Diplomarbeit

ROPLEX:
Natürlichsprachliche Beschreibung von generischen Roboterplandaten

Patrick Gebhard
page@cs.uni-sb.de

Lehrstuhl Prof. Dr. W. Wahlster
Fachbereich 14 – Informatik
Universität des Saarlandes
D-66123 Saarbrücken

Nach einem Thema von
Professor Dr. W. Wahlster

Februar 1999

Zusammenfassung

Die vorliegende Arbeit beschreibt eine Erklärungskomponente einer natürlichsprachlichen Schnittstelle, die in Robotersystemen für die Kommunikation mit Menschen eingesetzt werden kann. Die speziell dafür benötigten Bereiche des Roboterwissens bestehen einerseits aus Plänen, in denen Roboteranweisungen formuliert sind, und andererseits aus Diagnose- sowie Umweltinformationen.

Die für dieses Dialogmodul entwickelten Methoden zur Erfassung und Weiterverarbeitung von Daten erlauben einen vielseitigen Einsatz. Die textuellen Erklärungen von Roboter-Anweisungsplänen, Lage von Umweltobjekten und auch von Fehlern können in verschiedenen natürlichen Sprachen erzeugt werden. Zusätzlich ist eine natürlichsprachliche Konfigurationsschnittstelle erstellt worden, die es dem Menschen erlaubt, den Inhalt der Erklärungen mittels geschriebener und gesprochener Sprache zu manipulieren.

Bei der Realisierung dieser Dialogeinheit lag der Schwerpunkt auf der Entwicklung eines flexiblen Analyse- und Auswertungsverfahrens für die vom Roboter bereitgestellten Informationen und in der Konzeption einer sprachunabhängigen Zwischenrepräsentation.

Inhaltsverzeichnis

1 Einleitung 9

1.1 Stand der Forschung 10
1.1.1 Die FLAKEY Experimentalumgebung 10
1.1.2 Natürlichsprachliche Kommunikation mit dem Roboter TJ 11
1.1.3 Natürlichsprachliche Schnittstelle zu einem mobilen Roboter 12
1.1.4 Generierung koordinierter multimedialer Erklärungen 14
1.1.5 Das Expertensystem MYCIN 16

1.2 Einordnung in das Projekt REAL 16

1.3 Beschreibung der Arbeit 18
1.3.1 Vorgeschichte 18
1.3.2 Gliederung 18

2 Grundlagen 21

2.1 Die Dialogschnittstelle RONTRA 21
2.1.1 Komponenten 24
2.1.2 Wissensbasen 25
2.1.3 Funktionalität 26
2.1.4 Realisierung 27

2.2 Objektlokalisation durch BOLA 28
2.2.1 Eine Anytime-Architektur zur Beantwortung von Lokalisationsfragen 28
2.2.2 Arbeitsweise 29

2.3 Eingabeanalyse 30

2.4 Eine Anwendungsdomäne 32
2.4.1 Der autonome Roboter KAMRO 32
2.4.2 Eigenschaften von KAMRO 34
2.4.2.1 Szenariodaten 34
2.4.2.2 Plansyntax 36
2.4.2.3 Syntax und Semantik von Roboterbefehlen in Plänen 36
2.4.2.4 Statusinformationen 41
2.4.3 KANTRA-1 als Prototyp einer Dialogschnittstelle 42

3 Problembeschreibung und Lösungsansatz 45

3.1 Problembeschreibung 45

3.2 Anforderungen an den Erklärungsprozeß 46
3.2.1 Was soll erklärt werden? 46
3.2.2 Wie soll erklärt werden? 47

3.3 Ein erster Lösungsansatz 49
3.3.1 Vom Planschritt zur natürlichsprachlichen Erklärung 49

3.3.2 Beispiel einer Planschritterklärung 52
3.3.3 Diskussion 53

4 Realisierung und Implementation 57

4.1 Schnittstelle zu Robotern 59
4.1.1 Szenariodaten 59
4.1.2 Plan- und Roboteranweisungssyntax 60
4.1.3 Statusinformationen 63

4.2 Benutzerschnittstelle 64
4.2.1 Syntax der Tätigkeitsbeschreibung 64
4.2.2 Syntax der Fehlerbeschreibungen 67
4.2.3 Syntax von Lokalisationsbeschreibungen 69

4.3 Datenverwaltung 70
4.3.1 Verwaltung der Szenariodaten 70
4.3.2 Verwaltung der Diagnose- und Fehlerdaten 71
4.3.3 Der Verbrahmen als flexible Satzgrundlage 74

4.4 Grundlagen des Erklärungsprozesses 78
4.4.1 Basiswissen 79
4.4.2 Konfigurationswissen 81
4.4.2.1 Das Planschrittanalyseschema 82
4.4.2.2 Das Erklärungsprofil 83

4.5 Der Erklärungsalgorithmus 86
4.5.1 Wissensauffrischung 87
4.5.2 Planschrittanalyse 88
4.5.2.1 Beispiel einer Planschrittanalyse 89
4.5.3 Erklärungssynthese 90
4.5.3.1 Robotertätigkeit 91
4.5.3.2 Beispiel einer Erklärungssynthese 92
4.5.3.3 Fehlerbeschreibung 93

4.6 Natürlichsprachliche Fragen und Befehle 94
4.6.1 Eingaben bei Fehlersituationen 96
4.6.2 Lokalisationsfragen 96
4.6.3 Konfigurationsbefehle 97

4.7 Systemimplementation und Integration 97

5 Zusammenfassung und Ausblick 101

5.1 Zusammenfassung 101

5.2 Ausblick 102

6 Anhang 105

6.1 Eingabewissen 105
6.1.1 Eingabeidentifizierer 105
6.1.2 In ROPLEX definierte Eingabeidentifizierer 107
6.1.3 Identifizierungsmasken 108

6.2 Die graphische Benutzeroberfläche RONTRAGUI 111
6.2.1 Eigenschaften und Funktionalität 111
6.2.2 Konfiguration 114

6.3 Die Anwendungsdomäne CHEW 116
6.3.1 Schnittstelle zum CHEW-Roboter 117
6.3.2 Ein Beispiellauf 119

Literaturverzeichnis 125

Abbildungsverzeichnis 129

Index 133

1 Einleitung

Der Einsatz von intelligenten Servicerobotern in Haushalt und Industrie stellt hohe Ansprüche an die Interaktion zwischen Mensch und Maschine. Verteilte Multi-Robotersysteme, in denen Roboter unabhängig voneinander arbeiten, zeigen heute schon den Weg zu intelligenten Dienstleistungs- und Produktionsverfahren auf. Je mehr ein Roboter auf sich alleine gestellt ist, das heißt, autonom arbeiten muß, desto einfacher muß die Zusammenarbeit zwischen Mensch und Roboter funktionieren, um eventuelle Ausnahmesituationen besser verstehen und bewältigen zu können. In der heutigen Zeit ist eine zufriedenstellende Zusammenarbeit von Mensch und Roboter aufgrund ihrer nicht standardisierten Schnittstellen eher schwierig. Die Kommunikation mit Robotern basiert in den meisten Fällen auf einer speziell angepaßten Sprache. Sie verursacht bei Laien auf dem Gebiet der Robotik Verständnisprobleme und demzufolge eine eher abgeneigte Haltung. Es ist nicht unbedingt gewährleistet, daß die Handhabung von Robotern prinzipiell schnell und unkompliziert vonstatten geht.

Natürliche Sprache bietet sich als Kommunikationsmedium an, da es mit ihr auch für nicht speziell ausgebildete Menschen kulturbedingt einfach ist, ein technisches System anzusteuern. Eine weitere Erleichterung bei der Interaktion zwischen Mensch und Roboter ist die Fähigkeit des Roboters, autonom oder auf Anfrage Arbeitsvorgänge und Ausnahmesituationen in natürlicher Sprache dem Menschen zu beschreiben. Diese Eigenschaften ermöglichen dem Menschen in jeder Situation eine unkompliziertere und menschengerechtere Kommunikation.

Ziel dieser Arbeit ist es, eine Schnittstelle zwischen Roboter und Mensch zu realisieren, die einerseits den Roboter befähigt, dem Menschen eine natürlichsprachliche Beschreibung seiner Handlungen zu geben, und andererseits dem Menschen die Möglichkeit bietet, sich in natürlicher Sprache vom Roboter über spezifische Details der Roboterwelt berichten zu lassen. Der Schwerpunkt dieser Arbeit liegt in dem Versuch, die Schnittstelle zur Ansteuerung unterschiedlicher Roboter möglichst flexibel zu gestalten, und insbesondere die in der jeweiligen Roboterbefehlssprache formulierten Anweisungspläne und Umweltdaten in verschiedenen natürlichen Sprachen dem Menschen beschreiben zu können.

1.1 Stand der Forschung

Für die Entwicklung der umweltdaten- und planbasierten Erklärungskomponente ROPLEX[1] bedarf es Untersuchungen in dem Gebiet des natürlichsprachlichen Zugangs zu Robotern. Eines der bekanntesten Systeme ist SHAKEY [Nilson 84] beziehungsweise sein Nachfolger FLAKEY mit der Systemarchitektur SAPHIRA [Konolige et al. 96], die unter anderem anschließend beschrieben werden. Ein ähnliches System ist TJ [Torrance 94]. Darüber hinaus werden bereits existierende Erklärungskomponenten, insbesondere die Systeme ROBOTIC AID [Crangle & Suppes 95], COMET [Feiner & Keown 90] sowie die Erklärungskomponente des Expertensystems MYCIN [Buchanan & Shortliffe 84] genauer erörtert.

1.1.1 Die FLAKEY Experimentalumgebung

FLAKEY ist ein mobiler Roboter, dessen Einsatzgebiet eine Bürowelt ist, worin er unter anderem Transportaufgaben erfüllen muß. Ihm können einfache Befehle in natürlicher Sprache über ein Mikrophon erteilt werden. Sie werden durch sein integriertes Sensor-, Steuer- und Planungssystem SAPHIRA zuerst in Zielzustände umgewandelt, die als Grundlage für einen situationsabhängigen Ablaufplan dienen. Dieser wird dann sequentiell abgearbeitet. Ändern sich während der Ausführung die zu erreichenden Zielzustände, so wird ein neuer, der Situation angepaßter Ablaufplan erstellt.

Der Plangenerierung liegen einerseits die Objektinformationen des aktuell durch seine Kameras sichtbaren Bereichs vor, und andererseits Angaben, wo er in der Bürowelt Personen finden kann. Diese Lokationen werden ihm von einer Leitperson gezeigt.

Der für diese Arbeit interessante Aspekt von FLAKEY ist die Art und Weise, wie Menschen mit ihm kommunizieren können. Prinzipiell ist er in der Lage, einfache Befehle in natürlicher Sprache[2] zu verstehen. Ein einfacher Befehl um Daten aus einem Büro zu holen, ist beispielsweise: „`Get the file from Karen`“. Zudem kann er einfache Gesten, wie das Zeigen nach links oder rechts verstehen. Diese werden dazu benutzt, sprachliche Informationen zu ergänzen, wie zum Beispiel: „`This is Karen´s Office.`“. Der Roboter kann mit einfachen Befehlen dazu gebracht werden, den Sprecher zu finden, oder ihm zu folgen. Hat er ihn gefunden, gibt er als Feedback-Meldung aus: “`Here I am!`“. Findet er den Sprecher nach dem vorgesehenen Suchmuster nicht, meldet er: „`I can‘t find you!`“.

Die Hauptaufgabe von FLAKEY sind Transportdienste, wie das Überbringen von Nachrichten und Manuskripten. Findet er die Person nicht, der er eine Nachricht übergeben soll, generiert das Planungssystem einen neuen Ablaufplan, der ihn in ein anderes Büro führt, wo er die Möglichkeit hat, den Aufenthalt seiner Zielperson zu erfragen. Damit ist er in der Lage der Zielperson die Nachricht zu überbringen.

[1] **Ro**bot **pl**an **ex**planation

[2] Die Befehle sind in englischer Sprache formuliert.

1.1.2 Natürlichsprachliche Kommunikation mit dem Roboter TJ

Torrance beschreibt in seiner Arbeit eine natürlichsprachliche Dialogschnittstelle für den Roboter TJ. Der sich ähnlich wie FLAKEY autonom in einer Bürowelt bewegen kann. Er wird mit Kommandos, die ihm in englischer Sprache mittels der Tastatur erteilt werden, gesteuert. Sie dienen unter anderem dazu, ihn durch den Bürokomplex zu bewegen. Auf diese Art kann er ein Orts- und Wegegedächtnis aufbauen, das nachträglich natürlichsprachlich manipuliert werden kann.

Für die Bewegung in der Bürowelt steht dem Roboter ein Plan, der aus Teilplänen besteht, zur Verfügung. Teilpläne bestehen aus atomaren Bewegungsinformationen und Ortsbeschreibungen. Die Pläne zur Wegberechnung werden mit dem bekannten Kürzesten-Weg-Algorithmus von Dijkstra berechnet. Nach jeder Abarbeitung eines Teilplans wird ein neuer Plan erzeugt, der aus neuen Teilplänen für den verbleibenden Restweg besteht.

Bei der Spracherkennung wurden neue Wege beschritten. Für die traditionelle Art der Sprachverarbeitung[3] sah Torrance kein Anwendungspotential, da der Roboter nur wenige Kommandos verstehen kann. Er beauftragte eine Gruppe von zwei Testpersonen, sich einen Dialog mit TJ vorzustellen. Dabei sollten sie sich auf Dialoge beschränken, die aus Anweisungen, Mitteilungen und Fragen an den Roboter bestehen. Die Erkenntnisse daraus führten dazu, daß er bei der Spracherkennung ein Pattern-Matching-Verfahren einsetzte. Damit wurde insbesondere erreicht, daß die Stellung der Orte und Wege im Satz eindeutig definiert ist und sie deshalb leicht erkannt werden. Für die eigentliche Wiedererkennung von Orts- und Wegebezeichungen steht ein Lexikon zur Verfügung, in dem neu gelernte Orte und Wege gespeichert werden.

Wie eben angedeutet, kann zwischen drei verschiedenen Eingabemustern unterschieden werden:

- Anweisungen werden in dem Augenblick ausgeführt, in dem sie erteilt worden sind. Dies hat zum Nachteil, daß eine noch nicht abgeschlossene Ausführung einer Anweisung direkt abgebrochen wird. Um dies zu verhindern, können Anweisungen in die Schlüsselsequenz **... Then ...** [**Next ...**] **Finally ...** eingebettet werden:

  <Anweisung1>

  Then <Anweisung2>

  Finally <Anweisung3>

- Mitteilungen dienen dem Roboter zur Orientierung und zum Lernen von Wissen über räumliche Zusammenhänge.

- Fragen an den Roboter können sich auf die Büroumgebung beziehen. Es ist beispielsweise möglich, TJ nach der Richtung zu befragen, in die er schaut, oder den

[3] Traditionelle Sprachverarbeitungssysteme gliedern das Problem des Sprachverstehens in mehrere Schritte. Sie analysieren zunächst syntaktisch und danach semantisch die Eingabe. Dieses Verfahren neigt zu Mehrdeutigkeiten, die durch Heuristiken minimiert werden müssen, um eindeutige Ergebnisse zu erlangen.

Weg zwischen zwei ihm bekannten Orten zu beschreiben. Bei der Frage nach einer Wegbeschreibung wird zuerst durch das Planungssystem ein Plan generiert. Daran anschließend werden die enthaltenen Teilpläne verbalisiert. Beispiel:

Frage: `How do i get from MARK´S OFFICE to THE YELLOW TRASHCAN?`

Antwort:
```
To get from MARK´S OFFICE to THE YELLOW TRASHCAN,
You go north to ANITA´S OFFICE.
You go north to LYNNE´S OFFICE.
You go north to ROBERT´S OFFICE.
You go north to IAN´S OFFICE.
Then you go west to THE YELLOW TRASHCAN.
```

1.1.3 Natürlichsprachliche Schnittstelle zu einem mobilen Roboter

Im Buch *Language and Learning for Robots* von [Crangle & Suppes 95] wird unter anderem die natürlichsprachliche Ansteuerung des mobilen Manipulationssystems ROBOTIC AID vorgestellt. Es wurde entwickelt, um Menschen bei deren Aktivitäten in unstrukturierten Umgebungen, wie sie zum Beispiel in Wohnungen zu finden sind, zu unterstützen. Es besteht aus einem frei beweglichen Manipulator, der auch bei KAMRO zum Einsatz kommt, der auf einer omnidirektionalen mobilen Basiseinheit montiert ist. Zudem existiert eine stationäre Steuerkonsole. Dieses System wurde in einer Krankenhausumgebung als sprachgesteuerter Telemanipulator eingesetzt. Dazu können einfache, aus einem Wort bestehende Bewegungsbefehle in englischer Sprache benutzt werden.

Im allgemeinen Fall wird davon ausgegangen, daß Befehle an den Roboter über die Tastatur eingeben werden. Die syntaktische und semantische Analyse erzeugt daraus zunächst semantische Bäume. Die dabei verwendete Grammatik ist eine Phasenstrukturgrammatik, die aus Produktionsregeln mit Attribut-Wert-Annotationen, Lexikoneinträgen und semantischen Funktionen besteht. Die syntaktische Struktur der Eingabe ist in den Produktionsregeln codiert. In den Attribut-Wert-Annotationen sind zusätzliche grammatische Restriktionen untergebracht. [4] Die Lexikoneinträge bestehen aus partiellen Roboterplänen. Die semantischen Funktionen enthalten Regeln, wie sich Bedeutungen von übergeordneten Knoten der semantischen Bäume ableiten lassen. Zudem sind hier Wahrnehmungs- und Kognitionsfunktionen enthalten. Im Anschluß daran werden die semantischen Bäume unter Zuhilfenahme des Domänen- und Diskurswissens in Roboterbefehle übersetzt. In der Wissensbasis sind Informationen über die Umwelt, wie der Roboter sie wahrnimmt, enthalten.

Die Interpretation von Roboterbefehlen ist abhängig vom Kontext des Diskurses und vom dem des Raumes. Dazu gehören beispielsweise Informationen über die aktuelle Benutzerposition, Regeln wie Adjektive, wie zum Beispiel „groß“ interpretiert werden, welches Objekt sich gerade im Zentrum des Geschehens befindet, welche Abarbeitungsreihenfolge temporale Konjunktionen nach sich ziehen, und so weiter. Daraus

[4] Zum Beispiel wenn die Merkmalswerte aus verschiedenen Phrasen (mit agreement bezeichnet) übereinstimmen müssen.

wird ein interpretierter Befehl, das *High-Level-Roboterprogramm*, generiert. Darin sind folgende Informationen enthalten:

- Die benötigten Bewegungsroutinen.
- Die zeitliche Ordnung, in der die Bewegungsroutinen ausgeführt werden sollen und die logischen Bedingungen, unter denen sie aktiviert werden beziehungsweise terminieren sollen.
- Die Anfragen an das Weltmodell, die als Ergebnis Flächen oder Räume liefern, die räumlichen Ausdrücken in der natürlichen Sprache entsprechen.

Die noch nicht bestimmten Variablen des Roboterprogramms bekommen durch die kontextadaptiven Mechanismen konkrete Werte. Insbesondere sind dies zwei- (bei der Roboterbewegung) oder dreidimensionale (bei der Armbewegung) Koordinatenangaben, die von der Position des Benutzers, des Roboters, seines Armes und anderen Objekten abhängig sind.

Die kognitiven, perzeptiven und motorischen Fähigkeiten des Roboters lassen sich in vier Klassen einteilten:

- *Bewegung im Raum*: Prinzipiell sind dies zielorientierte Prozeduren, die die Bewegung ändern. Beispiele dafür sind: Stop, Pause, Weiterfahren, Fahren in eine bestimmte Richtung, Bewegung zu einem Objekt.
- *Testroutinen*: Über interne fest implementierte Routinen kann der Roboter überprüfen, ob er sich in einer bestimmten Region aufhält, ob er in eine bestimmte Richtung fährt, beziehungsweise aus einer bestimmten Richtung kommt. Darüber hinaus gibt er Auskunft über die zurückgelegte Zeit und Distanz sowie Drehung, die er während der letzten Aktion getätigt hat. Die Anworten beschränken sich auf „Ja“ oder „Nein“. Diese Testroutinen können auf zweierlei Art benutzt werden. Im `If`-Modus werden sie jeweils mit „Ja“ oder „Nein“ beantwortet. Damit kann der Roboterstatus überprüft werden. Der `When`-Modus dient dazu, erst dann eine an den Test gebundene Aktion durchzuführen, wenn dieser das Ergebnis „Ja“ zurückliefert.
- *Ortsbestimmungen*: Der Roboter besitzt zur Eigenbewegung ein zweidimensionales und zur Armbewegung ein dreidimensionales Weltmodell. Die Umweltobjekte sind durch 5er-Tupel repräsentiert, die aus folgenden Komponenten bestehen: Der Name des Objektes in der Realität, eine Maßzahl, die den Abstand zum Roboter ausdrückt, eine Himmelsrichtung, ein boolscher Wert, der beschreibt, ob sich das Objekt innerhalb oder außerhalb der Abstandsmaßzahl befindet, und eine Spezifikation des Koordinatensystems.
- *Kontrollstrukturen*: Sie repräsentieren logische und temporale Zusammenhänge in einem natürlichsprachlichen Befehl. Damit wird es dem Roboter möglich, neue Aktionsfolgen zu lernen. Diese können rekursiv verknüpft werden, um komplexe Aktionsabläufe zu beschreiben.

1.1.4 Generierung koordinierter multimedialer Erklärungen

Das System COMET[5] wurde entwickelt, um koordinierte multimediale Erklärungen für Wartungs- und Reparaturarbeiten von komplexen Systemen zu erzeugen. Das Referenzsystem ist eine tragbare Funkstation der amerikanischen Armee.

Für die Wartung und Reparatur größerer Systeme wird heutzutage meist eine umfangreiche Dokumentation in Form von Büchern oder Manuskripten benötigt. Ein Nachteil davon ist, daß die Dokumentation nicht automatisch auf individuelle Reparatursituationen eingestellt werden kann. Der entscheidende Vorteil des Systems COMET ist, daß es dynamisch Inhalt und Form von Erklärungen erzeugen kann. Dies wird vor allem durch die Variation der Form erreicht, das heißt welchen Inhalt Text und Graphik haben und wie sie präsentiert werden. Das System ist aus einem Inhaltsplaner, einem Mediengenerator für Text und Graphik, einem Koordinator, einem Layoutmanager sowie verschiedenen Wissensquellen aufgebaut, siehe Abbildung 1.

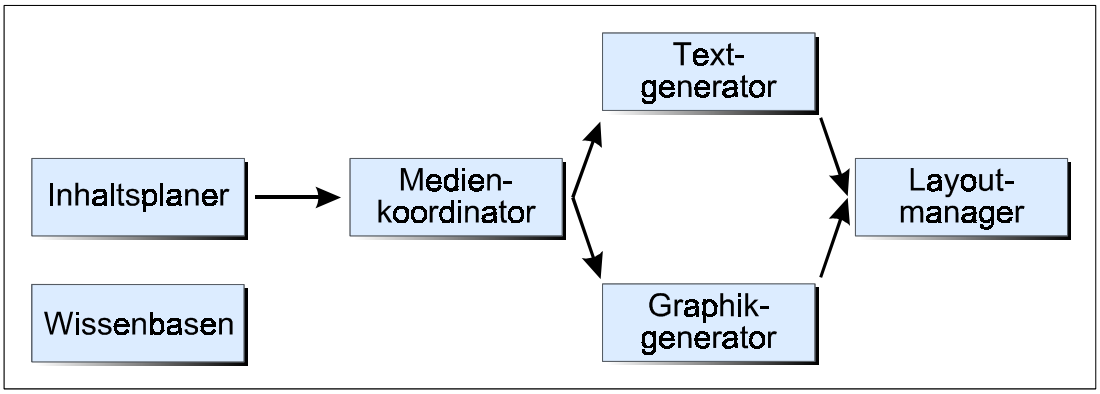

Abbildung 1: Arbeitsweise von COMET.

Das für die Erklärungen zugrundeliegende Wissen ist in zwei Klassen aufgeteilt. Statisches Wissen besteht zum einen aus der Objekthierarchie, die aus den Komponenten der Funkstation und deren jeweiligen Merkmalen aufgebaut ist, zum anderen aus den Reparatur- und Wartungsmethoden. Eine Planausführungskomponente, ein Regelsatz für Diagnosen von Gerätefehlern sowie ein System, das neue Diagnoseregeln aus vergangenen Fehlern erlernen kann, sind im dynamischen Wissen zusammengefaßt.

