft, Informatik und Psychologie und schloss mein Studium (1993) mit der Lizentiatsarbeit *PYGMALION - Grundlagen zu einem logikbasierten Dialogsystem* ab, die von Prof. Dr. Michael Hess betreut und von Prof. Dr. Harald Burger begutachtet wurde. Seit dem Abschluss meines Studiums arbeitete ich zuerst als Assistent und dann als Doktorand in der Gruppe *Requirements Engineering* von Prof. Dr. Martin Glinz am Institut für Informatik der Universität Zürich. Die vorliegende Dissertation habe ich im Rahmen des Nationalfondsprojekts *Attempto – Controlled English for Requirements Specifications* ausgearbeitet, das von Dr. Norbert E. Fuchs geleitet wird. Während der Zeit am Institut für Informatik habe ich die Vorlesungen *Natürlichsprachliche Benutzerschnittstellen*, *Deklarative Spezifikationen von Software*, *Logische Programmierung* und das Praktikum *Logische Programmierung* von Dr. Norbert E. Fuchs betreut sowie den gemeinsamen Vorlesungszyklus *Grundlagen der Softwaretechnik für wissensbasierte Systeme* von Prof. Dr. Rolf Peifer und Dr. Norbert E. Fuchs.