Bei einer Systemanfrage wird der Inhalt der Erklärung durch den Inhaltsplaner dynamisch generiert. Für die beschriebene Wartungsdomäne benutzt COMET zwei Planungsschemata. Durch sie können Fehler- und Objektbeschreibungen erzeugt werden. Für neue Domänen kann eine Bibliothek aus hierarchischen Schemata benutzt werden. Darin unterscheidet man domänenbezogene und allgemeingültige Schemata. Das Ergebnis der Inhaltsplanung ist eine Hierarchie aus logischen Formeln (LF´s), die den Inhalt repräsentieren. Der Medienkoordinator bekommt diese als Eingabe und typisiert den Teil der Erklärung, der als Text beziehungsweise als Graphik erscheinen soll. Im folgenden wird beschrieben, welcher Informationstyp für die textuelle oder graphische Erklärung in Frage kommt:

[5] **CO**ordinated **M**ultimedia **E**xplanation **T**estbed

Ortsinformationen	graphische Erklärung
Physikalische Attribute	graphische Erklärung
Einfache Aktionen	textuelle und graphische Erklärung
Zusammengesetzte Aktionen	textuelle und graphische Erklärung
Abhängigkeiten	textuelle Erklärung für Bindewörter, textuelle und graphische Erklärung für Aktionen
Abstrakte Aktionen	textuelle Erklärung

Aus diesen typisierten LF's werden erklärende Bilder und Text erzeugt. Für diesen Zweck existiert der Graphik- beziehungsweise Textgenerator. Dieser erzeugt Erklärungen in englischer Sprache. Das Hauptproblem dabei ist die Auswahl von semantischen Primitiven und die Konstruktion einer syntaktischen Struktur, die als Grundlage der Erklärung dient. Die Probleme, die dabei auftreten, lassen sich in die folgenden drei Gebiete aufteilen:

1. Implementation eines funktionalen Unifikationsformalismus´ und eines Textgenerators.
2. Auswahl von Bindewörtern.
3. Identifikation und Repräsentation der inhaltlichen Kombinierbarkeit sprachlicher Einheiten (Beispiel: Biene + summen, aber nicht Haus + dick).

Die Wahl, funktionale Unifikation einzusetzen, beruht auf der Tatsache, daß die Sprachgenerierung eng an die Wechselbeziehung von sprachlichen Zwängen gebunden ist. Damit ist COMET in der Lage, eine Erklärung auf die Anfrage[6]: „How do I load the transmission frequency into channel 1?“ zu erklären:

Set the channel knob to position 1.
Set the MODE knob to SC.
Set the FCTN knob to LD.
Now, enter the frequency:
First, press the FREQ button.
This will cause the display to show an arbitray number.
Next, press the CLR button in order to clear the display.
Next, enter the new frequency using the number buttons.
Next, record this number in order to check it later.
Finally, press Sto ENT.
This will cause the display to blink.

Neben der textuellen Erklärung erzeugt der Graphikgenerator passende Bilder. Beide Formen der Erklärung werden durch den Layoutmanager zusammengeführt und dem Benutzer präsentiert.

[6] Das Beispiel bezieht sich auf die oben im Abschnitt vorgestellte portable Funkstation.

1.1.5 Das Expertensystem MYCIN

MYCIN [Buchanan & Shortliffe 84] ist ein Expertensystem [Duda & Shortliffe 83], das entwickelt wurde, um bestimmte Infektionskrankheiten zu diagnostizieren und dazu eine entsprechende Therapie (mit Antibiotika) anzubieten. Zudem vermag es seine Argumentation im Detail zu erklären. Die Erklärungen werden von der Erklärungskomponente TEIRESIAS erzeugt, die in ([Buchanan & Shortliffe 84], Seite 331 f.) beschrieben wird. Ihre Entwicklung begann Anfang der Siebziger Jahre. Innerhalb der nächsten drei Jahre wurden Eigenschaften hinzugefügt und die allgemeine Funktionalität verbessert. Der Benutzer konnte Erklärungen auf Anfragen ([Buchanan & Shortliffe 84], Seite 349 f.) oder zum Faktenwissen ([Buchanan & Shortliffe 84], Seite 355 f.) bekommen.

Für die Erklärung von Anfragen wurde unter Verwendung spezieller Regeln ein Zielbaum aufgebaut. Mit ihm war es möglich, zu erklären warum eine bestimmte Entscheidung getroffen wurde und wie sie zustande kam. Für die Erzeugung der Erklärung wurde der Baum je nach Fragetyp in einer anderen Richtung durchlaufen. Handelte es sich um „Warum"-Fragen, dann wurde rückwärts entlang der Inferenzkette gesucht, bis eine passende Antwort gefunden wurde. Für jede folgende – sogenannte *metakommunikative* – „Warum"-Frage wurde weiter in dieser Richtung gesucht. Für die Erklärungen von „Wie"-Fragen wurde in die andere Richtung nach Anworten gesucht. In diesem Fall ist nach Teilzielen des aktuellen Ziels gesucht worden.

Stellte der Benutzer eine Frage zu einem Wert eines aktuellen Parameters, so wurde das Faktenwissen konsultiert. Die vom System aufgestellte Hypothese diente in diesem Fall als Antwort. War es notwendig, konnte MYCIN den Benutzer nach Werten von Parametern befragen. Zur Analyse und Beantwortung von Fragen wurden sogenannte *Templates* ([Buchanan & Shortliffe 84], Seite 354 f.) eingesetzt.

1.2 Einordnung in das Projekt REAL

Im Projekt REAL[7] – Interaktion von Objektlokalisation und Sprachproduktion – des Sonderforschungsbereichs 378 – Ressourcenadaptive kognitive Prozesse – an der Universität des Saarlandes wird in der ersten Förderungsperiode anhand von Beispielen bei der Generierung von sprachlichen Raumbeschreibungen in Dialogsituationen die ressourcenbeschränkte Objektlokalisation untersucht. Die damit verbundenen Realisierungen werden in Abbildung 2 präsentiert.

Der MOSES-Agent ist die Implementierung eines formalen Modells zur Generierung inkrementeller multimedialer Wegbeschreibungen unter Berücksichtigung von zeitlichen Ressourcen bei der Generierung und Präsentation [Maaß 94], [Maaß et al. 95], [Maaß 96]. Er hat als Einsatzgebiet eine unbekannte stadtähnliche dreidimensionale Umgebung. Zur Orientierung benutzt er visuelle Informationen und Kartenmaterial. Er erzeugt eine natürlichsprachliche Wegbeschreibung, indem er sich durch die simulierte Umgebung bewegt.

[7] **RE**ssourcen**A**daptive **L**okalisation

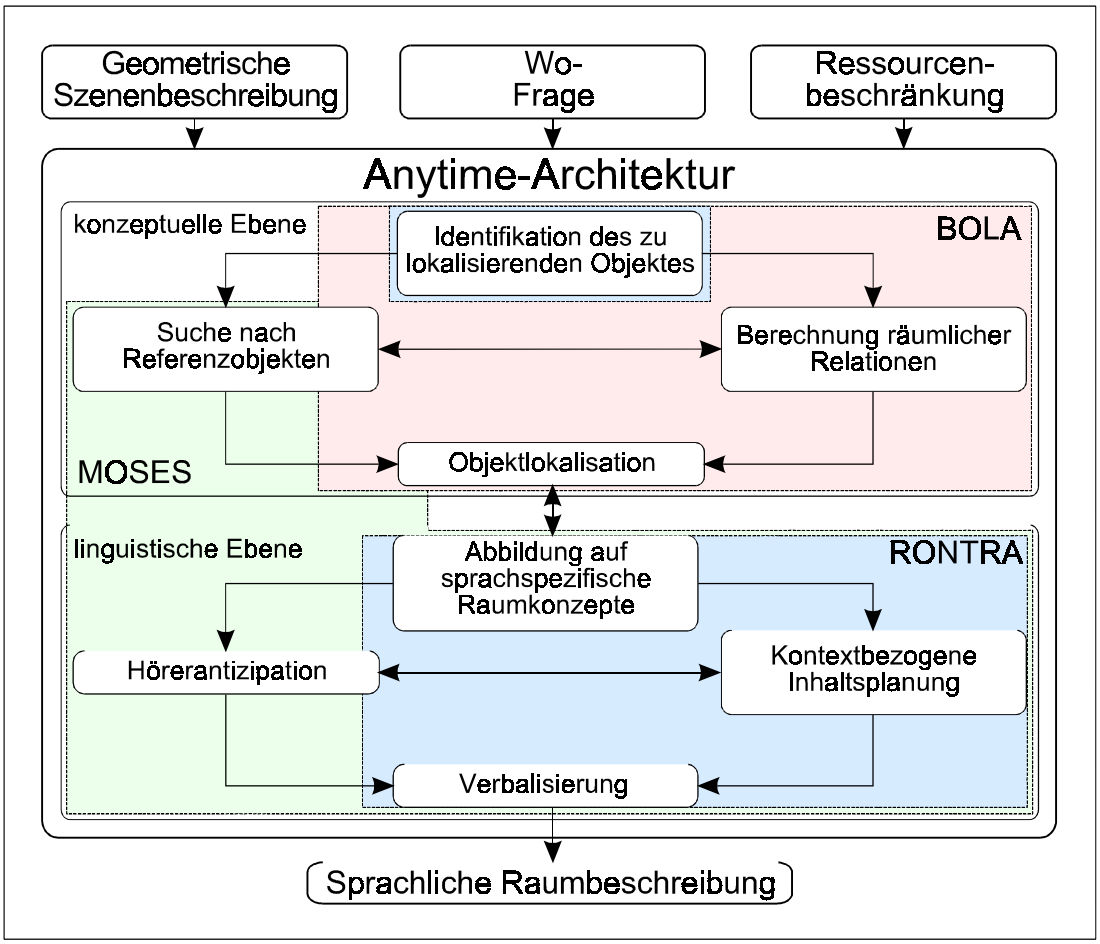

Abbildung 2: Im Projekt REAL realisierte Systeme.

Das System BOLA[8] ist die prototypische Implementierung eines algorithmischen Modells zur Beantwortung von Lokalisationsfragen (Wo-Fragen) unter Zeitdruck [Wahlster et al. 98]. Das System ist aus einer ressourcenadaptierenden Kombination von Anytime-Algorithmen aufgebaut. Die Qualität der resultierenden räumlichen Beschreibungen verbessert sich graduell mit steigender Verarbeitungszeit. Diese Realisierung erlaubt es, einem Kommunikationspartner unter variierenden Ressourcenbeschränkungen Lokalisationsfragen über seine aktuelle visuelle Umgebung in unterschiedlichen Raumszenarien zu beantworten.

RONTRA[9] ist ein System für natürlichsprachlichen Zugang zu autonomen Robotern [Laengle et al. 96], [Lüth et al. 94], [Stopp 97]. Dabei wurde insbesondere die Interpretation und Generierung räumlicher Ausdrücke betrachtet [Stopp et al. 94], [Stopp & Laengle 95], da dafür insbesondere visuell-geometrisches Wissen der Roboterumgebung mit natürlichsprachlichen Ausdrücken assoziiert werden muß [Laengle et al. 95]. Die verwendeten sprachspezifischen Raumkonzepte dienen dazu, Objekte zu identifizieren, Zielpositionen anzugeben und natürlichsprachliche Beschreibungen der räumlichen Umgebung zu liefern, wie sie beispielsweise im Zusammenhang mit Wo-Fragen erforderlich sind. Teil dieses Dialogsystems ist das Umwelt- und Planerklärungsmodul

[8] **B**eschränkt-**O**ptimaler **L**okalisations**A**gent

[9] **RO**bot's **N**atural language **TRA**nslator

ROPLEX[10], das in dieser Arbeit vorgestellt wird. Es realisiert die natürlichsprachliche Erklärung von Roboterbefehlsplänen und die Beschreibung der räumlichen Umgebung des Roboters im Zusammenhang mit Lokalisationsfragen. Der Prototyp KANTRA-1 [Gebhard & Jung 96] diente dabei als Ausgangsbasis. Er wurde innerhalb des Projektes VITRA[11] [Herzog et al. 96] des Sonderforschungsbereichs 314 – Künstliche Intelligenz und Wissensbasierte Systeme – in Zusammenarbeit mit dem **I**nstitut für **P**rozeßrechentechnik und **R**obotik (IPR) der Universität Karlsruhe erstellt.

1.3 Beschreibung der Arbeit

Dieses Kapitel dient dazu, den Aufbau und die Entstehung dieser Diplomarbeit zu beschreiben.

1.3.1 Vorgeschichte

Der Ausgangspunkt dieser Arbeit war der Prototyp KANTRA-1 einer Dialogschnittstelle. Mit ihm konnten natürlichsprachlich formulierte Anweisungen an den autonomen intelligenten Roboter KAMRO gestellt werden. Es war aber nicht möglich, die Tätigkeiten natürlichsprachlich bei Bedarf zu erklären, beziehungsweise Fragen zu Fehlern oder der Umwelt zu stellen. Basierend auf diesem Prototyp ist das Konzept der sprach- und roboterunabhängigen Dialogschnittstelle RONTRA entworfen worden. Neben der natürlichsprachlichen Ansteuerung des Roboters hat sie einerseits die Aufgabe, in verschiedenen Sprachen Erklärungen aus Anweisungsplänen und Umweltinformationen zu erzeugen, andererseits gibt sie dem Benutzer die Möglichkeit, die Ausgaben mittels natürlicher Kommandos zu konfigurieren.

1.3.2 Gliederung

Das erste Kapitel dient zur Orientierung und Einordnung in die Forschungsarbeit des aktuellen Sonderforschungsbereich 378 am Lehrstuhl Wahlster an der Universität des Saarlandes. Es werden unter anderem verwandte Arbeiten genannt und kurz vorgestellt.

Im zweiten Kapitel werden die theoretischen Grundlagen der Dialogschnittstelle RONTRA präsentiert. Zudem werden die für die Arbeitsweise von ROPLEX wichtigen Systeme BOLA und RIACS präsentiert.

Das sich anschließende dritte Kapitel widmet sich zuerst der allgemeinen Problembeschreibung. Dabei wird besonders auf die Schwierigkeiten bei der Informationsanalyse eingegangen. Anschließend stelle ich einen ersten Lösungsansatz vor, der die zuvor diskutierten Probleme verarbeitet. Das Kapitel endet mit einer Diskussion über die Vor- und Nachteile der vorgestellten Planerklärungsmethode.

Im vierten Kapitel wird die Realisierung vorgestellt. Zuerst werden die Schnittstellen zum Roboter und zum Menschen präsentiert. Daran anschließend werden die benötigten Dateneinheiten und die daraus aufgebauten Wissensbasen und Methoden vorge-

[10] **R**obot **PL**an **EX**planation

[11] **VI**sual **TRA**nslator

stellt, mit denen eine multilinguale Translation von Anweisungsplänen in natürliche Sprache möglich wird. Im weiteren Verlauf finden sich die Beschreibungen der Hauptalgorithmen, um aus Anweisungsplänen natürlichsprachliche Tätigkeitsbeschreibungen zu erzeugen.

Das fünfte Kapitel gibt eine Zusammenfassung der Eigenschaften von ROPLEX. Daneben wird die vorliegende Arbeit kritisch beleuchtet, und es werden die Möglichkeiten der Erweiterung vorgestellt.

Der Anhang zeigt konkrete Inhalte von Wissensbasen, die ROPLEX braucht, um natürlichsprachliche Eingaben semantisch interpretieren zu können. Zudem wird anhand eines Beispiellaufes die Anwendungsdomänen Chew präsentiert. Darüber hinaus wird die graphischen Benutzeroberfläche RONTRAGUI vorgestellt. Es werden ihre Eigenschaften, die Funktionalität und die Konfigurationsmöglichkeiten erklärt.

2 Grundlagen

2.1 Die Dialogschnittstelle RONTRA

Mit der roboterunabhängigen Dialogschnittstelle RONTRA [Stopp 97] ist es möglich, Aktionen des Roboters und Objekte in der Roboterwelt in natürlicher Sprache dem Menschen zu beschreiben. Eine weitere Eigenschaft ist die Fähigkeit, natürlichsprachliche Befehle in unterschiedliche Robotersteuersprachen zu übersetzen. Dadurch erreicht man eine große Flexibilität hinsichtlich der Interaktion mit verschiedenen Robotern. Die natürlichsprachlichen Befehle können aufgrund der Benutzung eines konfigurierbaren Chart-Parsers[12] in verschiedenen Sprachen formuliert werden. Das theoretische Modell von RONTRA, in dem die einzelnen Funktionen, die benötigten Wissensbasen und der Datenfluß modelliert sind, wird in Abbildung 3 präsentiert. Darüber hinaus schlägt dieses System einen Lösungsansatz für natürlichsprachliche Zugangssysteme zu autonomen Robotern vor. Wie in [Stopp 97] beschrieben, werden dabei folgende Kategorien betrachtet:

- *Sprachliche Fähigkeiten*:
  1. Erkennung von Alltagssprache
  2. Dialogfähigkeit
  3. Berücksichtigung von Temporaladverbien
  4. Auflösung komplexer Referenzausdrücke
  5. Interpretation komplexer räumlicher Ausdrücke
- *Kognitive Fähigkeiten*:
  1. Assoziation räumlicher Ausdrücke mit räumlichen Relationen
  2. Verwendung experimentell untermauerter und damit kognitiv angenäherter räumlicher Relationen
  3. Bidirektionale Verwendung räumlicher Relationen
  4. Ressourcenadaptierende Verarbeitung räumlicher Relationen
- *Modellierungsmerkmale*:
  1. Modularität
  2. Portierbarkeit
  3. Multilingualität
  4. Multimodalität
  5. Anytime-Verarbeitung

[12] Zum Beispiel das System SB-PATR, das eine Version von D-PATR ist [Karttunen 86].

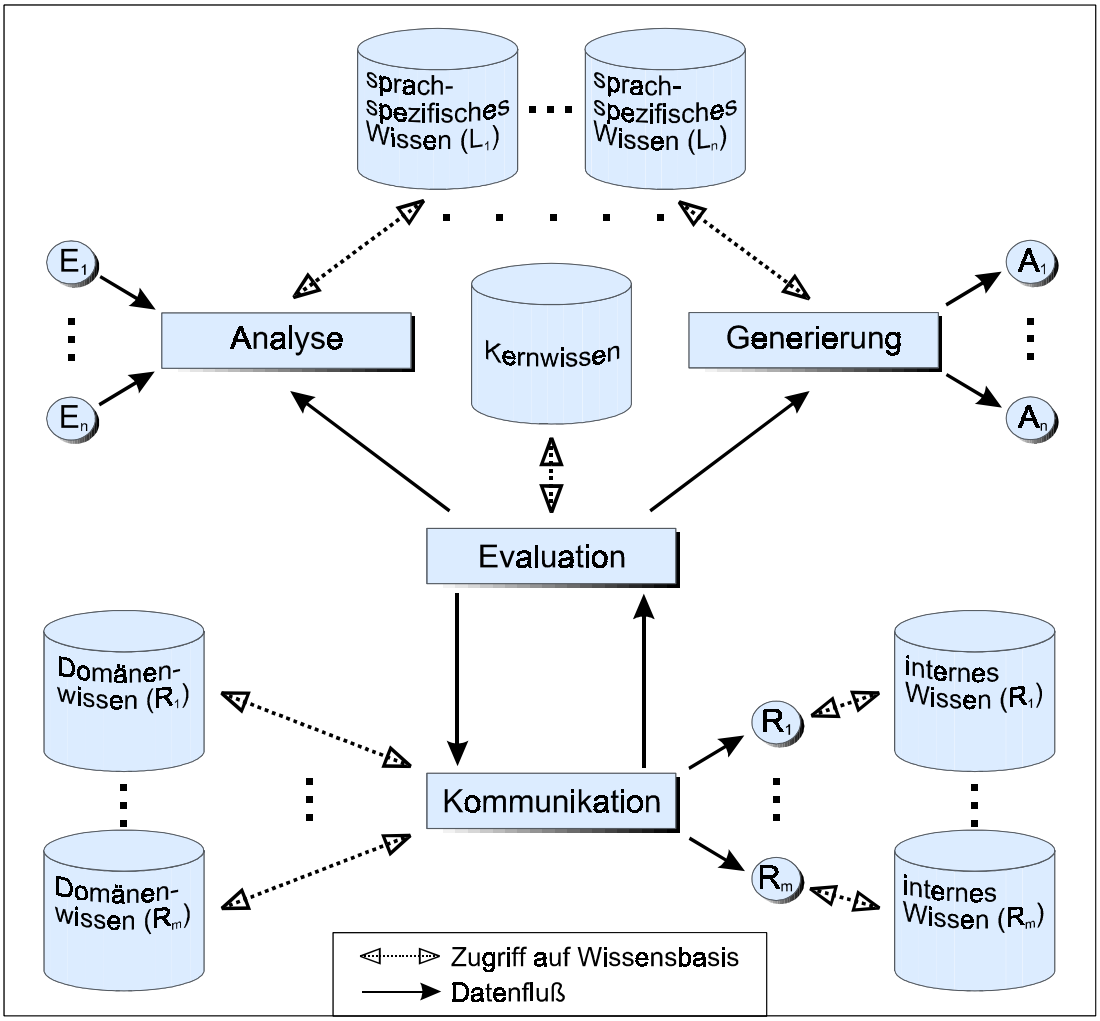

Abbildung 3: Modell der Dialogschnittstelle RONTRA.

Um mit dem Roboter zu kommunizieren, genügt es nicht, natürliche Sprache nur für Befehle zu verwenden. Vielmehr muß der Roboter die Fähigkeit besitzen, seinen Zustand zu beschreiben, seine Aktionen zu erklären und Rückfragen zu stellen, wenn ihm seine Sensoren und seine Wissensbasis keine eindeutige Interpretation für den gegebenen Befehl liefern. Beim natürlichsprachlichen Zugriff auf Roboter können drei Situationen der Mensch-Maschine-Interaktion unterschieden werden:

- *Auftragsspezifikation*: Befehle, die der Mensch an einen autonomen Serviceroboter gibt, können für diesen von komplexer oder einfacher Natur sein. Die Abstraktionsebene spielt für den Menschen keine Rolle. Er kann ohne weiteres komplexe Befehle, etwa das Zusammenbauen von Objekten, und einfache Befehle, etwa die Roboterbewegung, mischen. Für die Befehlsgenerierung sollte dieser Umstand berücksichtigt werden. Da der Mensch meist räumliche Beschreibungen anstatt Koordinaten in Befehlen an den Roboter benutzt, um räumliche Abhängigkeiten zu schildern, sollten diese analysiert und interpretiert werden können. Das heißt insbesondere, daß eine Abbildung von räumlichen Ausdrücken im Kontext auf konkrete Positionen in der Roboterwelt gefunden werden muß. Vergleichbares gilt auch für die Bezeichnung von Objekten, die in der Welt des Roboters vorkommen. Aus diesen Objektbezeichnungen, die auch räumliche Ausdrücke enthalten können, müssen

durch Abbildungsregeln eindeutige Objekte identifiziert werden. Aus diesen eben genannten Methoden baut sich das Modell der mehrstufigen Referenzsemantik auf (Beispiel: siehe Abbildung 4), das für die Umwandlung von natürlichsprachlich formulierten Befehlen gebraucht wird.

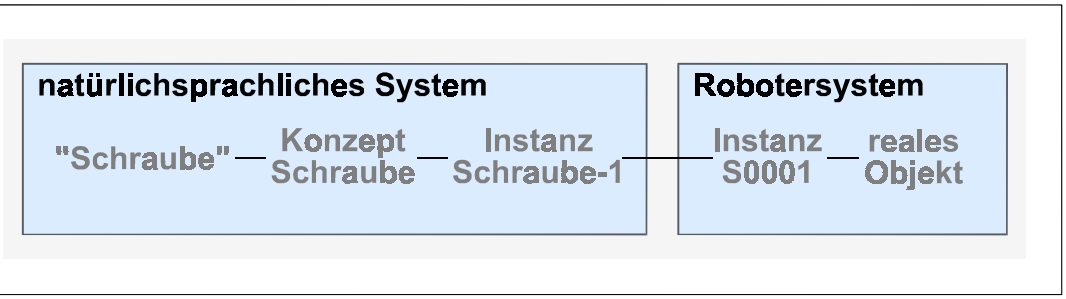

Abbildung 4: Mehrstufige Referenzsemantik bei natürlichsprachlichem Zugang zu Robotern.

- *Erklärungen:*
  1. *Fragen- und Ausführungserläuterung*: Die natürlichsprachliche Ausführungserläuterung setzt auf einer von der Robotersteuersprache unabhängigen Zwischenrepräsentation auf. Da diese unabhängig von der natürlichen Sprache gehalten ist, gibt es keine Bindung an eine spezielle Sprache, die in der Erklärung erzeugt werden muß. Da Serviceroboter meist die Fähigkeit besitzen, autonom Entscheidungen zu treffen, kann eine bestimmte Aufgabe auf verschiedene Arten gelöst werden. In diesem Fall ist es besonders wichtig, daß der Benutzer über die jeweiligen momentanen und zukünftigen Tätigkeiten des Roboters informiert wird. Für die Erklärungen von Fragen oder Robotertätigkeiten gibt es eine mehrstufige Referenzsemantik, in der die Assoziation von Roboterwissen mit natürlichsprachlichen Wissenseinheiten definiert ist.
  2. *Erklärung bei Fehlerkorrekturen*: Die Fähigkeit des Roboters, im Falle eines Fehlers eine Neuplanung durchzuführen, macht es für den Benutzer eventuell schwierig, sein Verhalten nachzuvollziehen oder vorauszusagen. Unter diesen Umständen kann es auf der Benutzerseite zu Mehrdeutigkeiten und Mißverständnissen kommen. Es ist daher sinnvoll, im Falle einer Planänderung die Gründe für die Änderung und deren Art anzugeben.
- *Aktualisierung des Umweltmodells*: Das durch die Sensoren beschränkte Sichtfeld und die gespeicherten Daten können in einer dynamischen, komplexen Umgebung kaum vollständig sein. Benutzer und Roboter sollten sich daher im natürlichsprachlichen Dialog gegenseitig behilflich sein, indem sie Umweltinformationen austauschen.

RONTRA stellt ein allgemeines Modell einer Dialogschnittstelle für Roboter dar. Gegenüber dem Prototyp KANTRA-1 einer Dialogschnittstelle sind in RONTRA viele Einschränkungen aufgrund flexibleren Datenmanagements nicht mehr vorhanden.

2.1.1 Komponenten

Im folgenden sind die Funktionalität der einzelnen Aktionskomponenten der Dialogschnittstelle RONTRA und ihre genauen Aufgabengebiete erklärt:

- *Analyse*: Die natürlichsprachliche Eingabe E_i wird auf ihre Gültigkeit und auf ihren Inhalt hin untersucht. Dazu wird zuerst ihr Satzaufbau (die Syntax) überprüft. Anschließend wird die sprachabhängige Repräsentation der Eingabe auf eine sprachunabhängige abgebildet, indem unter anderem räumliche Ausdrücke auf sprachunabhängige räumliche Relationen abgebildet werden. Für den gesamten Eingabeuntersuchungsprozeß steht das sprachspezifische Wissen L_i zur Verfügung.

- *Evaluation*: Die eingehenden Daten der Analyse- und Kommunikationskomponente werden weiterverarbeitet und danach an die entsprechenden Ein-/Ausgabekomponenten weitergeleitet. Dafür steht das Kernwissen zur Verfügung. Die Evaluationskomponente von RONTRA setzt sich aus den folgenden Modulen zusammen:

  - *Interpretation der natürlichsprachlichen Eingabe*: Die in der Analysekomponente bereits auf ihren Aufbau hin untersuchte Eingabe wird hier semantisch interpretiert. Entsprechend kanonisiert kann die natürlichsprachliche Eingabe einfacher weiterverarbeitet werden.

  - *Interpretation der Anweisungsplänen von Robotern*: Die vom Kommunikationsmodul bereits syntaktisch untersuchten Plandaten des Roboters werden hier semantisch untersucht. Eine dafür entwickelte Zwischenrepräsentation nimmt die relevanten Informationen aus dem aktuellen Planschritt des Anweisungsplans auf.

  - *Normierung der Daten*: Nach Bedarf müssen die weiterzuverarbeitenden Datenstrukturen um Informationen ergänzt werden, die nicht direkt in bereits vorliegenden Daten enthalten sind. Darunter fällt zum Beispiel die Auflösung beziehungsweise die Erzeugung von Referenzausdrücken.

  - *Generierung von Roboterbefehlen*: Die in einer sprachunabhängigen Struktur gespeicherten Befehlsinformationen werden mit Hilfe von Abbildungsregeln auf Aktionsklassen, worin Roboterbefehlsstrukturen modelliert sind, in konkrete Befehle an den Roboter übersetzt.

  - *Aktualisierung des Domänen- und Kernwissens*: Zustandsänderungen, die sich aus Roboteraktionen oder aus der natürlichsprachlichen Eingabe ergeben, werden bei jedem Verarbeitungszyklus im Domänen- beziehungsweise Kernwissen vermerkt, um den Wissensstand von RONTRA aktuell zu halten.

- *Kommunikation*: Diese Ein-/Ausgabekomponente steuert den Datenaustausch mit dem Roboter. Zum einen werden nach Bedarf das Domänenwissen (Umgebungs-, Plan- und Diagnosedaten) vom Roboter abgefragt und die Antworten syntaktisch analysiert; zum anderen hat sie die Aufgabe, die in der Evaluationskomponente generierten Befehle an den Roboter weiterzuleiten.

- *Generierung*: Diese Ausgabekomponente ist für die Verbalisierung der sprachunabhängigen Zwischenstruktur des Erklärungsmoduls zuständig. Die Zwischenrepräsentation kann folgende drei Arten der Information enthalten:

  1. Antworten auf Lokalisationsfragen[13] (sogenannte Wo-Fragen).
  2. Erklärungen von Roboterplanschritten.
  3. Beschreibungen von Fehlerdiagnosedaten.

  Für die Generierung der natürlichsprachlichen Ausgabe A_i steht das sprachspezifische Wissen L_i zur Verfügung.

2.1.2 Wissensbasen

Die Wissensbasen, die der Dialogschnittstelle zur Verfügung stehen, lassen sich wie folgt einteilen:

- *Sprachspezifisches Wissen* umfaßt alle für die Analyse und Interpretation von natürlicher Sprache gebrauchten Angaben. Diese lassen sich folgendermaßen einteilen:
  - *Morphosyntaktisches Wissen* besteht zum einen aus morphologischem Wissen, das der Dialogschnittstelle zur Verfügung steht und für die Analyse von natürlichsprachlichen Eingaben gebraucht wird, zum anderen sind hier Satzstrukturen gespeichert, die zur Generierung von natürlichsprachlichen Erklärungen benötigt werden.
  - *Konzeptuelles Wissen* definiert für die Befehlsgenerierung die Abbildungsregeln von Verben auf Aktionsklassen und die Referenzen von Roboterbefehlen auf Verbrahmen, die zur Erklärungsgenerierung benötigt werden.
  - *Sprachabhängiges semantisches Wissen* versetzt die Analysekomponente der Dialogschnittstelle unter anderem in die Lage, räumliche Ausdrücke in natürlichsprachlichen Anweisungen an den Roboter zu deuten. Des weiteren sind hier natürlichsprachliche Bezeichnungen, Personalpronomen aller Objekte der Roboterwelt gespeichert.

- *Domänenwissen* beinhaltet roboterabhängige Daten bezüglich der Umwelt und der möglichen Handlungen des Roboters. Das Wissen ist in folgende Gebiete aufgeteilt:
  - *Befehlsrepräsentation*: In diesem Wissensbereich ist die syntaktische Spezifikation der einzelnen Roboterbefehle gegeben, die unter anderem für die Erklärung von Plänen benötigt werden.
  - *Umgebungsrepräsentation*: Dieses Wissen gibt der Kommunikationskomponente die notwendigen Informationen, damit die vom Roboter erfaßten Umweltdaten interpretiert werden können. Das Umweltmodell von RONTRA kann mit diesem Wissen mit dem des Roboters konsistent gehalten werden.

[13] Ein Beispiel für eine Lokalisationsfrage ist: „Wo befindet sich die Tasse?“. Die in der Frage auftretenden Objekte sind umweltabhängig.

- *Planrepräsentation*: Hierin sind die Formate der Anweisungspläne von Robotern definiert.

- *Kernwissen* enthält sprach- und roboterunabhängige Informationen, die für die Dialogschnittstelle relevant sind:
  - *Dialoggedächtnis*: Darin werden die zuletzt an den Roboter gestellten natürlichsprachlichen Befehle abgelegt. Damit ist der Analysekomponente der Dialogschnittstelle der Kontext stets bekannt. Dies ermöglicht die korrekte Identifizierung von Personalpronomen in Befehlen. Daneben wird die Erkennung von Ellipsen[14] ([Drosdowski & Eisenberg], Seite 682 f.) ermöglicht.
  - *Sprachunabhängiges semantisches Wissen*: Darin sind die Aktionsklassen für die Befehlsgenerierung und die Verbrahmen für die Erzeugung von Erklärungen definiert. Die Aktionsklassen beinhalten die Strukturen von allen generierbaren Roboterbefehlen. Auf die Bedeutung des Verbrahmens wird im Kapitel 4.3.3 eingegangen.
  - *Aktionswissensbasis*: Ähnlich wie beim Dialoggedächnis werden in dieser Wissensbasis alle Robotertätigkeiten gespeichert, um eventuelle Anweisungsfolgen erkennen zu können. Wird in einer Folgetätigkeit des Roboters dasselbe Objekt manipuliert, so kann in der natürlichsprachlichen Erklärung das Personalpronomen für das Objekt verwendet werden.

- *Internes Wissen* spezifische Informationen des Roboters, wie zum Beispiel Kraft- oder Drehmomentwerte.

2.1.3 Funktionalität

In RONTRA gibt es verschiedene Wege, auf denen Wissen verarbeitet werden. Welcher dafür in Frage kommt, ist primär abhängig von der Informationsquelle, also dem Benutzer oder dem Roboter, und der Art der Information (Frage, Befehl oder Roboteraktionsdaten). Dabei wird zwischen den folgenden Bereichen unterschieden:

- *Anweisungen an den Roboter*: Der in natürlicher Sprache formulierte Befehl an den Roboter wird zunächst durch die Analysekomponente syntaktisch untersucht. Die so gewonnenen Informationen werden durch die Evaluationskomponente in sprachunabhängige Befehlsgenerierungsdaten umgewandelt. Zudem wird mit Hilfe des Kernwissens der Roboterbefehl generiert. Die Kommunikationskomponente übermittelt die fertige Anweisung an den Roboter.

- *Fragen an den Roboter*: Eine Frage vom Menschen an den Roboter wird zuerst von der Analysekomponente syntaktisch untersucht. Mit dem Ergebnis dieser Analyse erzeugt die Evaluationskomponente unter Zuhilfenahme des Kern- und Domänenwissens die Informationsbasis, die von der Generierungskomponente für die Synthese der Antwort gebraucht wird.

[14] Beispiel: Zuerst wird die Anweisung: „Hole einen Schraubendreher aus dem Werkzeugkasten!“ an den Roboter gestellt. Ein in der deutschen Sprache möglicher verkürzter Nachfolgebefehl an den Roboter ist beispielsweise: „Hole noch einen!“.

- *Erklärungen für den Menschen*: Die der Robotertätigkeit zugrundeliegenden Daten werden von dem Roboter an die Kommunikationskomponente geschickt, in der sie syntaktisch analysiert werden. Mit Hilfe dieser Daten und dem Kernwissen wird die zur Erzeugung von natürlichsprachlichen Beschreibungen benötigte Zwischenstruktur aufgefüllt. Sie enthält danach alle relevanten Informationen. Die Generierungskomponente erzeugt im letzten Schritt die fertige natürlichsprachliche Beschreibung.

Der Datenaustausch innerhalb und zwischen den einzelnen Komponenten von RONTRA sieht folgendermaßen aus:

- *Analyse* → *Evaluation*: Die Analysekomponente schickt *teilsemantische Wissenseinheiten*, das heißt teilweise analysierte Eingaben, an die Evaluationskomponente, die dort weiter verarbeitet werden.
- *Evaluation*: Je nach Situation müssen Daten, die aus unterschiedlichen Informationsquellen (aus der Analyse- oder der Kommunikationskomponente) stammen, in ein einheitliches Format umgewandelt werden.
- *Evaluation* → *Kommunikation*: Die interne Roboter-Befehlsrepräsentation wird an die Kommunikationskomponente geschickt.
- *Kommunikation* → *Evaluation*: Die Kommunikationskomponente versorgt die Evaluationskomponente mit den jeweils neuesten Umgebungsbeschreibungen, Anweisungsplänen und, sofern vorhanden, Fehlerdiagnosedaten.
- *Evaluation* → *Generierung*: Von der Evaluationskomponente an die Generierungskomponente werden teilsemantische Wissenseinheiten geschickt. Darin sind die Informationen enthalten, die zur Generierung einer natürlichsprachlichen Anwort oder Erklärung benötigt werden.

2.1.4 Realisierung

RONTRA ist ein Multiagentensystem (siehe Abbildung 5), das aus den folgenden vier Agenten besteht:

- *Ein-/Ausgabemodul RONTRAGUI*: Die graphische Benutzerschnittstelle RONTRA-GUI erweitert die Kommunikations- und Ausführungsfähigkeiten von RONTRA. Sie hat die Aufgabe, Befehle und Fragen in natürlicher Sprache an den Roboter per Tastatur und/oder Mikrophon zu erfassen und an das Steuermodul weiterzuleiten. Weiterhin werden hier die natürlichsprachlichen Erläuterungen von ROPLEX sowie die generierten Roboterbefehle von RIACS angezeigt beziehungsweise über Lautsprecher ausgegeben. Dieser Agent hat darüber hinaus die Aufgabe, den aktuellen Status der einzelnen Agenten zu visualisieren.
- *Erklärungsmodul ROPLEX*: Dieser Agent hat die Aufgabe, auf Plandaten zu warten; diese bei Eintreffen zu interpretieren und auf deren Grundlage natürlichsprachliche Erklärungen zu generieren.
- *Analyse- und Generierungsmodul RIACS*: Die Aufgabe dieses Agenten ist zum einen, bei Bedarf natürlichsprachliche Befehle und Fragen syntaktisch und seman-

tisch zu analysieren, und zum anderen, diese in für den Roboter verständliche Anweisungen zu übersetzen.

- *Steuermodul*: Das Modul steuert den gesamten Datenfluß von RONTRA. Zum Beispiel werden die durch RONTRAGUI erfaßten Eingaben an die entsprechenden Agenten weitergeleitet.

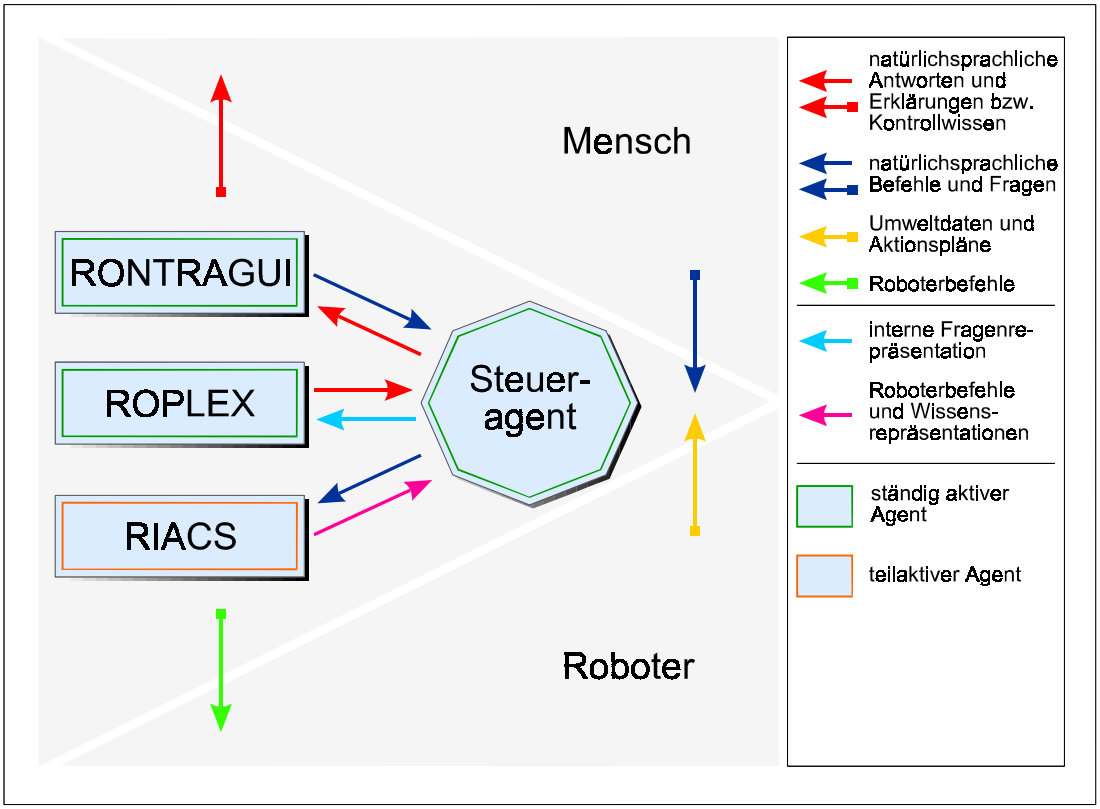

Abbildung 5: Multiagentenmodell von RONTRA.

Der Steueragent und die Agenten ROPLEX und RIACS stellen das *Kernsystem* von RONTRA dar.

2.2 Objektlokalisation durch BOLA

Für die Generierung von Anworten auf Lokalisationsfragen beziehungsweise für die Erzeugung von natürlichsprachlichen Erklärungen mit Lokationsbeschreibungen aus Anweisungsplänen werden *Lokationsausdrücke* [Wahlster et al. 98] benötigt. Das System BOLA, das im Projekt REAL prototypisch realisiert wurde, kann für diesen Zweck genutzt werden. Die Erzeugung von Lokationsausdrücken ist durch den Einsatz von Anytime-Algorithmen jederzeit unterbrechbar. Mit zunehmender Zeit verbessert sich jedoch die Qualität des Ergebnisses.

2.2.1 Eine Anytime-Architektur zur Beantwortung von Lokalisationsfragen

Im Projekt REAL wurde ein kognitives System zur Beantwortung von Lokalisationsfragen eines menschlichen Partners unter verschiedenen Ressourcenbeschränkungen konzipiert und implementiert. Dafür wurden Anytime-Algorithmen benutzt, die als

Sonderfälle von unterbrechbaren Algorithmen [Dean & Boddy 88] angesehen werden können. Die Systemarchitektur gliedert sich in zwei Hauptteile, die konzeptuelle und die linguistische Ebene [Wahlster et al. 98]. Die konzeptuelle Ebene enthält Module, die auf einer zwei- beziehungsweise dreidimensionalen Umgebungsrepräsentation arbeiten, um geeignete Referenzobjekte zu suchen und *räumliche Relationen* [Wahlster et al. 98] zu berechnen. Die Objektlokalisation wird durch die linguistische Ebene in eine kontextadäquate natürlichsprachliche *räumliche Beschreibung* transformiert.

2.2.2 Arbeitsweise

Um möglichst präzise Lokationsbeschreibungen zu erlangen, wird die Generierung auf unabhängige Teilaufgaben verteilt, die durch miteinander kommunizierende Anytime-Prozesse realisiert sind. Zu den wichtigsten zählen die Wahl des besten Referenzobjektes und die Auswahl einer räumlichen Relation, welche die Lagebeziehung zwischen dem zu lokalisierenden Objekt und potentiellen Referenzobjekt möglichst gut beschreibt.

ROPLEX benutzt neben dem Anytime-Modul zur Suche des besten Referenzobjektes das zur Auswahl einer besten räumlichen Relation. Da die Berechnungen unter Umständen zu lange dauern, um eine flüssige Erklärung zu garantieren, wird nach konstanter Zeit das Ergebnis abgefragt und weiterverarbeitet, obwohl dies bedeuten könnte, daß nicht die beste räumliche Relation ausgewählt wurde. Eine schematische Darstellung der Auswahl gibt Abbildung 6 wieder. Dabei sind die einzelnen Berechnungszyklen als vertikale Balken variabler Breite (Zeitdauer) visualisiert und entsprechend der prozentualen Ressourcenverteilung auf die beteiligten Prozesse gegliedert.

In einem ersten Schritt wird ein Referenzsystem aufgebaut. Dessen Ursprung liegt im jeweiligen Referenzobjekt und spannt durch seine drei kanonischen Achsen ein Halbraummodell auf. Die dadurch erreichte binäre Raumunterteilung kann sehr schnell ausgewertet werden. Das Ergebnis ist eine Präferenzliste der Halbräume sowie eine qualitative Distanzinformation. Damit könnte schon eine erste grobe Lagebeschreibung aus einer Distanz und der zum besten Halbraum gehörenden winkelabhängigen Relation generiert werden. Die Qualität kann jedoch noch nicht bestimmt werden.

Auf den Ergebnissen des Halbraummodells bauen zunächst zwei weitere Prozesse auf, die alle gefundenen Relationen hinsichtlich der Qualität auf ihre Anwendbarkeit und Präzision untersuchen. In weiteren Berechnungsschritten wird die Verfeinerung der Beschreibung angestrebt. Falls sich herausstellt, daß die Qualität einer Relation den Ansprüchen einer Lokalisationsbeschreibung nicht genügt, muß diese verbessert werden. Dies wird durch weitere, teilweise parallele Prozesse erreicht. Sie untersuchen auf der Grundlage der erzielten Zwischenergebnisse komplexere räumliche Strukturen:

- Zusammengesetzte räumliche Relationen, wie beispielsweise `links über` oder `rechts bei`.
- Spezielle Relationen, wie `auf` oder `zwischen`, die sich nicht als winkel- oder distanzabhängig charakterisieren lassen.
- Anwendung linguistischer Hecken, wie zum Beispiel `sehr`, `genau` oder `etwa(s)` auf räumliche Relationen.

Das Ergebnis der Berechnungen wird als *räumlicher Ausdruck* bezeichnet.

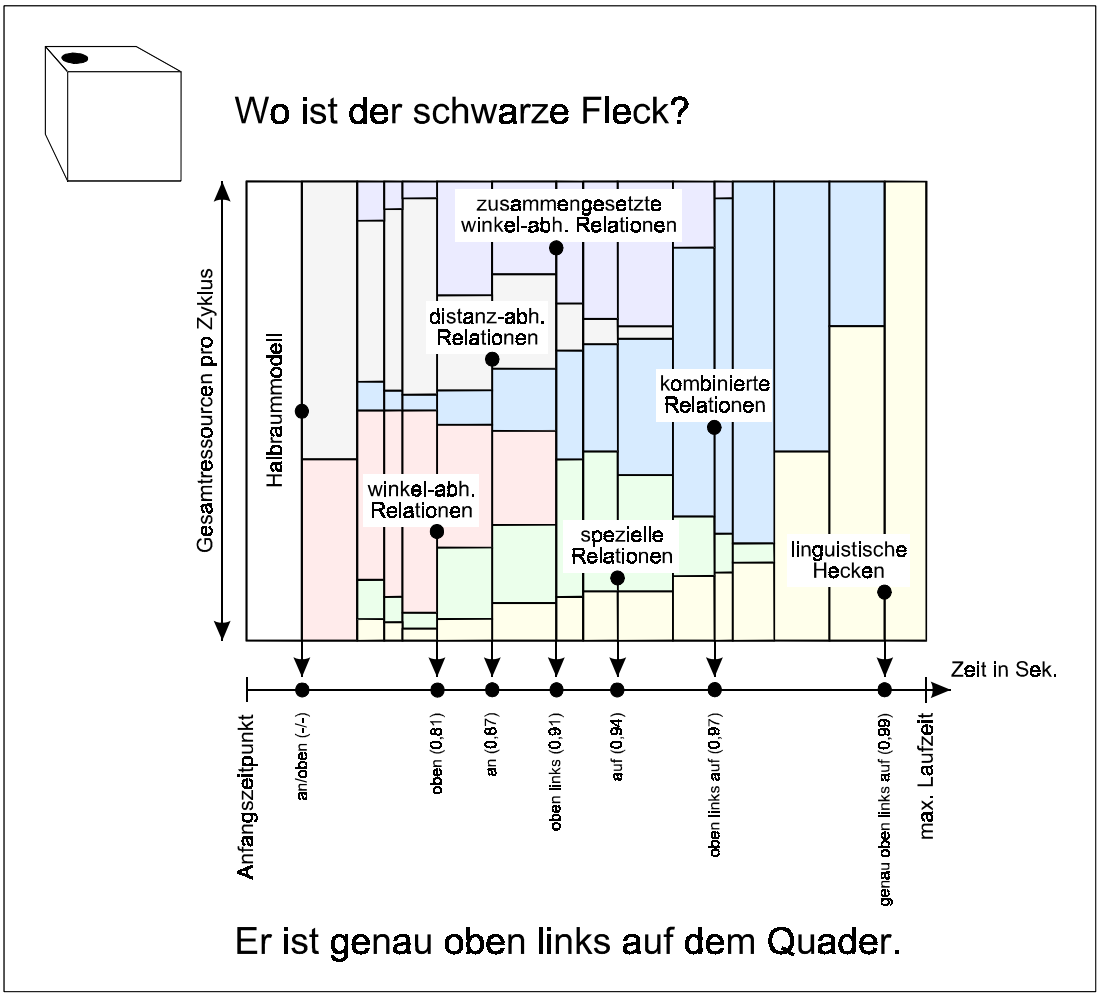

Abbildung 6:Beispiel für die Arbeitsweise von BOLA.

2.3 Eingabeanalyse

Für die natürlichsprachliche Ansteuerung von ROPLEX sind verschiedene Eingaben zugelassen. Sie lassen sich in zwei Kategorien einteilen:

- *Lokalisations- und Modalitätsanfragen*: Damit ist einerseits der Bereich der Anfragen (Wo-Fragen) gemeint, die sich auf die Lage von Szenarioobjekten beziehen, und andererseits die Menge der Fragen hinsichtlich des Fehlerhergangs sowie der Fehlerursache.

- *Einstellungsanweisungen*: Diese Menge umfaßt die Anweisungen des Menschen, die direkt auf die Funktionsweise des Planerklärungsalgorithmus´ einwirken.

Um Befehle, die in natürlicher Sprache formuliert sind, überhaupt verarbeiten zu können, ist es notwendig, alle relevanten Informationen der natürlichsprachlichen Anweisung zu entnehmen und diese in einer geeigneten Datenstruktur zu speichern. Traditionelle Sprachverarbeitungssysteme teilen das Problem der Verarbeitung natürlicher Sprache in verschiedene Schritte auf. Die *syntaktische Analyse*, bei der die grammatische Struktur einer Äußerung unabhängig von ihrer Bedeutung bestimmt wird, und an deren Ende ein sogenannter Parsebaum aufgebaut wird, der die Struktur der Eingabe repräsentiert. Anschließend daran stellt die *semantische Analyse* aus dem Parsebaum eine semantische Repräsentation her, die die Bedeutung der Äußerung im Kontext des jeweiligen Systems enthält. Dieser Vorgang wird in Abbildung 7 veranschaulicht.

Das RONTRA-Teilsystem RIACS bietet einen Zugang zu einem traditionellen Sprachverarbeitungssystem, das ROPLEX für die Analyse von Äußerungen in Anspruch nimmt. Die so gewonnenen semantischen Repräsentationen stehen dem Umwelt- und Planerklärungsmodul über die LISP-Datenstruktur *Eingabeinformation*[15] zur Verfügung.

Sie wird von ROPLEX benutzt, um Informationen über Verb, Fragewort, Subjekt, Objekt und andere Satzteile zu erhalten, die in einer Eingabe vorkommen können.

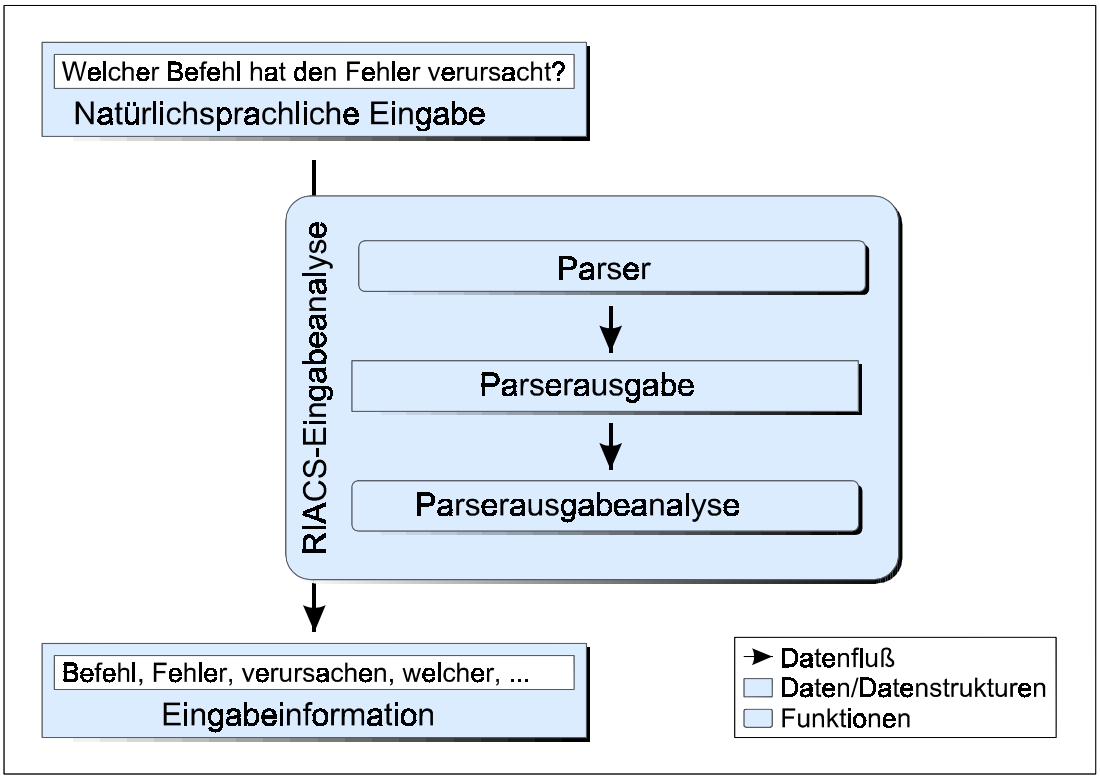

Abbildung 7: Arbeitsweise von RIACS.

[15] Der Aufbau der Datenstruktur wird in ([Jung 97], Seiten 96-98). (dort als Parserextraktliste bezeichnet) beschrieben.

2.4 Eine Anwendungsdomäne

Der autonome Serviceroboter KAMRO kann durch RONTRA vollständig angesteuert werden und stellt somit eine Anwendungsdomäne (mit KAMRO bezeichnet) dar. Die Anwendungsdomäne CHEW wird im Kapitel 6.3 beschrieben.

2.4.1 Der autonome Roboter KAMRO

KAMRO (siehe Abbildung 8) ist ein am IPR der Universität Karlsruhe entwickelter autonomer, mobiler Serviceroboter. Er besteht aus einer mobilen Arbeitsplatte, die durch ein Fahrwerk horizontal in alle Richtungen bewegt werden kann. An dieser Plattform sind an zwei am Ende der Platte senkrecht angeordneten Trägern Greifarme, die sogenannten Manipulatoren, befestigt. Der Roboter hat außerdem verschiedene Sensoren und Kameras für die Orientierung im Raum und die Steuerung der Manipulatoren. KAMRO ist in der Lage, Aufträge, wie den Zusammenbau oder den Transport von Werkstücken, autonom zu lösen.

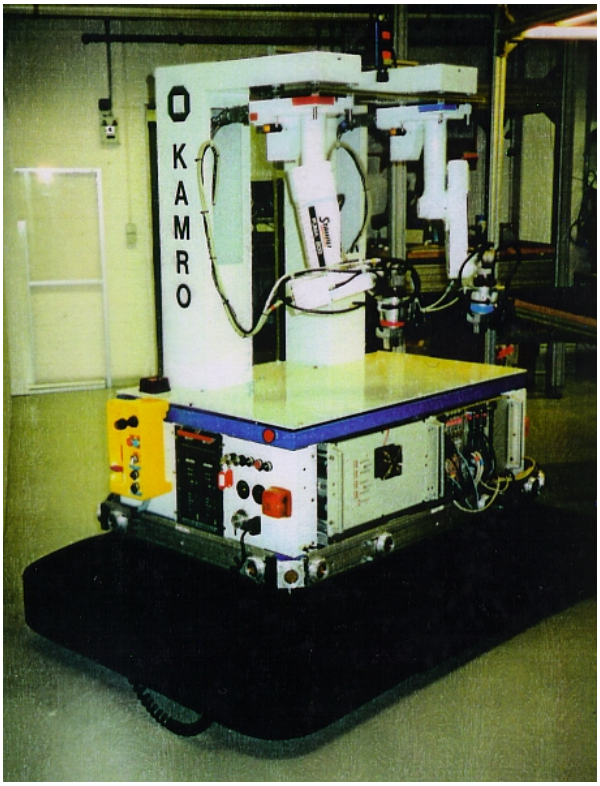

Abbildung 8: Der Roboter KAMRO.

Aufgaben, die KAMRO erledigen soll, können auf drei verschiedenen Abstraktionsebenen formuliert werden:

1. Komplexe Roboteroperationen (`assemble`, ...)
2. Implizite Elementaroperationen (`pick`, `place`, ...)
3. Explizite Elementaroperationen (`grasp`, `transfer`, ...)

Wird dem Roboter eine komplexe Aufgabe gestellt, so wird sie zur Laufzeit durch das Planungssystem FATE zunächst in eine Abfolge von impliziten Elementaroperationen umgewandelt, auf deren Grundlage dann ein Aufgabenplan aus expliziten Elementaroperationen generiert wird. Implizite Elementaroperationen in Aufgabenplänen werden zur Laufzeit in explizite Elemenatoroperationssequenzen umgewandelt. Die Art und Abfolge der expliziten Elementaroperationen hängt dabei von der Position und der Orientierung der Werkstücke auf der Arbeitsplatte ab.

Die Ausführung der Befehle wird durch das Echtzeit-Robotersteuersystem RT-RCS[16] ständig überwacht. Dessen Zustands- und Sensordaten werden an das Planungssystem zurückgegeben, so daß der Plan sofort geändert werden kann, falls dies notwendig werden sollte (siehe Abbildung 9). Tritt dabei eine Ausnahmesituation auf, berechnet FATE einen neuen Aufgabenplan, in den die Reaktion auf das plötzlich eingetretene Ereignis eingebunden ist. Solche Ereignisse sind zum Beispiel das unerwartete Fehlen oder das Auftauchen von Objekten. Die von FATE berechneten expliziten Korrekturanweisungen werden direkt über das RT-RCS an die Hardware von KAMRO weitergegeben.

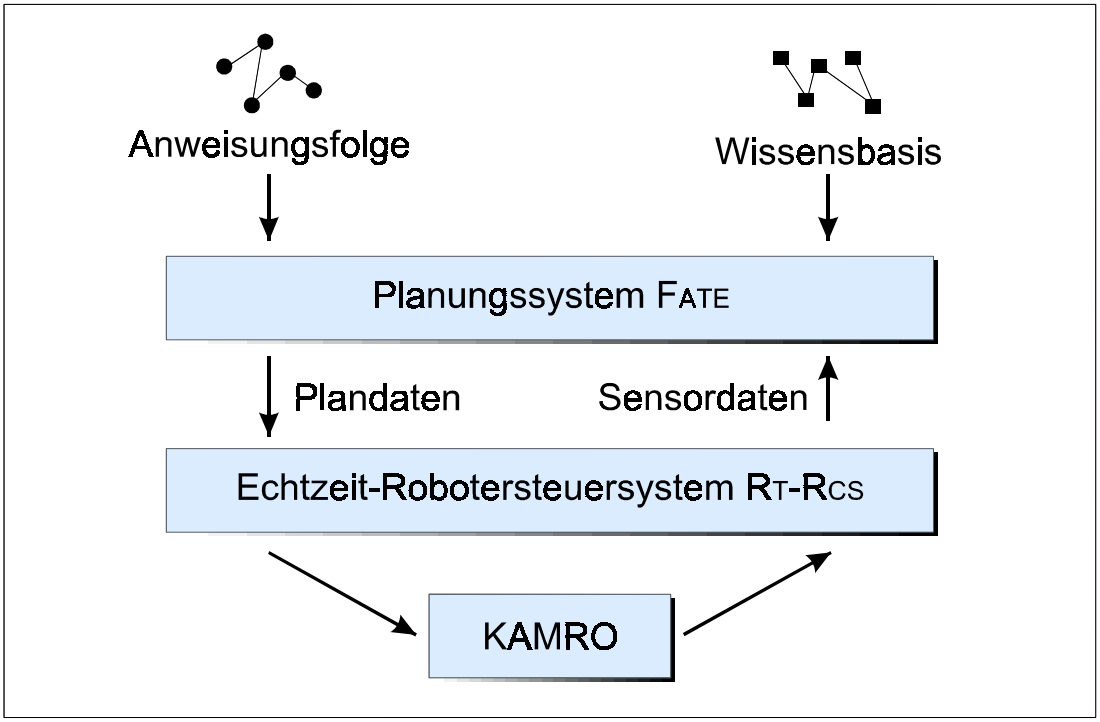

Abbildung 9: Arbeitsweise von FATE und RT-RCS.

Eine komplexe Roboteroperation, die durch das Planungssystem FATE in eine Sequenz von explizite Elementaroperationen zerlegt wird, ist zum Beispiel „`Setze das Cranfield-Assembly-Benchmark zusammen`!“. Dabei soll der Roboter eine Anzahl von Teilen zu einem Werkstück montieren. Die Teile des Cranfield-Assembly-Benchmarks[17] sind dabei willkürlich auf der Arbeitsplatte verteilt. FATE erstellt zunächst einen Ablaufplan expliziter Elementaroperationen, die dann nacheinander über das RT-RCS an die Hardware geschickt werden. Der Test ist dann beendet, wenn der Roboter alle Teile korrekt zusammengefügt hat (siehe Abbildung 10). Er dient unter anderem dazu, die Kommunikation zwischen dem Planungs- und dem Steuersystem von KAMRO zu untersuchen.

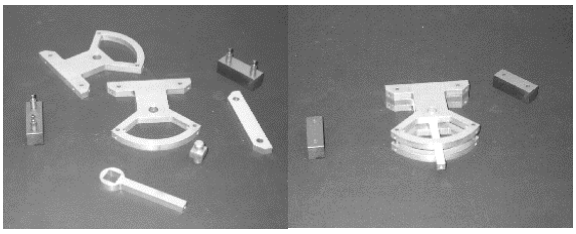

Abbildung 10: CAB-Teile und fertiger CAB.

[16] Abkürzend für **R**eal**T**ime-**R**obot **C**ontrol **S**ystem

[17] Wird abkürzend mit CAB bezeichnet.

2.4.2 Eigenschaften von KAMRO

Der autonome Roboter KAMRO analysiert durch Sensoren seine Umwelt, um deren Aufbau zu erfahren. Diese Informationen sind das Basiswissen des Roboters, wodurch er in der Lage ist, in seiner näheren Umgebung Aufgaben zu erledigen. Das dafür von KAMRO benötigte Wissen gliedert sich in drei Teilbereiche:

- *Umweltdaten (auch Szenariodaten genannt)* repräsentieren die „Weltvorstellung" des Roboters. Darin sind die Eigenschaften der einzelnen Objekte, wie Koordinaten, Farbe und Form, zusammengefaßt.
- *Pläne* versetzten KAMRO in die Lage, komplexe Aufgaben zu lösen. Die Pläne sind aus atomaren impliziten und expliziten Roboteranweisungen aufgebaut. Je nach Situation gibt es Planstrukturen, die für den Zusammenbau von Objekten vorgesehen sind, und solche, die in Ausnahmesituationen eingesetzt werden, um diese zu beheben. Ursachen für solche Situationen sind beispielsweise Positionierungsfehler oder das Fehlen von Objekten.
- *Statusnachrichten*: Sie spiegeln den aktuellen Zustand wider, in dem sich KAMRO befindet. Diese Informationen werden vom Planungssystem FATE generiert und dienen in erster Linie der Überwachung von KAMROs Aktionen.

In den folgenden Abschnitten werden die Eigenschaften der von KAMRO gelieferten erklärungsrelevanten Daten genauer beschrieben.

2.4.2.1 Szenariodaten

Damit sich der autonome Roboter KAMRO in der Umgebung, in der er agieren soll, zurechtfindet, muß er auf irgendeine Art seine Umwelt erfassen. Er besitzt dazu diverse Ultraschall- und Infrarotsensoren, sowie Schwarz-Weiß-Kameras. Aus allen auf diese Art erfaßten Umweltobjekten kann der Roboter eine Repräsentation der realen Umwelt, das sogenannte Roboterszenario, generieren.

```
ONR = <Anzahl der folgenden Objektbeschreibungen>
(<ZS> <OI> <OL> <OT> <OP>)^ONR
```

Abbildung 11: Aufbau des Objektbeschreibungsformats von KAMRO.

Vor jeder Tätigkeit, die KAMRO ausführt, untersucht er das Szenario neu. Die gefundenen Szenarioobjekte werden im in Abbildung 11 beschriebenen *Objektbeschreibungsformat* in einer Datei abgelegt. Im folgenden werden die Einheiten, aus denen sich eine Objektbeschreibung zusammensetzt, erklärt:

- Der *Zeitstempel* (`ZS`) gibt an, zu welchem Zeitpunkt das Objekt zuletzt bewegt wurde.
- Die *Objektinstanz* (`OI`) ist die genaue, eindeutige Bezeichnung des Objektes. Sie besteht aus dem Namen der Objektklasse und der eigentlichen Instanzbezeichnung, die durch einen Doppelpunkt getrennt werden.

- *Objektlage* (`OL`): Sie gibt die Lage an, die das Objekt in der Welt annimmt. Beispielsweise kann ein Zylinder auf seiner Grundfläche stehen oder auf seinem Mantel liegen. In der Welt von KAMRO wird durch die Objektlage die jeweilige Objektseite angegeben, auf der das Objekt ruht. Die Seitenplatte des Cranfield-Assembly-Benchmarks kann zum Beispiel drei verschiedene stabile Lagen annehmen. Dabei bedeutet die Lagebeschreibung `s1`, daß das Objekt auf seiner Grundfläche steht. Die Bezeichnung `s2` steht für das Liegen des Objektes auf einer seiner Seitenfläche. Die Lagebeschreibung `s3` bedeutet, daß die Seitenplatte auf einer Kantenfläche steht.
- *Ausrichtung des Objektes* (`OT`): Sie gibt die Drehung des Objektes im Uhrzeigersinn bezüglich einer definierten Nullstellung an.
- *Position des Objektes* (`OP`): Grundsätzlich wird hiermit die Objektposition im aktuellen Szenario angegeben.

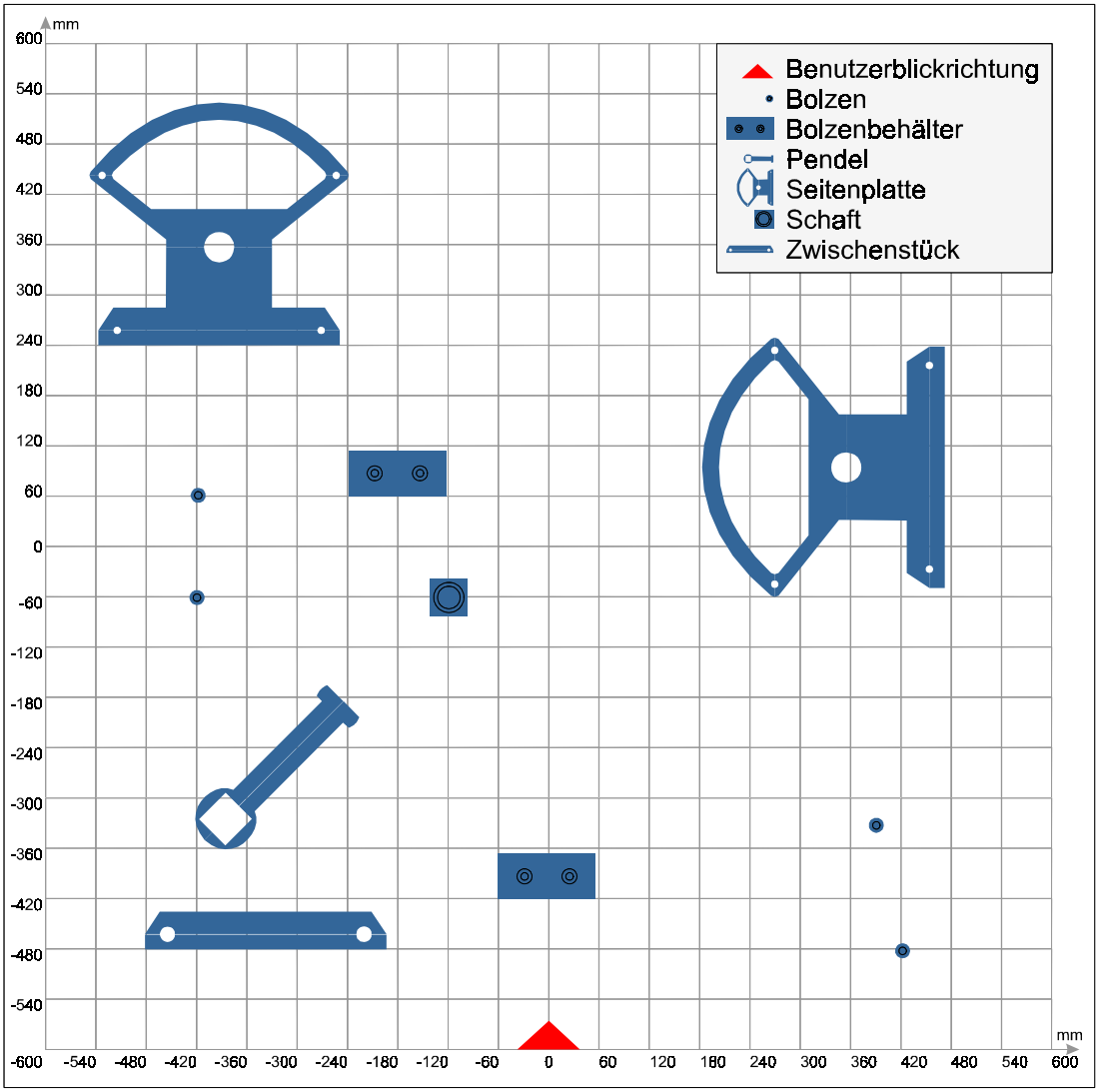

Abbildung 12: Beispielszenario mit Objekten des CAB.

Der folgende Abschnitt zeigt eine Beschreibung eines CAB-Szenarios (siehe dazu Abbildung 12), wie sie von KAMRO geliefert wird.

```
11
0 lever:1 s1 135 [-420,-360,0]
0 shaft:1 s1 0 [-100,-80,0]
0 sideplate:1 s1 0 [-540,240,0]
0 sideplate:2 s1 90 [470,-50,0]
0 spacer:1 s1 0 [-420,50,0]
0 spacer:2 s1 0 [-420,50,0]
0 spacer:3 s1 0 [-420,-480,0]
0 spacer:4 s1 0 [390,-340,0]
0 spacerreservoir:1 s1 0 [-240,60,0]
0 spacerreservoir:2 s1 0 [-60,-420,0]
0 spacingpiece:1 s1 0 [-480,-480,0]
```

Dieses Beispiel gibt eine ursprüngliche Objektposition wieder, das durch den Zeitstempel `0` zu Beginn jeder Objektbeschreibung gekennzeichnet ist. Alle Objekte (**`lever`**:**`1`**, ...) liegen auf ihrer Grundfläche (`s1`). Wie Abbildung 12 zeigt, sind das Pendel (**`lever`**:**`1`**) um 135 Grad und die zweite Seitenplatte (**`sideplate:2`**) um 90 Grad bezüglich ihrer Nullstellung gedreht. Die Koordinateninformation (Beispiel: `[-420,-360,0]` für das Pendel) am Ende einer Objektbeschreibung gibt die aktuelle Position des Objekts an.

2.4.2.2 Plansyntax

Das KAMRO-System unterscheidet bei Plänen zwischen Anweisungs- und Fehlerbehebungsplänen [Laengle et al. 96]. In Abbildung 13 wird deren allgemeiner Aufbau beschrieben. Anweisungspläne bestehen im wesentlichen aus einer endlichen Liste von impliziten, expliziten und/oder komplexen Roboterbefehlen. Dabei gibt es keine Regeln, in welcher Reihenfolge die Befehle aufgelistet werden müssen. Im Gegensatz dazu haben Fehlerpläne ein Gliederungselement, die sogenannte *Sektionsinformation*. Sie beschreibt kurz in codierter Form die Auswirkung der sich in der anschließenden Liste befindlichen impliziten, expliziten beziehungsweise komplexen Roboterbefehle. Fehlerpläne können eine endliche Anzahl dieser Sektions-Strukturen beinhalten.

2.4.2.3 Syntax und Semantik von Roboterbefehlen in Plänen

Die Menge der Befehle, die für KAMRO relevant sind, ist in sogenannte *explizite* und *implizite Elementaroperationen* (*EEOs*, *IEOs*) sowie *komplexe Roboteranweisungen* unterteilt. Die Primitiven der Anweisung sind die EEOs. Implizite Elementaroperationen und komplexe Roboterbefehle werden vom Planungssystem FATE bei der Befehlsausführung in EEOs zerlegt. Die Syntax von nicht in Plänen benutzten Befehlen entspricht bis auf die Schlüsselsequenz `eodef` der Syntax derjenigen Befehle, die in Plänen zu Einsatz kommen. Abbildung 14 zeigt beispielhaft zwei in der Syntax für Befehle an den Roboter KAMRO formulierte Planschritte, die aus den impliziten Befehlen „Greifen“ (**`pick`**) und „Ablegen“ (**`place`** ... **`placed`**) bestehen.

Die Syntax von expliziten Elementaroperationen sowie die Beschreibung der in den Operationen benutzten Variablen werden im folgenden beschrieben:

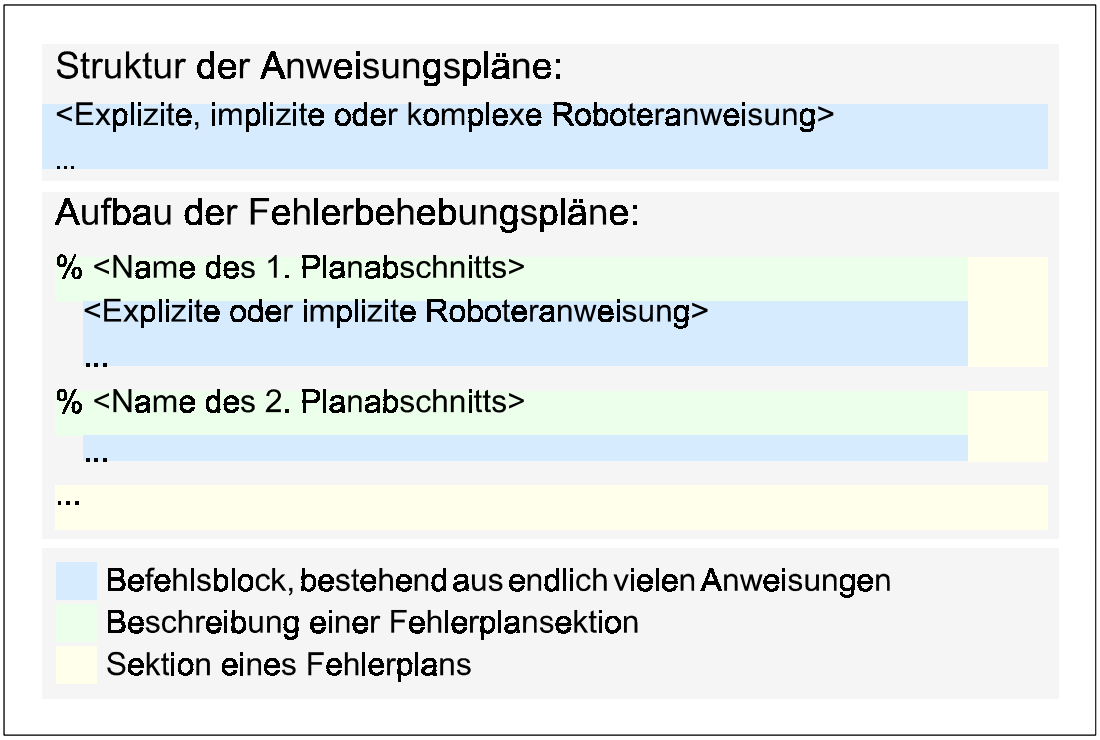

Abbildung 13: Syntax der verschiedenen Planformen für KAMRO.

- Vorsichtiges Loslassen eines Objektes:

```
eodef (detach, <P>,  [
                       [agent_name, <A>],
                       [gripobj_id, [workpiece, <OI>]],
                       [gripobj_pos, [<CO>], <WL>]
                     ]
      )
```

- Vorsichtige Bewegung eines Manipulators zur Zielkoordinate:

```
eodef (finemotion, <P>,  [
                           [agent_name, <A>],
                           [gripobj_pos, [<CO>], < WL >]
                         ]

      )
```

- Vorsichtiges Greifen eines bestimmten Objektes:

```
eodef (grasp, <P>, [
                     [agent_name, <A>],
                     [gripobj_id, [workpiece, <OI>]]
                     [gripobj_pos, [<CO>], < WL >]
                   ]
      )
```

- Einstellen der Öffnungsweite des Manipulators:

```
eodef (setgripper, <P>, [
                          [agent_name, <A>],
                          <W>
                        ]
       )
```

- Schnelle Bewegung eines Manipulators zur Zielkoordinate:

```
eodef (transfer, <P>, [
                         [agent_name, <A>],
                         [gripobj_pos, [<CO>], < WL >]
                       ]
       )
```

Die impliziten Elementaroperationen von KAMRO sind:

- Greifen eines bestimmten Objektes:

```
eodef (pick, <P>,  [
                     [agent_name, <A>],
                     [gripobj_id, [workpiece, <OI>]]
                   ]
       )
```

Greifen des CAB-Objektes `lever:1` (Hebel):

```
eodef (pick, 3,
       [[agent_name, select],
        [gripobj_id, workpiece, lever:1]]
       ]
      )
```

Ablegen des CAB-Objektes `lever:1`:

```
eodef (place, 3,
       [[agent_name, select],
        [gripobj_id, workpiece, lever:1]]
        [gripobj_pos, [placed, lever:s1, 0, [0, 450], [workcell, assembly_station]]]
       ]
      )
```

Roboterbefehl
Befehlsausführender Mechanismus
Objektinstanz
Objektposition

Abbildung 14: Beispieloperationen für den Roboter KAMRO.

- Ablegen eines bestimmten Objektes auf das Arbeitsgebiet:
  `eodef` (**`place`**, <*P*>, [
  [`agent_name`, <*A*>],
  [`gripobj_id`, [`workpiece`, <*OI*>]],
  [`gripobj_pos`,
  [**`placed`**, <*SL*>, <*OR*>, [<*CO*>], < *WL* >]
  ]
  ]
)
- Zusammenfügen von zwei Objekten:
  `eodef` (**`place`**, <*P*>, [
  [`agent_name`, <*A*>],
  [`gripobj_id`, [`workpiece`, <*OI*>]],
  [`gripobj_pos`,
  [**`connected`**, <*SL*>,
  [[<*CT*>, <*OR*>, [<*CO*>], <*T*>, <*L*>, < *WL* >]]
  ]
  ]
  ]
)

Folgende komplexe Elementaroperation wird aktuell von KAMRO verstanden:

- Zusammenbau-Anweisung:
  `eodef` (**`assemble`**, <*P*>, `cranfield_assembly_benchmark`)

Die Zeichenfolge `eodef`[18] markiert den Beginn einer Elementaroperation. Die Variablen, die in den Elementaroperationen zur Strukturierung verwendet wurden, haben folgende Bedeutungen:

- Priorität $P \in \{$`2`, `3`$\}$.
  Hiermit wird festgelegt, mit welcher Priorität der Befehl vom Roboter behandelt wird. Ist zum Beispiel ein Befehl mit Priorität 2 typisiert, dann können nur Befehle, die unter Priorität 1 eingestuft sind, diesen Befehl unterbrechen. Der Standardwert der Priorität ist 3. Nur elementare Befehle wie der *Halt-Befehl*[19] haben Priorität 1.
- Greifarm (Manipulator) $A \in \{$`blue`, `red`$\}$.
  Mit dieser Information wird der Greifarm festgelegt, der den Befehl ausführen soll. Dabei steht `blue` für den linken Manipulator und `red` für den rechten.
- Objektinstanz $OI \in \{$`lever:1`, `shaft:1`, `spacer:1`, `spacer:2`, `spacer:3`, `spacer:4`, `spacingpiece:1`, `spacerreservoir:1`, `spacerreservoir:2`, `sideplate:1`, `sideplate:2`$\}$.

[18] Steht abkürzend für **e**lemental **o**peration **def**inition.

[19] Systembeinflußende Befehle wie „Halt“ gehören nicht zur Klasse der Elementaroperationen.

Die oben gezeigten Beispielwerte sind Instanzen aus dem Cranfield-Assembly-Benchmark.

- Koordinaten $CO \in \{[\texttt{x y z}] \mid \texttt{x}, \texttt{y} \in [-600; 600], \texttt{z} \in [0; 600]\}$.
  Die verwendete Längeneinheit ist Millimeter. Der Wertebereich der Koordinaten ist beschränkt, da KAMRO in einer speziell für ihn angefertigten Arbeitsumgebung operiert.

- Arbeitsumfeld $WL \in \{[\texttt{workcell}, \texttt{assembly\_station}]\}$.
  Prinzipiell ist an dieser Stelle der Plananweisung der Ort, wo KAMRO seine Operation ausführt, angeben. Da die Arbeiten von KAMRO während der Entstehung dieser Arbeit in einer Spezialumgebung stattfanden, besteht die Menge der Arbeitsumfeldbeschreibungen nur aus einem Eintrag.

- Stabile Lage $SL \in \{$`lever:s1`, `shaft:s1`, `shaft:s2`, `spacer:s1`, `spacer:s2`, `spacerreservoir:s1`, `spacerreservoir:s2`, `spacingpiece:s1`, `spacingpiece:s2`$\}$. Die Beispielwerte sind auf die CAB-Objekte bezogen.

  Die stabile Lage gibt an, wie das Objekt in der aktuellen Umgebung steht oder liegt. Zum Beispiel kann ein Verbindungsbolzen (`spacer`) aufrecht stehen, was durch die Zeichenkette `spacer:s1` ausgedrückt wird. Die Zeichenkette `spacer:s2` gibt an, daß der Verbindungsbolzen auf der Seite liegt.

- Orientierung $OR \in [-180; 180]$.
  Sie gibt an, um wieviel Grad das Objekt bezüglich seiner Ursprungslage gedreht ist.

- Öffnungsweite $W \in [0, 50]$.

  Mit ihr wird die Öffnungsweite (die Einheit ist Millimeter) der Greifhand von KAMRO bestimmt.

- Fügeoperation $CT \in \{$`assembling`, `fixing`, `hole_on_pin`, `pin_on_hole`$\}$.
  Die Roboteranweisung **`place`** ... **`connected`** wird durch die Angabe der Fügeoperation weiter spezifiziert. Die dem Roboter KAMRO bei einem Fügebefehl zur Verfügung stehenden Operationen beschreiben die Art, wie zwei Objekte zusammengefügt werden. Die Fügemethode `hole_on_pin` besagt, daß die Löcher des gehaltenen Objektes auf die sogenannten Halterungsstifte des anderen Objektes gesteckt werden. Mit `pin_on_hole` ist die umgekehrte Methode gemeint, wobei die Halterungsstifte des gehaltenen Objektes in die Löcher des anderen Objektes gesteckt werden. Die Fügemethode `assembling` besagt, daß zwei Objekte locker zusammengefügt werden. Die Methode `fixing` besagt, daß die in die Fügeoperation involvierten Objekte fest miteinander verbunden werden. Objekte, die auf diese Art zusammengefügt wurden, können nicht mehr getrennt werden. Sie bekommen deshalb eine gemeinsame neue Objektbezeichung.

- Fügetiefe $T \in [0; 600]$.
  Die Fügetiefe ist ein weiterer Parameter der Methoden `pin_on_hole` beziehungsweise `hole_on_pin`. Das dabei verwendete Längenmaß ist Millimeter.

- Objektlochmenge: $L \subseteq \{$`lever:s1:h1`, `sideplate:s1:h1`, `sideplate:s1:h2`, `sideplate:s1:h3`, `sideplate:s1:h4`, `sideplate:s1:h5` `spacerreser-`

voir:s1:h1, spacerreservoir:s1:h2, spacingpiece:s1:h1, spacing-piece:s1:h2}
Mittels dieser Information werden die bei einer Fügeoperation verwendeten Löcher der betroffenen CAB-Objekte definiert.

2.4.2.4 Statusinformationen

Der autonome Roboter wird permanent durch das Echtzeit-Robotersteuersystem überwacht. Die so gewonnenen Statusinformationen werden dem Planungssystem übermittelt, das daraus je nach Situation folgende Nachrichten generiert:

- Im „normalen“ Planabarbeitungsfall: `plan <Planname>: <Schritt>`

  Der `Planname` gibt den aktuellen Anweisungsplan von KAMRO an. Um zu überprüfen, welche Anweisung der Roboter aktuell ausführt, ist in dieser Statusinformation der momentan ausgeführte Planschritt (`Schritt`) angegeben.

- Im Fall der Fehlerplanabarbeitung: `plan <Planname>: <Sektion>.<Schritt>`

  Wird ein Fehlerplan abgearbeitet, so wird die Statusinformation um die Sektionsinformation ergänzt.

- Im akuten Fehlerfall: `diag <IEO> <Fehler> <Ursache> <EEO> <Ziel>`

  Mögliche Einträge für die Felder `Fehler`, `Ursache` und `Ziel` finden sich in Abbildung 15.

`diag <IEO> <Fehler> <Ursache> <EEO> <Ziel>`

Mögliche Fehler bzw. Ursachen und deren Bedeutung:

`crash`	Kollision
`crash-coagent`	Kollision mit Manipuator
`not-servo`	Servomotorfehler
`not-startposition`	Falsche Anfangsposition
`not-endposition`	Falsche Endposition
`not-free-agent`	Der Greifer ist schon besetzt
`not-open-agent`	Der Greifer ist geschlossen
`not-position-bo`	Ein starres Objekt sollte gegriffen werden
`not-positon-o`	Ein falsches Objekt befindet sich an der Position
`not-grasped-not-o`	Ein falsches Objekt wurde gegriffen
`not-grasped-o`	Ein Objekt wurde falsch gegriffen
`not-freespace-placegoal`	Es ist kein Platz zum Hinstellen vorhanden
`not-o-picktarget`	Ein anderes Objekt liegt an "Greifstelle"
`not-freespace-picktarget`	Es ist kein Platz vorhanden, um ein Objekt zu greifen
`not-coagent-grasped-o`	Anderer Manipulator hat falsches Objekt gegriffen
`not-coagent-not-grasped-o`	Anderer Manipulator hat Objekt falsch gegriffen

Mögliche Ziele und deren Bedeutung:

`verify-world-model`	Die Umweltdaten werden neu erfasst
`remove-disturbing-o`	Stoerende Objekte werden entfernt
`repick-o`	Das Objekt wird erneut gegriffen
`ignore`	Das Objekt wird erneut gegriffen

Abbildung 15: Fehlerbeschreibungen von KAMRO.

Meldet das RT-RCS eine Fehlersituation, so generiert das Planungssystem die Diagnosemeldung `diag`, worin Angaben über den fehlerverursachenden impliziten und expliziten Befehl (`IEO`/`EEO`) sowie Kurzinformationen über Fehler (`Fehler`), Ursache (`Ursache`) und Fehlerbehebung (`Ziel`) enthalten sind. Ist beispielsweise beim Greifen (**`pick`**-Befehl) eines Bolzens (**`spacer`**:**`1`**) ein Fehler aufgetaucht, könnte das Feld `Fehler` mit der Information `not-freespace-picktarget` besetzt sein. Das bedeutet konkret, daß für das Greifen des Bolzens kein Platz vorhanden ist. Das `Ziel`-Feld der Diagnosemeldung könnte demzufolge mit `remove-disturbing-o` belegt sein, was andeutet, daß störende Objekte vom Roboter entfernt werden.

2.4.3 KANTRA-1 als Prototyp einer Dialogschnittstelle

KANTRA-1 [Gebhard & Jung 96] ist der erste Prototyp der natürlichsprachlichen Dialogschnittstelle KANTRA [Lüth et al. 94], [Stopp et al. 94]. Damit war eine natürlichsprachliche Ansteuerung des Roboters KAMRO mittels der IEOs **`pick`** und **`place`** ... **`placed`** möglich. Eine Aufgabe des Prototyps ist, Befehle an den Roboter, die in natürlicher Sprache formuliert sind, in elementare für KAMRO verständliche Roboterbefehle zu übersetzen. Dazu wird im ersten Schritt die durch einen Parser erzeugte syntaktische Repräsentation des natürlichsprachlichen Befehls analysiert. Im zweiten Schritt wird eine semantische Repräsentation von räumlichen Relationen aufgebaut, auf deren Grundlage die elementaren Roboterbefehle generiert werden.

Wie Abbildung 16 zeigt, wird zuerst die natürlichsprachliche Eingabe für den Serviceroboter KAMRO durch einen Parser syntaktisch untersucht. Im Prototyp KANTRA-1 wird der Chart-Parser SB-PATR mit der Morphologiekomponente MORPHIX ([Finkler & Neumann 88], Seiten 11-19), die auf ein Stammformenlexikon aufgesetzt ist, verwendet. Die anschließende semantische Analyse hat zur Grundlage die Merkmalstruktur, die vom Parser bei der grammatischen Untersuchung der natürlichsprachlichen Eingabe erzeugt wurde. Das Ergebnis der semantischen Analyse sind Wortlisten, die alle semantischen Informationen der natürlichsprachlichen Anweisung beinhalten. Der Vorteil dabei ist, daß diese Struktur schneller angesprochen werden kann und somit die Informationen für die semantische Analyse schneller verfügbar sind, als wenn sie direkt aus der Merkmalstruktur abgefragt werden müßten. Die Wortlisten sind die Grundlage für die Generierung der Roboterbefehle. Dazu sind zunächst die in jeder Wortliste enthaltenen Informationen über die Art der Operation und die daran beteiligten Objekte[20] (beim **`pick`**-Befehl nur das zu greifende Objekt, beim **`place`**...**`placed`**-Befehl das zu plazierende Objekt und das Bezugsobjekt, das zur Berechnung der Zielkoordinaten verwendet wird) auszuwerten. Die Roboteroperationen sind in KANTRA-1 auf **`pick`** und **`place`**...**`placed`** beziehungsweise das jeweilige natürlichsprachliche Verb-Pendant greifen, nehmen, stellen, ablegen und so weiter beschränkt. Das System ANTLIMA[21] [Schirra & Stopp 93] liefert die Werkzeuge für die Objektidentifikation. Dazu wird eine Referenzsemantik für die in natürlichsprachlichen Roboterbefehlen vorkommenden räumlichen

[20] Die Objekte beschränken sich auf die Teile des Cranfield-Assembly-Benchmarks.

[21] **Ant**icipation of the **l**istender' **ima**gery (Vorwegnahme der Bildvorstellung des Hörers).

Ausdrücke aufgebaut. Damit können Koordinatendaten ausgewertet werden, die für die Synthese von Roboterbefehlen gebraucht werden.

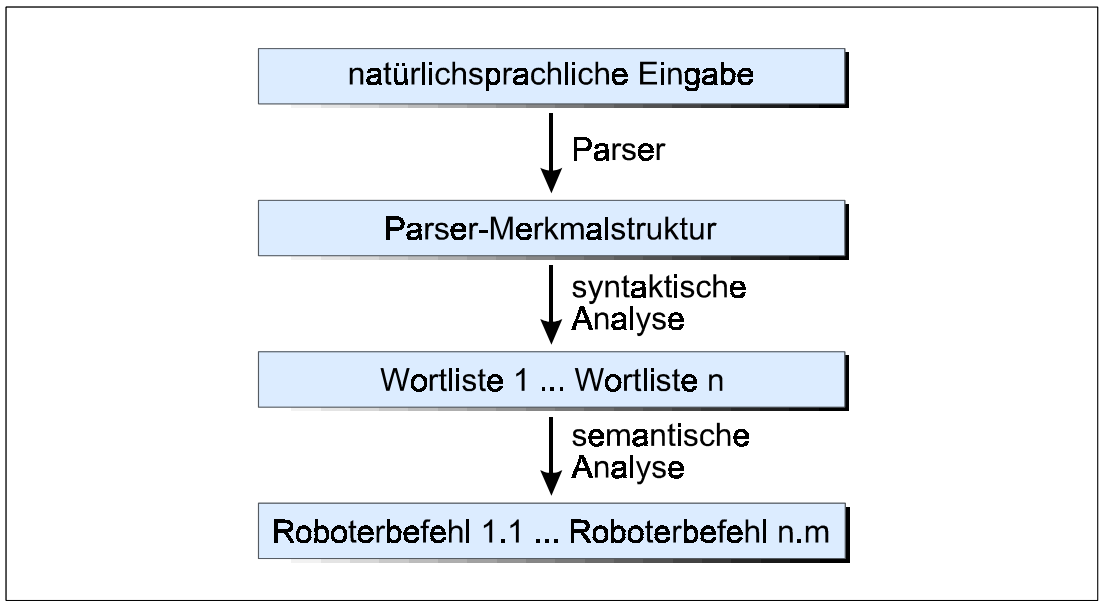

Abbildung 16: Arbeitsweise von KANTRA-1.

Die Kommunikation zwischen KANTRA-1 und KAMRO verlangt einen gegenseitigen Datenaustausch. Allgemein handelt es sich dabei um Steuerdaten für das jeweilige System. Für die Generierung der Roboterbefehle werden von KAMRO Informationen über Ort und Lage der aktuellen Objekte benötigt. Zusammengefaßt sind das die Szenarioinformationen. Im Austausch dazu erhält KAMRO Roboterbefehle, die aus natürlichsprachlichen Sätzen generiert worden sind. Der reine Datenaustausch geschieht in folgenden zwei Schritten:

1. Die Szenarioinformationen werden bei jedem Befehl an den Roboter neu angefordert und in einer temporären Datei abgelegt. Diese Maßnahme ist unabdingbar, da die meisten Anweisungen zu einer Veränderung des Szenarios führen und ihrerseits die Berechnung der erforderlichen räumlichen Relationen beeinflussen. Damit wird das Szenariowissen konsistent gehalten.
2. Der von KANTRA-1 erzeugte Roboterbefehl wird an KAMRO übermittelt.

Der Datenaustausch geschieht mittels PVM[22] [Geist et al. 94]. PVM ist ein portables, programmierbares Nachrichtenaustauschsystem, das dafür entwickelt worden ist, verschiedene Computersysteme zu vernetzen, um damit einen virtuellen Computer zu schaffen. Dieser virtuelle Computer kann als ein eigenständiges System angesehen werden. Dabei ist es nicht relevant, wo die einzelnen Rechner sich physikalisch befinden und mit welchem Betriebssystem sie ausgestattet sind. Für das Errichten einer solchen Anlage ist nur erforderlich, daß die einzelnen Computersysteme über ein Netzwerk miteinander verbunden sind.

KANTRA-1 dient als Grundlage für die Entwicklung der Dialogschnittstelle RONTRA.

[22] **P**arallel **V**irtual **M**achine

3 Problembeschreibung und Lösungsansatz

3.1 Problembeschreibung

Daten, die zur Generierung für natürlichsprachliche Erklärungen von Roboterbefehlsplänen oder für Antworten auf Lokalisationsfragen gebraucht werden, weisen, je nach Robotersystem, eine andere Syntax auf. Damit eine flexible Erfassung dieser Daten gegeben ist, muß der Datenanalyseprozeß auf das jeweilige Format angepaßt werden können. Dies trifft besonders auf das Erfassen von Plänen zu. Zu diesem Zweck muß dem Datenerfassungsystem das Wissen über die Syntax von Planschritten, die es verstehen soll, gegeben werden können.

Ein weiterer Aspekt, der bei der Erzeugung von natürlichsprachlichen Erklärungen eine wichtige Rolle spielt, ist die zeitgleiche Erklärung der vom Roboter aktuell ausgeführten Aktion. Dabei muß einerseits eine Auswahl der Informationen getroffen werden können, die im aktuellen Planschritt und in den aktuellen Umweltinformationen zu finden sind, da eine zu große Informationsanhäufung die Verständlichkeit der Erklärungen eher herabsetzen als erhöhen würde. Anderseits sollten zur Überprüfung der korrekten Planabarbeitung möglichst viele Angaben in der natürlichsprachlichen Erklärung untergebracht werden können. [23] Damit ein flexibles Arbeiten mit dem Roboter möglich wird, sollte der Erklärungsalgorithmus bezüglich der Informationskomplexität der Erklärung einstellbar sein, um den Bedürfnissen unterschiedlicher Benutzer gerecht zu werden. Generell muß die Analyse und Auswertung der Planschritte fast ohne Zeitverluste ablaufen, um asynchrone Erklärungen zu vermeiden. Daher müssen auch Umweltinformationen schnell und unproblematisch zur Verfügung stehen.

Damit sich der Benutzer in dem Roboterszenario besser orientieren kann, muß es die Möglichkeit geben, den Aufenthalt von Szenarioobjekten zu erfragen. Die Antwort soll neben der Bezeichnung des zu suchenden Objektes eine eindeutige räumliche Beschreibung enthalten. Die dafür benutzten Algorithmen können auch für Erklärungen von Plänen dienen, in denen räumliche Beschreibungen, wie zum Beispiel: „Ich greife gerade die Tasse *vor dem Teller*.“, benutzt werden, um ein Objekt genauer zu spezifizieren.

[23] „*Make your conversational contribution such as is required, at the stage at which it occurs, by the accepted purpose or direktion of the talk exchange in which you are engaged*“ [Grice 75].

Um die Planerklärung in verschiedenen natürlichen Sprachen zu ermöglichen, sollte die interne Zwischenrepräsentation der Planinformation möglichst flexibel hinsichtlich der Informationsaufnahme sein. Ihr Aufbau darf explizit keine roboter- und sprachabhängigen Strukturen mehr enthalten, da dies unter anderem bedeuten würde, daß für jedes Robotersystem ein anderes Kernwissen notwendig wäre.

Die Schwierigkeit der natürlichsprachlichen Erklärung von Planschritten für Roboter besteht demnach in der Erfassung und Organisation der zur Generierung der natürlichsprachlichen Erklärungen notwendigen Daten und der Art und Weise, wie eine für den Menschen möglichst optimale Beschreibung der Planschritte erzeugt werden kann.

3.2 Anforderungen an den Erklärungsprozeß

Parallel zur Beschreibung der vom Roboter augenblicklich ausgeführten Tätigkeit soll der Mensch die Art und Weise der Erklärungen auf einfachstem Weg manipulieren können, um somit ein für ihn bestes Ergebnis bezüglich der Tätigkeitsbeschreibung zu erlangen. Die beiden Fragen, *wie* beziehungsweise *was* dem Menschen bei einer Planabarbeitung erklärt werden soll, werden in den beiden anschließenden Kapiteln beantwortet.

3.2.1 Was soll erklärt werden?

Voraussetzung ist, daß der Robotertätigkeit ein *Plan* zugrunde liegt, in dem die einzelnen *Roboteranweisungen* (*Roboterbefehle*), die als *Planschritte* modelliert sind, untereinander aufgelistet sind. Sie sagen dem Roboter, wie er Objekte in seiner Umwelt manipulieren soll. Die Dinge der Umwelt, die direkt von ihm manipuliert werden, heißen *Aktionsobjekte*. Die Gegenstände, auf die sich in einer Anweisung bezogen wird, beziehungsweise die zur Identifizierung von Aktionsobjekten in der Umwelt dienen, heißen *Referenzobjekte*.

Eine Aufgabe, wie zum Beispiel das Zusammenbauen eines Turmes aus Bauklötzchen, die der Roboter bewältigen soll, besteht aus endlich vielen Roboteranweisungen, die in einem *Anweisungsplan* (*Befehlsplan*) zusammengefaßt sind. Tritt nun während der Abarbeitung eine *Fehlersituation* auf, so versuchen autonome Serviceroboter, durch einen Anweisungsplan den aufgetretenen Fehler zu korrigieren. Es zu beachten, daß dabei wiederum eine Fehlersituation eintreten kann. In einer Montagedomäne kann zwischen zwei Fehlerklassen unterschieden werden:

- *Systemfehler*: Darin sind die Fehler zusammengefaßt, die durch die Fehlertoleranz des Robotersystems verursacht wurden [Trevelyan & Nelson 87]. Ein Beispiel dafür ist eine nicht funktionierende Kamera, weil die computergesteuerte Lichtquelle ausgefallen ist.
- *Inkonsistenzfehler*: Sie haben ihren Ursprung in der Inkonsistenz des Weltmodells [Srinivas 77]. So ein Fehler könnte beispielsweise aus dem Befehl resultieren, ein unbewegliches Objekt zu greifen.

Die Aufgabe der Erklärungskomponente soll unter anderem darin bestehen, dem Menschen, der die Situation visuell beobachtet, eine in natürlicher Sprache formulierte *Tätigkeitsbeschreibung* dessen zu geben, was der Roboter gerade tut. Dazu müssen die Befehle eines Anweisungsplans in natürliche Sprache übersetzt werden. Als Grundlage dienen dabei die Umwelt- und Planschrittinformationen. Wie oben angedeutet, kann es bei der Planabarbeitung durch den Roboter zu einer Fehlersituation kommen. Autonome Roboter sind in der Lage, die meisten Fehler ohne fremde Hilfe zu korrigieren. Für den Fall, daß das Robotersystem den Fehler nicht selbst beheben kann und um Mißverständnisse auf Benutzerseite zu vermeiden, werden *Fehler*- und *Tätigkeitsbeschreibungen* generiert. Sie sind auch für den Fall, daß der Mensch dem Roboter weiterhelfen soll, von Nutzen. Da die Erklärung einzelner Roboterbefehle im Fehlerfall möglicherweise zu weiteren Unklarheiten führen kann, sollen zusammenfassende Beschreibungen von mehreren Roboterbefehlen, sogenannte *Verlaufsbeschreibungen*, möglich sein. Zusätzlich soll das Erklärungsmodul die Eigenschaft haben, Antworten auf mögliche Fragen hinsichtlich der *Fehlerlösung* und der *Fehlerursache* zu generieren.

Da es allgemein während Tätigkeitsbeschreibungen zu Unklarheiten bezüglich der Position von Umweltobjekten kommen kann, muß jederzeit die Möglichkeit bestehen, der Erklärungskomponente (Lokalisations-)Fragen diesbezüglich zu stellen.

3.2.2 Wie soll erklärt werden?

Diese Problemstellung läßt sich auf die Frage nach der Definition der Schnittstelle zwischen dem Erklärungssystem und dem Menschen zurückführen. Der dabei zur Erklärung benutzte Roboter ist ein fiktiver, plangesteuerter autonomer Serviceroboter, der ähnlich wie KAMRO in einer Montageumgebung Objekte manipulieren kann. Die Kriterien, die diese Schnittstelle erfüllen soll, sind im folgenden aufgelistet.

- *Einfache Benutzerführung*: Alle Einstellungen, die auf die Erläuterung der Robotertätigkeit Einfluß nehmen, sollen auf einem möglichst einfachen Weg getätigt werden können. Dazu bietet sich zum einen die Möglichkeit der Konfiguration des Erklärungsmoduls über die natürliche Sprache, zum anderen die Konfiguration über einen kontextsensitives Dialogfenster, wie es bei modernen Programmen üblich ist, an.
- *Geringe Informationskomplexität*: Im allgemeinen Fall darf die Komplexität der Erklärung ein gewisses Maß nicht überschreiten, da es dadurch Verständnisprobleme geben kann [Grice 75]. So kann zum Beispiel die Tätigkeitsbeschreibung des Befehls an den Roboter ein bestimmtes Objekt (hier ein Würfel) an der Position 118, 212, 0 auf dem Tisch zu greifen, mit einem zu hohen Informationsgehalt zu Verwirrungen führen, wie das folgende Beispiel zeigt:

  ```
  Ich hebe im Augenblick den Würfel hinter der Tasse auf dem
  Tisch an der Position 118 212 0 mit meiner rechten Hand auf.
  ```

  Eine für Menschen adäquate Erklärung derselben Tätigkeit zeigt folgendes Beispiel:

```
Ich hebe den Würfel auf.
```

- *Monitorfähigkeiten*: Im Gegensatz zu der Forderung, die Informationsdichte der Beschreibung von Roboteraktivitäten nicht über ein gewisses Maß steigen zu lassen, ist es in manchen Fällen notwendig, sich alle Details einer Robotertätigkeit zur Kontrolle anzeigen zu lassen.
- *Variable Fehlermeldungen*: Nicht in jedem Arbeitsumfeld ist die vollständige Fehlerbeschreibung notwendig. Wie auch bei der Tätigkeitsbeschreibung gefordert, soll die Fehlerbeschreibung völlig frei bezüglich ihres Informationsgehaltes einstellbar sein.
- *Satzvariationen*: Die Erklärungen sollten zwar bei gleicher Datengrundlage immer denselben Informationsgehalt haben, jedoch in ihrer Syntax unterschiedlich sein, da dies zur Auflockerung des Erklärungsprozesses dient. Folgende stilistische Methoden können dabei zur Auflockerung dienen:
  - Gebrauch von verschiedenen Verben für dieselbe Tätigkeit. So kann die Arbeit des Roboters, die durch einen Befehl, etwas zu greifen, ausgelöst wurde, zum Beispiel mit den Verben `packen`, `aufheben`, `greifen` und `nehmen` beschrieben werden.
  - Unterschiedliche Bezeichnungen für das Aktionsobjekt. Wie auch bei den Verben, kann das Objekt, mit dem der Roboter gerade arbeitet, verschiedene natürlichsprachliche Bezeichnungen haben. Beispielsweise kann der Stift (`spacer`) aus dem CAB auch mit `Nagel`, `Bolzen`, `Zapfen` oder `Stachel` bezeichnet werden.
  - Verwendung von Zeitadverbien, wie `jetzt`, `nun`, `gerade` oder ähnliche.
  - Umstellen der *Lokationsbeschreibungen* (siehe dazu auch *räumliche Beschreibung*, Kapitel 2.2) im Satz. Mit Lokationsbeschreibungen sind die Satzteile gemeint, die den Ort des Aktionsobjektes oder die Modalität, wie es an diesen Ort gekommen ist, beschreiben. So kann ein Hinlegen-Befehl auf unterschiedliche Art in natürlicher Sprache beschrieben werden:
  - `Ich stelle den Würfel mit meiner linken Hand an der Position 0 450 0 auf den Tisch.`
  - `Ich greife den Würfel vor der Tasse an der Position 0 450 0 mit meiner linken Hand.`
- *Erzeugung von Fließtext*: Der Erklärungsalgorithmus soll erkennen, wenn dasselbe Objekt Gegenstand der vorangegangenen und der aktuellen Beschreibung der Roboteraktivität ist. Die Zusammengehörigkeit der Tätigkeiten soll sich dann auch in der Satzkonstruktion widerspiegeln, so daß dieses Merkmal für den Benutzer sofort offensichtlich wird. Ein Beispiel dafür ist die Befehlssequenz, ein Objekt zu greifen und danach abzulegen:

```
Ich greife den Würfel.
```

```
Ich stelle gerade den Würfel auf den Tisch an die Position 0
450 0.
```

Anstatt die Befehle unabhängig voneinander in zwei Sätzen zu beschreiben, sollen sie durch das Bindewort `und` miteinander verknüpft werden. Dabei wird das zweite Objekt durch ein Personalpronomen ersetzt. Die Erklärung als Fließtext soll demnach lauten:

```
Ich greife den Würfel und lege ihn an die Position 0 450 0
auf den Tisch.
```

Eine weitere Anforderung an das Erklärungssystem ist die Kontextsensitivität, das heißt, bei Lokalisationsfragen oder Fragen bezüglich einer Fehlersituation soll die Informationen dazu nur solange verfügbar sein, wie der Roboter die Situation nicht verändert hat.

3.3 Ein erster Lösungsansatz

In diesem Kapitel wird ein einfacher Ansatz zur Lösung des Planerklärungsproblems vorgestellt. Dabei wird zur syntaktischen Untersuchung von Anweisungs- und Fehlerplänen die *musterorientierte Planschrittanalyse* verwendet. Dieses Verfahren unterscheidet die Planschritte anhand ihres individuellen syntaktischen Musters.

So kann mit einer Tabelle, die Muster mit unvollständigen Erklärungen assoziiert, genau festgelegt werden, wie der Befehl im Planschritt beschrieben werden muß, indem mit dessen Muster und der Tabelle ein Erklärungsformular bestimmt wird. Für die endgültige Tätigkeitsbeschreibung muß es noch vervollständigt werden. Dazu werden die identifizierten Teile des Musters durch eine weitere Tabelle mit natürlichsprachlichen Bezeichnungen assoziiert und in das Formular eingetragen. Der gesamte Beschreibungsprozeß wird in Abbildung 17 präsentiert.

Im weiteren Verlauf werden die Vor- und Nachteile dieser Methode erörtert und diskutiert. Die Erkenntnisse, die man durch die Analyse dieses Lösungsansatzes gewinnt, sind Teil der theoretischen Grundlage des Plan- und Umwelterklärungsmoduls ROPLEX.

3.3.1 Vom Planschritt zur natürlichsprachlichen Erklärung

Genaugenommen ist die schrittweise Erläuterung eines Roboterbefehlsplans ein Übersetzungsprozeß, in dem Aussagen über bestimmte Sachverhalte in natürliche Sprache umgewandelt werden. Jede dieser Roboterbefehle, aus denen sich der Anweisungsplan zusammensetzt, hat eine individuelle syntaktische Charakteristik, die als *Befehlsmuster* bezeichnet werden soll. Es liegt daher nahe, für alle möglichen Roboteranweisungen die Befehlsmuster anzusehen und für jedes dieser Befehlsmuster eine oder mehrere sogenannte *Satzmaske(n)* in natürlicher Sprache bereitzustellen und diese in einer *Befehlsmustertabelle* zu speichern.

Eine Satzmaske ist ein unvollständiger Satz, wobei an den Stellen, an denen sich eigentlich das Objekt oder die Lokationsangaben (siehe Abschnitt 3.2.2) befinden, ein

Platzhalter existiert. Dieser steht, je nach Satzlage, für Objekte im Nominativ, Dativ oder Akkusativ. In Abbildung 18 ist ein Beispiel einer Befehlsmustertabelle für fiktive Greifen- und Ablegen-Anweisungen eines Roboters gegeben. Die linke Spalte der Tabelle enthält zeilenweise die unterschiedlichen Befehlsmuster des jeweiligen Roboters. In der einem Befehlsmuster zugehörigen Zeile der rechten Spalte sind die für die natürlichsprachliche Erklärung passenden Satzmasken enthalten. Beispielsweise kann für die Beschreibung des Befehls, mit der rechten Hand einen Würfel zu greifen (`befehl(greifen, r, würfel)`), aus der Befehlsmustertabelle die Satzmaske `Ich greife <OBJ-AKK> mit <OBJ-DAT>.` ausgewählt werden.

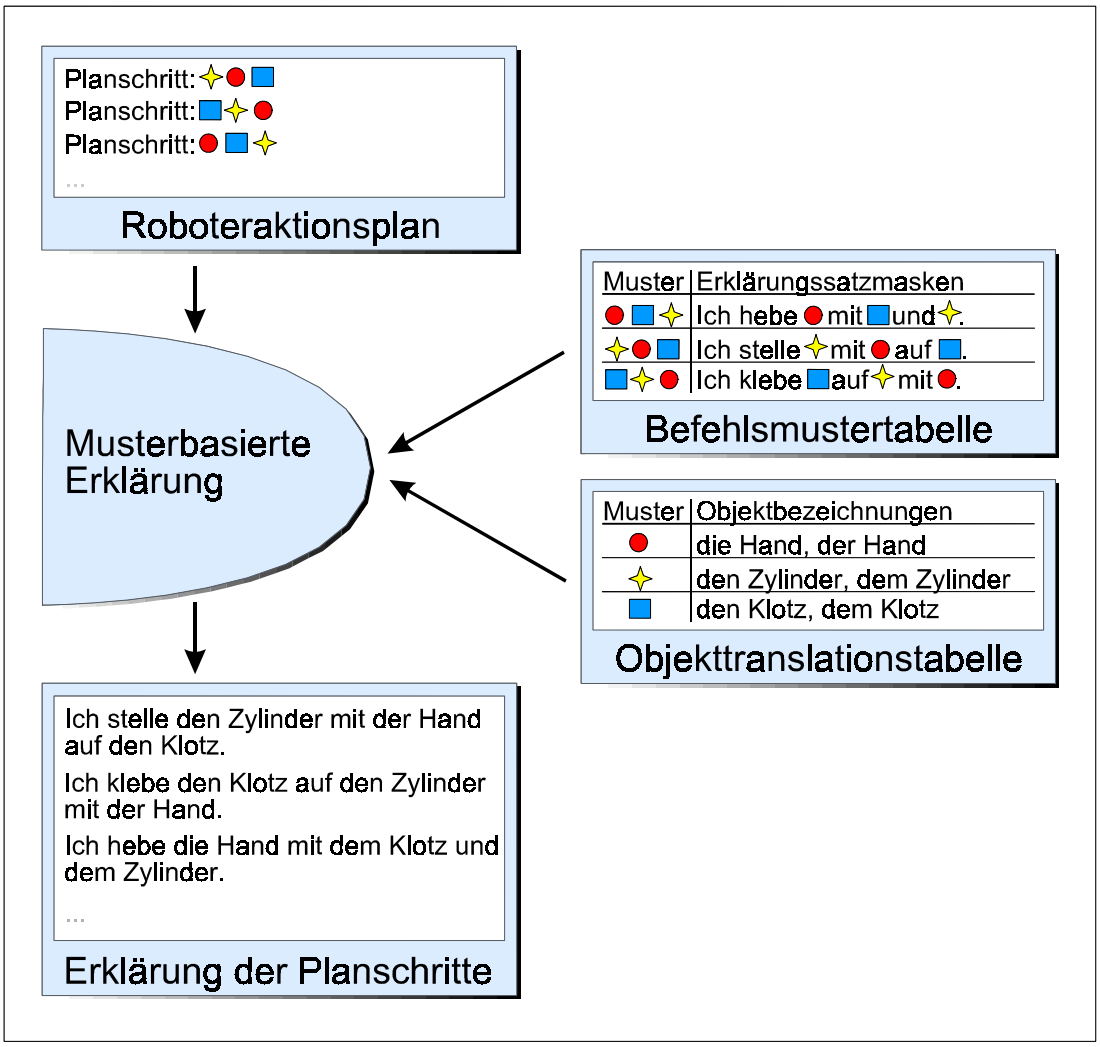

Abbildung 17: Einfacher Ansatz einer Planschritterklärungskomponente..

Damit eine vollständige natürlichsprachliche Erklärung eines Planschrittes erzeugt werden kann, muß diese Satzmaske um die natürlichsprachlichen Pendants der erklärungsrelevanten Variablenwerte[24] des zugrundeliegenden Roboterbefehls ergänzt werden. Dafür braucht diese Methode eine *Objekttranslationstabelle* für die Variablen des Roboterbefehls. Sie enthält die natürlichsprachlichen Objektbezeichnungen der Umweltobjekte, mit denen der Roboter hantiert, und die natürlichsprachlichen Bezeichnungen der Teile, mit denen der Roboter die Umwelt manipuliert. Zusätzlich beinhaltet

[24] Es handelt sich hier um Objekt-, Manipulator- und Positionsangaben des Roboterbefehls.

diese Tabelle kasusabhängige Daten der Objekte, wie zum Beispiel der Artikel, die für die Komplettierung der Satzmaske gebraucht werden.

Befehlsmuster	Satzmasken
befehl (greifen, ..., ...)	Ich greife <OBJ-AKK> mit <OBJ-DAT>. Ich hebe <OBJ-AKK> mit <OBJ-DAT> auf.
befehl (ablegen, ..., ..., [...], ...)	Ich stelle <OBJ-AKK> mit <OBJ-DAT> auf <OBJ-AKK> an die Position <POS>. Ich lege <OBJ-AKK> auf <OBJ-AKK> an die Position <POS> mit <OBJ-DAT>.

Abbildung 18: Beispiel einer Befehlsmustertabelle.

Ein Beispiel für eine Objekttranslationstabelle der musterorientierten Planerklärung zeigt Abbildung 19. Hierin beinhaltet die linke Spalte Objekte eines fiktiven Szenarios und die Bezeichnung der Manipulatoren eines Roboters. Die korrespondierende Zeile in der rechten Spalte der Tabelle enthält im Falle, daß es sich um ein Objekt der Umwelt handelt, eine natürlichsprachliche Bezeichnung sowie die dem Objekt zugehörigen Artikel für den Nominativ, Dativ und Akkusativ. Der Genitiv ist bei dieser Form der Erklärung nicht notwendig; er würde nur in Erklärungen gebraucht, in denen die Zugehörigkeit eines Objektes zu einem anderen ausgedrückt werden muß, wie zum Beispiel: „`Der Henkel der Tasse` ...“. Handelt es sich hingegen um einen Manipulator des Roboters, so steht in der zugehörigen Zeile der rechten Spalte der in natürlicher Sprache formulierte Teilsatz, der dieses Gefüge beschreibt.

Szenario-Objekt	Natürlichsprachliche Objektinformationen
banane-1	Zylinder, der, dem, den
würfel-1	Würfel, der, dem, den
tisch-1	Tisch, der, dem, den
r	der rechten Hand
l	der linken Hand

Abbildung 19: Beispiel einer Objekttranslationstabelle.

Der eigentliche Ablauf der Planschritterklärung ist denkbar einfach und vollzieht sich in drei Etappen:

1) Gleichzeitig mit der Roboteraktion, die der Planschritt auslöst, wird dieser von der Erklärungskomponente untersucht. Das musterorientierte Plananalyseverfahren sucht mit dem momentanen Befehlsmuster in der Befehlsmustertabelle nach einer Satzmaske. Existieren davon mehrere, so wird per Zufall eine ausgewählt.

2) In diese Satzmaske werden dann an die entsprechenden Stellen die jeweiligen natürlichsprachlichen Gegenstücke der Variablen des aktuellen Roboterbefehls, die der Algorithmus der Objektranslationstabelle entnimmt, gefüllt. Da diese Erklärungen immer ähnlich aufgebaut sind, ist genau definiert, in welchem Kasus das natürlichsprachliche Pendant der Roboterbefehlsvariable steht.

3) Die vollständig gefüllte Satzmaske ist die natürlichsprachliche Erklärung des aktuellen Planschrittes und kann dem Menschen zugeführt werden.

3.3.2 Beispiel einer Planschritterklärung

Ein fiktiver Anweisungsplan für einen Roboter, in dem er einen Würfel greifen und danach wieder ablegen soll, wird durch die beiden folgenden Anweisungen realisiert, die in der Syntax verfaßt sind, die in der Befehlsmustertabelle von Abbildung 17 verwendet wurde:

`befehl (greifen, r, würfel-1)`

`befehl (ablegen, r, würfel-1, [100 100 0], tisch-1)`

Da die Abarbeitung des Anweisungsplans sequentiell geschieht, wird zuerst der Greifen-Befehl abgearbeitet. Zeitgleich dazu sucht die Erklärungskomponente in der Satzmustertabelle eine passende Maske. In diesem Fall kann zwischen den beiden Satzmasken

`Ich greife <OBJ-AKK> mit <MANI-DAT>.` und

`Ich hebe <OBJ-AKK> mit <MANI-DAT> auf.`

entschieden werden. Der nächste Schritt ist die Ergänzung der Satzmaske durch die natürlichsprachlichen Gegenstücke des Objektes und des Manipulators. Diese Informationen findet der Planerklärungsalgorithmus in der Objekttranslationstabelle. Für den `würfel-1` entnimmt er die entsprechende natürlichsprachliche Information (`Würfel, den, der, der`). Da der Kasus des Objektes in der zufällig ausgewählten Satzmaske `Ich greife <OBJ-AKK> mit <MANI-DAT>` der Akkusativ ist, wird die Objektphrase `den Würfel` an die Stelle `<OBJ-AKK>` der ausgewählten Satzmaske eingefügt. Die in natürlicher Sprache formulierte Angabe der Manipulatorinformation `r` ist nach der Objekttranslationstabelle die Objektphrase `der rechten Hand`, die der Algorithmus an die Stelle `<MANI-DAT>` der noch unvollständigen Satzmaske einträgt. Die erzeugte Erklärung für den Greifen-Befehl ist demnach:

`Ich greife den Würfel mit der rechten Hand.`

Die sich nun anschließende Erklärung für den im Plan nachfolgenden Ablegen-Befehl wird nach dem gleichen Schema generiert, nur daß zusätzlich die Positionsangabe und das natürlichsprachliche Pendant des Referenzobjektes in eine passende Satzmaske eingetragen werden. Eine mögliche, in natürlicher Sprache formulierte Erklärung für den Ablegen-Befehl lautet demzufolge:

`Ich lege den Würfel mit der rechten Hand auf den Tisch an die Position 100 100 0.`

3.3.3 Diskussion

Die Vorzüge der musterorientierten Planerklärungsmethode bestehen hauptsächlich in der Schnelligkeit und Flexibilität. Da diesem Verfahren einfachste Datenstrukturen und Suchalgorithmen zugrunde liegen, kann man leicht durch Verwenden von Methoden wie Hashing, Caching oder Indizierung höhere Geschwindigkeiten bei der Erzeugung der natürlichsprachlichen Erklärung erreichen.

Betrachtet man die für diese Planerklärungsmethode benutzten Datenstrukturen, so stellt man fest, daß sie sehr leicht zu überschauen sind. Dies hat automatisch zur Folge, daß die Wartung sowie die Konfiguration des Erklärungsalgorithmus´ einfach ist. Durch das Anpassen der Befehlsmuster an eine andere Befehlssyntax ist es möglich, innerhalb kürzester Zeit Planschritte anderer Roboter zu analysieren und natürlichsprachliche Erklärungen dafür zu generieren. Wie auch die Anpassung an andere syntaktische Gegebenheiten von Roboterplanschritten, ist auch die Erzeugung von einfach aufgebauten Erklärungen in anderen Sprachen, wie zum Beispiel Englisch oder Französisch, leicht realisierbar. Dazu müssen lediglich die Satzmasken in die jeweilige Sprache übersetzt und die Objekttranslationstabelle angepaßt werden.

Der wohl auffälligste Nachteil der musterorientierten Planerklärung ist die Tatsache, daß sie sich für die Erzeugung der Erklärungen in natürlicher Sprache nur der Informationen annimmt, die der aktuelle Planschritt beinhaltet. Dies kann bei einigen Robotersystemen, so auch KAMRO, Nachteile mit sich bringen. Wie man aus Unterabschnitt 2.4.2.3 entnehmen kann, fehlt bei der **`pick`**-Anweisung die Positionsangabe, an der das Objekt gegriffen wird. Mit dieser Wissenslücke ist es jedoch unmöglich, Erklärungen zu erzeugen, in denen die Information der Position des zu greifenden Objektes enthalten ist. Darüber hinaus hat dieses System auch keine Informationen über die anderen Umweltobjekte. Dadurch kann kein Kontextwissen aufgebaut werden, was wiederum zur Folge hat, daß in Erklärungen keine Lokationsbeschreibungen enthalten sein können. Dies ist gerade bei Objekten, die in der „Roboterwelt“ häufig anzutreffen sind, eine große Unzulänglichkeit. Stellt man sich beispielsweise ein Szenario mit Objekten des Cranfield-Assembly-Benchmarks vor, so stellt man fest, daß es dort vier Bolzen (`spacer:1` – `spacer:4`) gibt. Die durch das musterorientierte Planerklärungsverfahren generierte Erläuterung eines Greifen-Befehls könnte folgendermaßen aussehen:

`Ich hebe den Bolzen mit meiner rechten Hand auf.`

Für die Erklärung weitaus sinnvoller und auch verständlicher ist folgende Beschreibung derselben Anweisung:

`Ich hebe den Bolzen vor der Seitenplatte mit meiner rechten Hand auf.`

Durch die Erweiterung der Lokationsbeschreibung `vor der Seitenplatte` wird der visuelle Fokus des Beobachters an die Stelle gerückt, an der die Aktion stattfindet. Dadurch wird der Bolzen genau identifiziert. Auch bei der Frage, wo sich ein bestimmtes Objekt befindet, ist es notwendig, daß die Informationen der anderen Szenarioobjekte dem Erklärungsalgorithmus vorliegen, da es auch in diesem Fall verständlicher ist, den

Aufenthalt des gesuchten Objektes mit Umgebungsinformation und nicht nur mit reinen Positionsangaben zu beschreiben.

Aus diesen Nachteilen ergibt sich die Notwendigkeit, die oben beschriebene musterorientierte Planerklärungsmethode um eine Wissensbasis zu erweitern, in der die geforderten Informationen für alle Umweltobjekte enthalten sind. Darin sollten auch, wie in der Objekttranslationstabelle, die natürlichsprachlichen Bezeichnungen der Umweltobjekte, sowie deren Ergänzungen im Nominativ, Dativ und Akkusativ enthalten sein. Wie sich herausgestellt hat, muß diese Wissensstruktur auch die Information über die jeweilige Objektposition gespeichert haben, da es durchaus Planschritte von Robotern geben kann, in denen keine Positionsangabe bezüglich des Handlungsobjektes gemacht wurde, diese jedoch für eine präzisere natürlichsprachliche Erklärung der Robotertätigkeit benötigt wird.

Eine weitere Designschwäche des musterorientierten Planerklärungsverfahrens ist die Datenstruktur *Satzmaske*, die als Grundlage der natürlichsprachlichen Erklärungen dient. Die Bedingungen, speziell die der Satzvariation, die an die Robotertätigkeitsbeschreibungen in natürlicher Sprache (siehe Abschnitt 3.2.2) gemacht worden sind, können nur durch weitgehend redundante zusätzliche Einträge in die Befehlsmustertabelle erzeugt werden. Schon bei einer nicht allzu großen Menge von Roboterbefehlen leidet die Übersichtlichkeit in dieser Wissensstruktur. Auch bei der Erzeugung von natürlichsprachlichen Erklärungen der Roboteraktionen mit vermindertem oder erhöhtem Informationsgehalt muß dieses Verfahren auf zusätzliche Einträge in der Befehlsmustertabelle zurückgreifen. Zudem existieren keine Satzmasken für Antworten auf Fragen bezüglich der Objektposition (Lokalisationsfragen) oder des Fehlers. Einen ganz besonderer Fall stellt die Erzeugung von Erklärungsfließtext dar. Hier wird zum einen das Personalpronomen des Handlungsobjektes gebraucht, zum anderen eine weitere Form der Satzmaske, die mit der Verknüpfung `und` beginnt.

Aufgrund der Tatsache, daß durch diese Anforderungen die Kapazitäten der Befehlsmustertabelle ausgeschöpft werden, ist es notwendig, eine Datenstruktur zu entwickeln, die diesen Ansprüchen gerecht wird und darüber hinaus handhabbar bleibt. Zudem ist es sinnvoller, das Fehlerwissen vom Planerklärungswissen getrennt zu verwalten, da es sich bei den aus dem Fehlerwissen erzeugten Erklärungen um Antworten handelt, die dem Benutzer in der gesamten Fehlerkorrekturzeit zur Verfügung stehen sollten und demnach dem System länger bekannt sein müssen. Demgegenüber steht die Erklärung des Planschritts, die zeitgleich zur Roboteraktion erzeugt wird und die nach relativ kurzer Zeit nicht mehr benötigt wird.

Abschließend läßt sich sagen, daß der musterorientierten Planerklärungsmethode wichtige Wissensbasen fehlen und sie hinsichtlich der Anforderungen an eine Planerklärungskomponente, wie sie das Kapitel 3.2 beschreibt, ungenügende Datenstrukturen besitzt. Es hat sich herausgestellt, daß folgendes Wissen unbedingt notwendig ist:

- *Objektwissen*: Darin sind alle objektbezogenen Angaben zu finden.
- *Fehlerwissen*: Diese Kenntnisse bestehen aus den Angaben, was, beziehungsweise wodurch der Fehler aufgetreten ist, sowie aus einem möglichen Lösungsweg.

- *Kontextwissen*: Dadurch können Lokationsbeschreibungen erzeugt werden, die unter anderem für die Beantwortung von Fragen nach dem Ort und in Erklärungen gebraucht werden.
- *Grammatikwissen*: Diese Datenstruktur stellt das Kernwissen dar, in dem alle für die natürlichsprachliche Erklärung von Roboterplanschritten notwendigen Informationen enthalten sind.

Darüber hinaus sollten die Wissensbasen so entworfen sein, daß deren Wissen auch von anderen Modulen des Dialogsystems, wie zum Beispiel RIACS, genutzt werden kann.

4 Realisierung und Implementation

Damit eine wechselseitige Kommunikation in einer natürlichen Sprache zwischen Mensch und Maschine möglich wird, müssen die Äußerungen eines jeden Kommunikationspartners vom anderen verstanden werden. Da weder der Mensch noch die Maschine die „Muttersprache“[25] des anderen direkt versteht, muß ein Übersetzungsverfahren existieren, das es einerseits ermöglicht, daß sich die Maschine dem Menschen mitteilen kann und das anderseits dem Menschen die Möglichkeit gibt, mit der Maschine zu kommunizieren.

Beide Richtungen der Verständigung sind durch die Dialogschnittstelle RONTRA realisiert. Sie ermöglicht der Maschine, Informationen über Sachverhalte aus ihrer Sicht dem Menschen in natürlicher Sprache mitzuteilen. Zudem kann der Mensch Befehle und Fragen in natürlicher Sprache an die Maschine richten.[26] In dieser Arbeit wird hauptsächlich die Kommunikationsrichtung von der Maschine zum Menschen untersucht beziehungsweise das Verfahren vorgestellt, das es der Maschine erlaubt, dem Menschen natürlichsprachliche Informationen über seine Tätigkeit zu geben.[27] Ein weiteres Gebiet der Kommunikation zwischen Mensch und Maschine, das in dieser Arbeit erörtert wird, ist die Behandlung von Fragen bezüglich der Umwelt oder der Robotertätigkeit. Von der konzeptuellen Ebene aus gesehen, werden die Fähigkeiten der Maschine einerseits um das natürlichsprachliche Beschreiben ihrer Handlung erweitert und andererseits um die Eigenschaft ergänzt, Antworten auf Fragen bezüglich Umwelt und Tätigkeit zu generieren. Dafür sind bestimmte syntaktische Gegebenheiten Voraussetzung. So ist es zum Beispiel notwendig, daß die Tätigkeiten, die von der Maschine ausgeführt werden, in einer syntaktischen Struktur, einem sogenannten Anweisungsplan, hintereinander aufgelistet sind.

Im vorangegangenen dritten Kapitel wurde eine einfache Form der schrittweisen Planerklärung vorgestellt. Anhand dieses Beispiels wurden Defizite dieser Erklärungsmethode und der verwendeten Datenstrukturen aufgedeckt. Die so gewonnenen Erkenntnisse dieser Analysen flossen in die Entwicklung von Datenstrukturen, die für einen flexibleren Einsatz im Bereich der natürlichsprachlichen Planschritterklärung konzipiert sind. Zusammen bilden sie die Grundlage einer Wissensbasis, die den in Kapitel

[25] Die Muttersprache bei Maschinen sei die Menge von Befehlen, die sie interpretieren und verstehen können.

[26] Diese Richtung der Kommunikation ist zur Zeit auf die Sprache Deutsch beschränkt.

[27] Dieses Verfahren ermöglicht es, Erklärungen in anderen Sprachen, wie zum Beispiel Englisch oder Französisch, zu erzeugen.

3.2 gestellten Anforderungen an eine Plan- und Umwelterklärungskomponente gerecht wird.

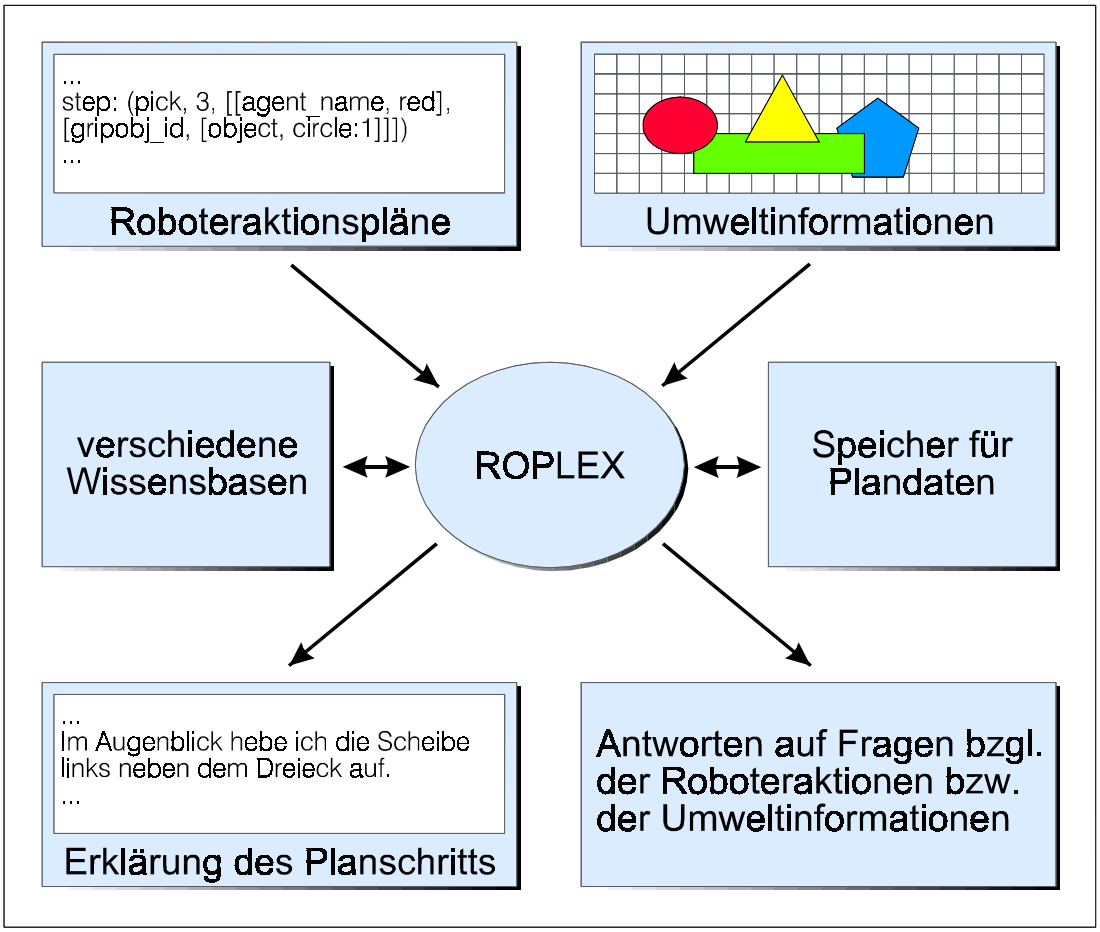

Abbildung 20: Arbeitsweise von ROPLEX.

In diesem Kapitel wird die Realisierung von ROPLEX (siehe Abbildung 20) beschrieben. Insbesondere wird dessen Planerklärungsverfahren sowie das Datenverarbeitungsmodell präsentiert. Das Plananalyseverfahren unterscheidet sich wesentlich von dem musterorientierten Analyseverfahren, das immer den gesamten Planschritt als Suchmaske für erklärungsrelevante Informationen verwendet. Es wird eine weitaus anpassungsfähigere Untersuchungsmethode, die *sequenzenorientierte Planschrittanalyse* (siehe Abschnitt 4.5.2) benutzt, um Informationen zu extrahieren. Diese werden dabei mit Schlüsselzeichen identifiziert.

Neben der natürlichsprachlichen Planbeschreibung hat ROPLEX die Aufgabe, aus Umweltinformationen der Maschine Antworten auf Fragen über Ort und Lage von Umweltobjekten zu generieren. Zudem können, sofern Fehlerdiagnoseinformationen vorliegen, natürlichsprachliche Antworten auf Anfragen über Ursache und Hergang einer Fehlersituation generiert werden. Allgemein kann der Informationsgehalt von Erklärungen durch Konfigurationsbefehle in natürlicher Sprache auf die jeweiligen Bedürfnisse des Benutzers eingestellt werden. Die dafür notwendigen Methoden und Wissensbasen werden im weiteren Verlauf dieses Kapitels vorgestellt.

4.1 Schnittstelle zu Robotern

Damit Robotertätigkeiten und Objektlokationen überhaupt in natürlicher Sprache beschrieben werden können, muß es eine Schnittstelle geben, über die das Robotersystem relevante Informationen an ROPLEX schicken kann. Für die Erzeugung von Beschreibungen in natürlicher Sprache muß der Roboter die folgenden drei unterschiedlichen Informationsarten bereitstellen können:

- *Umwelt- oder Szenariodaten*: Sie repräsentieren die Weltvorstellung des Roboters. Für die Generierung von natürlichsprachlichen Erklärungen sind vor allem die Informationen der einzelnen Objekte und deren Eigenschaften wie Koordinaten, Farbe, Form und so weiter zwecks Aktualisierung und Konsistenzhaltung der internen Szenariorepräsentation interessant.
- *Pläne*: Darin sind die Aufgaben repräsentiert, die der Roboter erledigen soll.
- *Statusinformationen*: Sie repräsentieren den aktuellen Zustand, in dem sich der Roboter gerade befindet. Unter anderem wird hier die Information über den jeweiligen vom Roboter auszuführenden Planschritt erwartet.

Die Definitionen der einzelnen Schnittstellenbereiche werden in den folgenden Abschnitten gegeben.

4.1.1 Szenariodaten

Damit sich ein Roboter die Welt „vorstellen“ und darin agieren kann, muß er auf irgendeine Art seine Umwelt erfassen. Die so gesammelten und verarbeiteten Informationen über die Gegenstände der Umwelt sind die *Szenariodaten* eines Roboters.

Je nach Robotertyp fallen diese mehr oder weniger umfangreich aus. Für die Generierung von natürlichsprachlichen Beschreibungen der Welt oder von Roboteraktionen sind nicht alle Informationen über das Szenario notwendig. Welche davon ROPLEX vom Roboter fordert und in welchem Format sie vorliegen müssen, ist im folgenden beschrieben:

- *Objektname* (**`Oname`**): Er ist die exakte Bezeichnung des Objektes. Sie muß jedoch nicht zwangsläufig im direkten Bezug zur wirklichen Bedeutung des realen Objektes stehen. Der Objektname ist der Referent für die Menge von natürlichsprachlichen Bezeichnungen, die für den Gegenstand in Frage kommen. Er kann aus beliebigen, aber endlich vielen, Zeichen bestehen.
- *Objektlage* (**`Opos`**): Die hier gemachte Aussage gibt die Art an, wie der Gegenstand in der Welt steht oder liegt. Die Objektlage kann aus beliebig, aber endlich vielen, Zeichen bestehen.
- *Objektkoordinaten* (**`Ocoord`**): Es werden zwei- und dreidimensionale Koordinatenangaben ohne Maßeinheit verstanden, die folgendes Format einhalten müssen: `[x, y, z]`, mit `x, y, z` ∈ `[-65536; 65536]`. Die Angabe der `z`-Koordinate ist nicht

notwendig. Wichtig hierbei ist, daß zwischen verschiedenen Ausgangspunkten zur Berechnung der Position unterschieden wird. Im allgemeinen Fall ist die Positionsangabe jedoch relativ zum Roboter.

- *Objektdrehung* (**Orot**): Sie gibt die Drehung des Objektes bezüglich der „Nullstellung“ an. Das Format dieser Angabe muß folgender Bedingung gehorchen: `Orot` ∈ [`-360`; `360`]

Eine vollständige *Objektbeschreibung* muß aus den gerade vorgestellten Informationseinheiten bestehen. Das Format, in dem sie angeordnet sein müssen, wird in Abbildung 21 beschrieben. Die Variablen **S** und **T** stehen dabei für mögliche Kommentare.

<S> <Oname> <Opos> <Orot> <Ocoord> <T>, wobei T, S beliebige Zeichenketten (ASCII-Zeichen) sein können.

Abbildung 21: Format der Objektbeschreibung.

Ein Robotersystem kann seine kompletten Szenariodaten mitteilen, indem es alle Objekte des Szenarios in obigem Format in einer dafür vorgesehenen Datei zeilenweise ablegt. Falls es nicht möglich ist, die Lage (**Opos**) oder Drehung (**Orot**) in Objektbeschreibungen anzugeben, müssen definierte Platzhalterwerte im jeweiligen Format verwendet werden. Sie können später in Erklärungen ausgeblendet werden. Alle anderen Angaben sind für eine sinnvolle Arbeitsweise unbedingt notwendig.

4.1.2 Plan- und Roboteranweisungssyntax

Die Beschreibung von Robotertätigkeiten basiert primär auf den Informationen, die in Anweisungsplänen gegeben werden. Genauer betrachtet sind das Planschritte, in denen Befehle an den Roboter modelliert sind. Damit sie identifiziert werden können, muß zuerst die übergeordnete syntaktische Planstruktur erkannt werden. In welchem Format sie vorliegen müssen, ist in Abbildung 22 beschrieben. Zur Übersichtlichkeit wurde die akzeptierte Sprache (Menge) $\mathcal{L}_0$ von Plänen mittels regulärer Ausdrücke [Lewis & Papadimitriou 81] beschrieben. Die endliche Menge $\mathcal{Z}$ ist dabei so definiert, daß sie alle für die Planerzeugung benötigten Zeichen beinhaltet. Planschritte müssen, damit sie später erkannt werden können, mit einer charakteristischen Zeichenfolge, dem *Anweisungsidentifizierer* (**Aid**) beginnen. Die Sprache $\mathcal{P}_3$ enthält alle potentiellen Planschritte. Verlaufsbeschreibungen von mehreren Robotertätigkeiten werden aus codierten Kurzbeschreibungen generiert, die im Plan an geeigneter Stelle hinterlegt werden können. Sie müssen zu Beginn das Schlüsselzeichen `%` aufweisen. Alle möglichen Kurzbeschreibungen sind in der Sprache $\mathcal{P}_2$ zusammengefaßt. Die Sprachen $\mathcal{P}_0$ und $\mathcal{P}_1$ stehen für mögliche Kommentare, die nicht weiter von der Analysekomponente betrachtet werden, aber durchaus verwendet werden können. Das sich anschließende Beispiel gibt einen Plan wieder, der zur Sprache der akzeptierten Pläne gehört. Die verwendete Zeichenfolge für den Anweisungsidentifizierer ist **ROBCOM**. Die Zeilenangaben auf der linken Seite sind kein Bestandteil des Plans. Sie werden später zur weiteren Erklärung benötigt.

```
1  ;Dies ist Testplan für den fiktiven Roboter Robby.
2  ;Die Roboterbefehle sind alle erfunden
3  ;
4  ; Der folgende Befehl veranlaßt Robby in das Malstudio zu fahren.
5  ROBCOM FAHREN IN STUDIO 1
6  ;
7  % malenRotKreuz
8  ROBCOM GREIFEN PINSEL
9  ROBCOM TAUCHEN PINSEL IN FARBE ROT
10 ROBCOM MALEN HORIZONTAL STRICH 6 CM START 4, 10
11 ROBCOM MALEN VERTIKAL STRICH 6 CM START 7, 4
12 ;Jetzt mal was anderes % anschaltenMusik
13 Test, ob das klappt:  ROBCOM DRÜCKEN MUSIKSCHALTER 1
14 ;
15 ;So das war's mit der Demonstration
```

Die Zeilen 1 bis 4, 6, 14 und 15 bestehen aus Kommentar[28] und sind für die Erzeugung von Erklärungen irrelevant. Sie sind durch die Sprache $\mathcal{P}_0$ in $\mathcal{L}_0$ repräsentiert. Planschritte sind in den Zeilen 5 und 8 bis 11 im Beispiel anzutreffen. In Zeile 13 steht zwar auch ein Planschritt, davor steht jedoch ein Kommentar. Der Inhalt dieser Zeilen sind Wörter der Sprache, die sich aus der Verknüpfung der Sprachen $\mathcal{P}_1$ und $\mathcal{P}_3$ ergeben. Kurzbeschreibungen findet man in den Zeilen 7 und 12. Diese Wörter sind Elemente der Sprache, die sich aus der Verknüpfung der Sprachen $\mathcal{P}_1$ und $\mathcal{P}_2$ ergeben.

$\mathcal{Z}$ = Alphabet, das alle Zeichen enthält, die für die Planerzeugung notwendig sind.
Definitionen:

$\mathcal{Z}_0 := \{\omega \in \mathcal{Z}^* \mid \omega$ enthält das Teilwort **<Aid>** und das Zeichen **%** nicht$\}$

$\mathcal{Z}_1 := \{\omega \in \mathcal{Z}^* \mid \omega$ endet mit Zeilenumbruchzeichen ↵ und enthält sonst kein ↵$\}$

$\mathcal{P}_0 := \{\omega \in \mathcal{Z}_0 \mid \omega$ endet mit ↵$\}$

$\mathcal{P}_1 := \{\omega \in \mathcal{Z}_0 \mid \omega$ enthält kein ↵$\}$

$\mathcal{P}_2 := \{\omega \in \mathcal{Z}_1 \mid \omega$ beginnt mit **%** und enthält sonst kein **%** und **<Aid>**$\}$

$\mathcal{P}_3 := \{\omega \in \mathcal{Z}_1 \mid \omega$ beginnt mit dem Teilwort **<Aid>** und enthält sonst kein **%**, **<Aid>**$\}$

$$\mathcal{L}_0 := \{\omega \in (\mathcal{P}_0 \cup \mathcal{P}_1\mathcal{P}_2 \cup \mathcal{P}_1\mathcal{P}_3)^* \mid \omega \text{ ist ein Plan}\}$$

Abbildung 22: Akzeptierte Sprache von Plänen.

Der genaue Aufbau der akzeptierten Planschritte wird anschließend erörtert. Die in $\mathcal{L}_0$ verwendete Sprache $\mathcal{P}_3$ ist die Obermenge der tatsächlich identifizierbaren Befehle. Das Format der *Planschrittdaten*, die für Tätigkeitsbeschreibungen relevant sind, wird im folgenden vorgestellt:

[28] Mit Kommentar eine Zeichenfolge gemeint, die nicht mit dem Zeichen **%** oder dem Teilwort **Aid** beginnt oder sie enthält.

- *Anweisungsidentifizierer* (**Aid**): Er ist eine charakteristische Zeichenfolge, anhand derer ein Planschritt identifiziert wird.
- *Befehlsname*: Er kann bis zu zwei unterschiedlichen, beliebig langen, endlichen Zeichenfolgen (mit **Rcmd1**, **Rcmd2** bezeichnet) bestehen, die willkürlich in einem Planschritt angeordnet sind. Im vorangegangenen Beispiel sind die Befehlsnamen des Planschritts in Zeile zehn MALEN und HORIZONTAL. Durch sie ist eine Repräsentation der Verben, die in der Erklärung verwendet werden können, möglich. Befehlsnamen stehen also für eine Klasse von Verben, welche die Tätigkeit des Roboters ausdrücken.
- *Befehlsausführender Mechanismus* (**Rmani**): Hier ist der Teil des Roboters angegeben (zum Beispiel Arm, Fahrwerk oder Kamera), der die Tätigkeit ausführt. Die Information über den befehlsausführenden Mechanismus kann in einer beliebig langen, endlichen Zeichenfolge untergebracht werden.
- *Aktionsobjekt* (**Aobj**): Es repräsentiert die Objektinstanz, die manipuliert werden soll. Sie kann auf die Klasse von natürlichsprachlichen Objektbezeichnungen, die für die Objektinstanz in Frage kommen, abgebildet werden. Das Aktionsobjekt, auch *Handlungsobjekt* genannt, kann, wie der befehlsausführende Mechanismus, aus einer beliebig langen aber endlichen Zeichenfolge bestehen.
- *Berührungsobjekt* (**Bobj**): Damit ist die Objektinstanz gemeint, die mit dem Aktionsobjekt durch irgendeine Aktion (zum Beispiel durch fügen oder zusammenstecken) in Berührung kommt. Das Berührungsobjekt kann, wie der befehlsausführende Mechanismus, aus einer beliebig langen, aber endlichen Zeichenfolge bestehen.
- *Lageinformation*: Darunter sind alle Informationen zusammengefaßt, die die *Umgebung* (**Umg**), die Position und die spezielle *Ausrichtung des Objektes* (**AOplc**) beschreiben. Außer der Positionsangabe können alle durch eine beliebig lange, aber endliche Zeichenfolge repräsentiert werden. Die *Positionsangabe des Aktionsobjektes* (**AOpos**) muß folgendem Format gehorchen: `[x, y, z]`, mit `x, y, z` $\in$ `[-65536; 65536]`.

Für die gerade vorgestellten Dateneinheiten, die zur Generierung von natürlichsprachlichen Tätigkeitsbeschreibungen gebraucht werden, verlangt das Planschrittanalyseverfahren keine expliziten Erkennungsmerkmale in Form von Zeichen oder Zeichenketten. Die minimalen Angaben, die für eine Tätigkeitsbeschreibung gebraucht werden, sind die des Anweisungsidentifizierers und des Befehlsnamens. Wie oben angedeutet, müssen Planschrittdaten in einer Zeile des Anweisungsplanes stehen. Die Reihenfolge der einzelnen Einheiten ist beliebig. Sie müssen lediglich durch beliebige, nicht leere aber endliche Zeichenketten getrennt werden. Die akzeptierte Sprache der Planschritte $\mathcal{L}_1$ wird in Abbildung 23 präsentiert.

$$\mathcal{L}_1 := \{\omega \in \mathcal{P}_3 \mid \omega \text{ enthält } \texttt{<Rcmd1>} \text{ oder } \texttt{<Rcmd2>}\}$$

Abbildung 23: Akzeptierte Sprache der Planschritte.

Die akzeptierten Roboterbefehle sind Wörter der Sprache $\mathcal{L}_2$. Sie besteht aus Elementen, die bis auf den führenden Anweisungsidentifizierer mit den Elementen der Sprache $\mathcal{L}_1$ übereinstimmen. Einzelne Roboteranweisungen können zum Beispiel bei der Diagnosemeldung von Fehlern auftreten (siehe nächstes Kapitel). Bezogen auf den voran gegangenen Anweisungsplan (Seite 63) läßt sich folgendes Beispiel für Wörter der Sprachen $\mathcal{L}_1$ beziehungsweise $\mathcal{L}_2$ konstruieren: Der Planschritt (**ROBCOM** `MALEN HORIZONTAL STRICH 6 CM START 4, 10`) aus Zeile 10 ist ein Wort der Sprache $\mathcal{L}_1$. Das Wort ohne den Anweisungsidentifizierer **ROBCOM**, also `MALEN HORIZONTAL STRICH 6 CM START 4, 10` ist ein Wort der Sprache $\mathcal{L}_2$.

4.1.3 Statusinformationen

Neben Szenariodaten und Plänen werden weitere Informationen benötigt, die bei jeder Veränderung in dem Roboterszenario neu in einer *Steuerdatei* abgelegt werden müssen. Diese wird, nachdem sie interpretiert wurde, zwecks Quittierung gelöscht. Darin können zum einen die Angaben des Anweisungsplans und des *Anweisungszeigers* (`plan ...`) stehen. Dieser gibt den Plannamen und die Planzeile der Roboteranweisung an, die gerade vom Roboter abgearbeitet wird. Zum anderen können an dieser Stelle Informationen über eingetretene Fehlersituationen, die sog. *Diagnoseinformationen* (`diag ...`) stehen. Den Inhalt und das Format der verschiedenen *Statusinformationen* werden in Abbildung 24 beschrieben.

```
1: plan <Planname>: <Schritt>
2: plan <Planname>: <Sektion>.<Schritt>
3: diag <Anweisung> <Fehler> <Ursache> <Unteranweisung> <Ziel>
```

Abbildung 24: Inhalte der Steuerdatei.

Der **Planname** dient dabei als Referenz auf eine gleichnamige Datei, in der der eigentliche Anweisungsplan zu finden ist. Die **Schritt**-Angabe repräsentiert die Zeile des Planschrittes, in der die Roboteranweisung steht, die momentan vom Roboter ausgeführt wird. Im Fehlerfall kann vor der Schrittangabe die **Sektion** des Planes stehen. Sie zeigt auf die Zeile im Plan, in der eine Verlaufsbeschreibung von Roboteranweisungen steht.

Der Inhalt der Felder **Anweisung** und **Unteranweisung** repräsentiert den Roboterbefehl, bei dem ein Fehler eingetreten ist. Da eine Anweisung aus mehreren Unteranweisungen bestehen kann, erlaubt die Diagnoseinformation die Angabe der Unteranweisung, bei der ein Fehler eingetreten ist. Zum Beispiel könnte der Befehl „`Greife eine Tasse!`“ aus den Unteranweisungen: „`Bewege die Hand zur Tasse!`“, „`Öffne die Hand!`“, „`Umschließe die Tasse mit der Hand!`“ und „`Hebe die Tasse an!`“ bestehen. Ihr Format muß mit dem der Roboteranweisungen (siehe Abschnitt 4.1.2) übereinstimmen.

Der Fehler und dessen Ursache kann als endliche Zeichenkette codiert in den entsprechenden Datenzellen **Fehler** und **Ursache** angegeben werden. Zudem besteht die Möglichkeit, eine codierte Angabe über die Vorgehensweise, wie der Fehler mögli-

cherweise behoben wird, anzugeben. Diese Information kann in der Datenzelle **`Ziel`** als endliche Zeichenfolge angegeben werden.

Ist das Robotersystem nicht in der Lage, Aussagen über den Fehler oder dessen Ursache zu generieren, so muß für die fehlende Information das korrespondierende Feld mit definierte Platzhalterwerten belegt werden, damit das Format der Datenstruktur gewahrt bleibt. Sie können später in Erklärungen ausgeblendet werden. Genauso verhält es sich für andere fehlende Angaben.

4.2 Benutzerschnittstelle

Dieser Abschnitt widmet sich dem Satzaufbau der natürlichsprachlichen Erklärungen. Es werden die Satzformen vorgestellt, die notwendig sind, um die Tätigkeit eines Roboters in einem abgegrenzten Szenario und dieses selber beschreiben zu können.

Die Erklärungen von Tätigkeiten des Roboters haben einen *Sachaspekt*, das heißt sie unterliegen Sachzwängen und haben einem *Situationsaspekt*, der sich aus dem Sprecher, der etwas an einem Ort zu einer bestimmten Zeit tut, zusammensetzt ([Eichler & Bünting 96], Seite 22). Der Sachaspekt der *Tätigkeitsbeschreibung* ist durch das Szenario bestimmt, in dem der Roboter handeln. Die von ROPLEX erzeugte natürlichsprachliche Beschreibung der Roboteraktion soll den Sach- und Situationsaspekt der Tätigkeit ausdrücken.

Zur Lösung dieses Problems können nicht alle möglichen *Satzformen* verwendet werden, da den natürlichsprachlichen Erklärungen von Roboteraktionen zeitliche und ressourcenbedingte[29] Grenzen gesetzt sind. Eine Einschränkung ist die Festlegung des Subjektes auf das Wort *Ich*. Der Roboter fungiert somit als Sprecher, der selbst seine Handlungen beschreibt. Bevor konkrete Aussagen über den Aufbau und die Gliederung des Prädikats gemacht werden können, ist zu überlegen, welche Informationen darin enthalten sein sollen, siehe dazu Kapitel 3.2. Der *Prädikatsverband* beinhaltet die möglichen Satzteile, wie zum Beispiel, Dativ-, Akkusativobjekte oder Präpositionen.

Neben den benötigten Satzformen für die Tätigkeitsbeschreibungen sind weitere notwendig, um ein breites Band von Fehlerbeschreibungen und Antworten erzeugen zu können. Generell werden für Antworten keine umfangreichen Satzformen benötigt. Für Fehlerbeschreibungen werden Satzmasken (siehe Abschnitte 3.3.1 und 4.4.1) benutzt, da diese Informationen nicht allzu komplex sind.

4.2.1 Syntax der Tätigkeitsbeschreibung

Je höher der Informationsgehalt einer Erklärung ist, desto höher ist die Komplexität des Prädikatsverbandes. Die höchste Informationsstufe, in der alle Details der Umwelt beziehungsweise des Planschritts verarbeitet werden, schreibt vor, wie der Prädikatsverband auszusehen hat. Der Aufbau ist jedoch abhängig von der jeweiligen Robotertätigkeit. Er läßt sich grob in drei Kategorien einteilen:

[29] Damit sind Speicher und Rechenkapazität von Computern gemeint.

1. *Einfache Objektmanipulation*: Hier manipuliert der Roboter jeweils nur ein Objekt. Dies kann zum Beispiel das `Greifen eines Objektes` oder das `Hinlegen eines Objektes` sein.
2. *Komplexe Objektmanipulation*: Bei dieser Art der Robotertätigkeit werden immer zwei Objekte beeinflußt. Das kann beispielsweise das `Zusammenstecken von zwei Objekten` sein.
3. *Bewegen-Handlung*: Hier wird kein Objekt manipuliert. Darunter fallen die roboterspezifischen Möglichkeiten, etwas (aber keine Szenarioobjekte) zu bewegen. Dies kann zum Beispiel die `Roboterarmbewegung` oder das `Fahren des Roboters an eine andere Position` sein.

Welche Satzglieder für den Prädikatsverband zur Verfügung stehen, wird im folgenden erörtert:

- *Verben* beschreiben die Robotertätigkeit. Sie teilen sich in das `Verbpräfix` und das `Verbsuffix` auf. Diese Gliederung ergibt sich aus der Verwendung im Satz. So hat beispielsweise das Verb `aufheben` das Präfix `heben` und das Suffix `auf`.
- *Adverben* werden für die Beschreibung von temporalen (`Adverb der Zeit`) und modalen (`Adverb`) Aspekten der Robotertätigkeit gebraucht. Beispiele dafür sind: `gerade`, `jetzt` beziehungsweise `vorsichtig`, `exakt`.
- *Objektgruppen* werden für die Beschreibung von Objektmanipulationen benötigt. Im Satzglied `Akkusativobjekt` wird das vom Roboter manipulierte Objekt beschrieben. Um diese Objekte beziehungsweise deren Manipulationen genauer spezifizieren zu können, existieren die Satzglieder `Modalergänzung im Dativ` und `Lokal- und Modalergänzung im Dativ oder Akkusativ`. Beispiele dafür sind: `ich greife den Würfel mit der roten Seite`, beziehungsweise: `Ich greife die Tasse vor der Kaffeekanne mit meiner linken Hand`.
- *Präpositionalphrasen* werden bei der Beschreibung von komplexen Objektmanipulationen gebraucht. Das dafür vorgesehene Satzglied ist die `Präpositionalphrase im Dativ oder Akkusativ`. Sie steht für eine Präposition und ein Dativobjekt, das durch die Robotertätigkeit manipuliert wird. Ein Beispiel dafür ist: `Ich füge die Schraube mit der Mutter zusammen`.

Wie der Aufbau des Prädikatsverbandes in Abhängigkeit von der Art der Robotertätigkeit aussieht, zeigen die anschließenden Definitionen. Abbildung 25 zeigt ein Szenario, in dem die Objekte Zylinder, Würfel und Knetmasse beziehungsweise die Benutzerblickrichtung (rotes Dreieck) modelliert sind. In den nachfolgenden Beispielen sind diese Gegebenheiten verarbeitet.

- Die einfache Objektmanipulation:

  `<Verbpräfix>` `<Akkusativobjekt>` `<Modalergänzung im Dativ>` `<Adverb der Zeit>` `<Adverb>` `<Lokal- und Modalergänzung im Dativ oder Akkusativ>` `<Verbsuffix>`.

Beispiel einer Erklärung der Greifen- und Ablegen-Handlung:

`Ich greife den Würfel mit der roten Seite jetzt vorsichtig vor dem Zylinder an der Position -10, -10, 0 mit meiner Hand.`

`Ich lege den Würfel mit der roten Seite nun langsam auf dem Tisch an der Position 60, 60, 0 mit meiner Hand ab.`

Dabei ist zu beachten, daß die Lokal- oder Modalergänzungen im Dativ `vor dem Zylinder an der Position -10, -10, 0 mit meiner Hand` und `auf dem Tisch an der Position 60, 60, 0 mit meiner Hand` beliebig permutiert werden können.

Je nach Verbwahl kann sich der Kasus der Lokalergänzung ändern. Betrachtet man dabei die Lokalergänzung `an der Position x, y, z`, so stellt man diese Tatsache leicht fest, wenn anstatt des verwendeten Verbs `ablegen` das Verb `legen` benutzt wird. Dieser Umstand wird mit folgendem Beispiel erläutert:

`Ich lege den Würfel mit der roten Seite nun langsam auf den Tisch an die Position 60, 60, 0 mit meiner Hand.`

Durch das Verb `ablegen` erhält die Präposition `an` die Bedeutung einer Ortsangabe, die im Dativ steht. Verwendet man aber das Verb `legen`, so ändert sich die Bedeutung von der Präposition `an` in eine Richtungsangabe, die im Akkusativ steht. Ähnlich verhält es sich bei den Präpositionen `vor`, `auf`, `hinter` und `neben`.

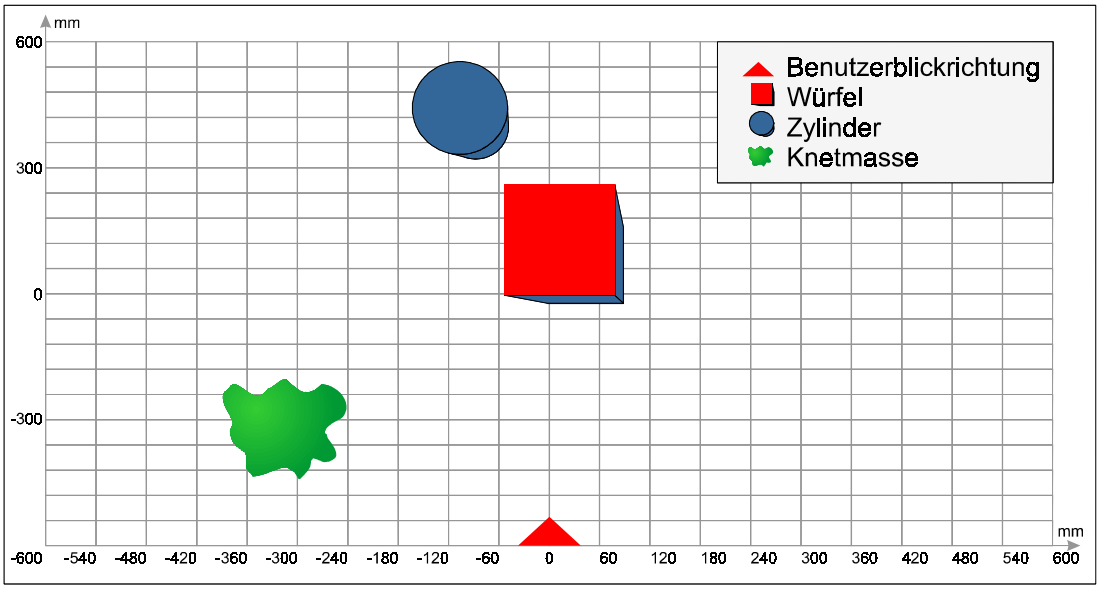

Abbildung 25: Beispielszenario mit einfachen Objekten.

- Die komplexe Objektmanipulationshandlung:

  `<Verbpräfix> <Akkusativobjekt> <Modalergänzung im Dativ> <Adverb der Zeit> <Adverb> <Präpositionalphrase im Dativ oder Akkusativ> <Lokal- und Modalergänzung im Dativ oder Akkusativ> <Verbsuffix>.`

Beispiel einer Erklärung der Zusammenfügen-Handlung:

```
Ich füge den Würfel mit der roten Seite im Augenblick langsam
mit der Knetmasse an der Position -400, -400, 0 mit meiner linken
Hand zusammen.
```

Zur Variation der Satzstrukturen kann bei einfachen und komplexen Objektmanipulationshandlungen, in denen dasselbe Objekt Gegenstand der Tätigkeit ist, Fließtext erzeugt werden. Dabei wird der Erzähler `Ich` durch das Bindewort `und` ersetzt. Zudem kann das Akkusativobjekt durch ein passendes Personalpronomen ersetzt werden.

- Die Bewegen-Handlung

  ```
  <Verbpräfix> <Akkusativobjekt> <Adverb der Zeit> <Adverb>
  <Lokal- und Modalergänzung im Akkusativ> <Verbsuffix>.
  ```

  Beispiel:

  ```
  Ich bewege meine rechte Hand nun vorsichtig an die Position 60,
  60, 0 hin.
  ```

  In solchen Satzkonstruktionen steht die Lokalergänzung immer im Akkusativ.

Die gerade vorgestellten Strukturen des Prädikatsverbandes stellen deren maximale Komplexität dar. Für Variationen der natürlichsprachlichen Erklärung können einzelne Angaben weggelassen werden. Je nach Kontext sind diese aber unterschiedlich. Eine weitere, verbunabhängige Variation des Satzbaus stellt die Verschiebung des Teilsatzes `<Adverb der Zeit> <Adverb>` zwischen das `<Verbpräfix>` und dem `<Akkusativobjekt>` dar.

Eine zusätzliche Auflockerung ist die Verschiebung vom `<Adverb der Zeit>` vor das Subjekt (`Ich`). Dazu muß zusätzlich das `<Verb>` vor dem Subjekt stehen, so daß sich folgender neuer Anfang für obige Prädikatsverbände ergibt:

```
<Adverb der Zeit> <Verb> Ich ... .
```

4.2.2 Syntax der Fehlerbeschreibungen

Erklärungen bei außerplanmäßigen Ereignissen benötigen anders geartete Satzstrukturen. Wie bei Tätigkeitsbeschreibungen ist der Aufbau des Prädikatsverbandes bei Fehlerbeschreibungen abhängig von dem beim Eintreten des Fehlers ausgeführten Roboterbefehl. Der Umstand, daß der Roboter wegen eines Fehlers die Befehlsausführung abbrechen mußte, wird dem Hörer durch die Verwendung des Modalverbs `wollte` angezeigt. Dies hat aber zur Folge, daß in den Beschreibungen die `Verbgrundform` anstatt des `Verbpräfixes` und `Verbsuffixes` verwendet werden muß.

Zur Beschreibung der Fehlersituation sind die nachstehenden Satzmuster notwendig:

- Aufbau der Fehlerbeschreibung im Falle, daß eine Bewegen-Handlung die Fehlerursache ist:

<Akkusativobjekt> <Adverb> <Lokalergänzung im Akkusativ> <Verbgrundform> wollte.

Beispiel einer Fehlerbeschreibung:

Es ist ein Fehler aufgetaucht, als ich meine rechte Hand vorsichtig an die Position 60, 60, 0 bewegen wollte.

- Aufbau der Fehlerbeschreibung im Falle, daß eine komplexe Objektmanipulationshandlung die Fehlerursache ist:

  <Akkusativobjekt> <Modalergänzung im Dativ> <Adverb> <Präpositionalphrase im Dativ oder Akkusativ> <Lokal- und Modalergänzung im Dativ> <Verbgrundform> wollte.

  Beispielbeschreibung einer Fehlersituation, die durch eine Zusammenfügen-Handlung verursacht wurde:

  Es ist ein Fehler aufgetaucht, als ich den Würfel mit der roten Seite fest auf die Knetmasse an der Position -400, -400, 0 drücken wollte.

- Aufbau der Fehlerbeschreibung im Falle, daß eine einfache Objektmanipulationshandlung Handlung die Fehlerursache ist:

  <Akkusativobjekt> <Modalergänzung im Dativ> <Adverb> <Lokal- und Modalergänzung im Dativ oder Akkusativ> <Verbgrundform> wollte.

  Beschreibung einer eventuellen Fehlersituation, die bei einer Greifen-Handlung vorkommen könnte:

  Es ist ein Fehler aufgetaucht, als ich den Würfel mit der roten Seite vorsichtig an der Position 60, 60, 0 mit meiner rechten Hand aufheben wollte.

Wie auch bei den Beschreibungen von Robotertätigkeiten können hier je nach Kontext bestimmte Angaben unterdrückt werden, sofern es die Situation verlangt, siehe Abschnitt 4.4.2.

Neben Fehlersituationen müssen Fehlerursache und Lösung beschrieben werden können. Dafür werden *Fehlersatzmasken* oder statische Meldungen, die als *Situationsbeschreibungen* bezeichnet sind, verwendet. Sie sind abhängig vom Roboter und seiner Domäne. Der Aufbau der Fehlersatzmasken ist denkbar einfach:

<**Vortext**> <**Objekt**> <**Nachtext**>.

Die Platzhalter <**Vor-** und **Nachtext**> sind für beliebigen, aber statischen Text vorgesehen. An der <**Objekt**>-Position kann ein Befehl für das Erklärungssynthesemodul gegeben werden, der das Einfügen des aktuellen Handlungsobjektes veranlaßt. Da es je

nach Satzlage in einem anderen Kasus stehen kann, existieren die dafür vorgesehenen Füllbefehle: <**OBJ-AKK**>, <**OBJ-NOM**> und <**OBJ-DAT**>. Diese Informationen müssen bereits vor der Generierung bekannt sein. Beispiele für Fehlersatzmasken sind:

Zum Greifen <**OBJ-DAT**> ist es zu eng.
<**OBJ-NOM**> wurde von mir falsch gegriffen.

Nimmt man an, daß bei Ausführung des Befehls, eine Tasse zu greifen, ein Fehler aufgetaucht ist, sähe die Objektergänzung in den obigen Fehlersatzmasken folgendermaßen aus:

Zum Greifen der Tasse ist es zu eng.
Die Tasse wurde von mir falsch gegriffen.

4.2.3 Syntax von Lokalisationsbeschreibungen

Für die Generierung von Antworten auf Lokalisationsfragen werden *Lokalisationssatzmasken* eingesetzt. Im Gegensatz zur Fehlersatzmaske geben sie keine Möglichkeiten, statischen Text unterzubringen. Eine Lokalisationssatzmaske besteht aus dem Platzhalterfeld <**Verb**> und den Füllbefehlen für Objekte (<**OBJ-NOM**>, <**OBJ-AKK**>) und Präpositionen (<**PREP**>):

<**OBJ-NOM**> <**Verb**> <**PREP**> <**OBJ-AKK**>.

Auf diese Weise wird einerseits durch den Befehl <**OBJ-NOM**> das Objekt, nach dessen Lage gefragt wurde, an die entsprechende Stelle eingefügt. Andererseits wird ein geeignetes Referenzobjekt beziehungsweise eine passende lagebeschreibende Präposition durch die Befehle <**OBJ-AKK**> und <**PREP**> an die dafür vorgesehenen Stellen eingefügt. Die Präpositionen und das Referenzobjekt werden aus den Berechnungen des Systems BOLA (siehe Kapitel 2.2) bestimmt. Bevor die Antwortsynthese abgeschlossen ist, wird mit einem zufällig ausgewählten Element aus einer Menge von geeigneten Verben (zum Beispiel: ist, liegt, steht, ...) der Platzhalter <**Verb**> belegt.

Betrachtet man das in Abbildung 12 gezeigte CAB-Szenario als Grundlage für die Frage: „Wo ist das Zwischenstück?“ , so würde folgende Antwort erzeugt werden:

Das Zwischenstück liegt links vor dem Pendel.

Trifft die Objektbezeichnung in einer Lokalisationsfrage auf mehrere Objektinstanzen zu, wird für die Berechnung des Referenzobjektes zufällig eine passende Instanz ausgewählt. Dies wäre beispielsweise bei der Frage: „Wo ist ein Bolzen?“ [30] der Fall.

[30] Es ist dabei irrelevant, ob der bestimmte oder unbestimmte Artikel verwendet wird.

4.3 Datenverwaltung

Dieses Kapitel befaßt sich mit der Analyse und Aufbereitung der zur Generierung von natürlichsprachlichen Robotertätigkeits- und Szenariobeschreibungen gebrauchten Angaben. Zudem werden hier die Datenstrukturen vorgestellt, die diese Informationen aufnehmen können. Für die Kommunikation zwischen dem Roboter und RONTRA wird ein *Steuerungsprotokoll* benötigt, in dem genau festlegt ist, welche Daten in welchem Format an das jeweilige System geschickt werden.

Das System RONTRA stellt eine Datei- und Socketschnittstelle zur Verfügung, über die Roboter benötigte Daten abgeben kann. Das heißt, für jede der im Kapitel 4.1 vorgestellten Informationsarten sind unterschiedliche Dateien vorgesehen, die RONTRA, insbesondere ROPLEX zur Gewinnung der erklärungsrelevanten Informationen zur Verfügung stehen.

4.3.1 Verwaltung der Szenariodaten

Eine geeignete Repräsentation der Szenariodaten ist für die Generierung von Erklärungen notwendig. Die Erzeugung von Antworten auf Wo-Fragen und Lokalisationsbeschreibungen, die in Tätigkeitserklärungen verwendet werden, basieren darauf. Aber auch das Befehlssynthesemodul RIACS arbeitet auf diesen Daten. Da intensiv mit ihnen gearbeitet wird, sind sie in einer unabhängigen Struktur gespeichert, so daß sie schneller und ohne großen Rechenaufwand zur Verfügung stehen. Das Modell der Szenariorepräsentation sieht vor, neben den Lageangaben für Objekte des Roboterszenarios auch deren natürlichsprachliche Bezeichnungen zu beinhalten. Damit wird erreicht, daß alle erklärungsrelevanten Daten in einer Struktur zusammengefaßt sind. Sie wird durch die *Objektinformationsstruktur*, deren Aufbau in Abbildung 26 beschrieben ist, realisiert. Sie ist Teil des Basiswissens des Roboterbefehlsgenerierungsmoduls und des Planerklärungsmoduls. Die Repräsentation gliedert sich in insgesamt drei Bereiche:

1. *Verwaltungsvariablen*: Darunter sind die Angaben zusammengefaßt, die zum Ansprechen der Datenstruktur dienen (**`name`**, **`ra-name`**, **`an-name`**). Darüber hinaus sind hier die Verweise zu den Objektrepräsentationen (**`holes`**) angegeben, die für die Objektlöcher stehen.
2. *Natürlichsprachliche Angaben*: Sie bestehen aus den natürlichsprachlichen Objektinformationen, wie Bezeichnung (**`nl-name`**), Artikel (**`det`**) und Personalpronomen (**`ppron`**).
3. *Lageinformationen*: Sie geben Drehung (**`orient`**), Koordinaten (**`coord`**) und Lage (**`pos`**) des Objektes an. Daneben geben sie Auskunft über die Objektumgebung (**`loc`**). Ausschließlich diese Angaben werden während des Einsatzes aktualisiert.

Vor dem Einsatz müssen bereits die Informationen der ersten beiden Bereiche verfügbar sein. Die Lageinformationen werden bei jeder Robotertätigkeit aktualisiert. Anschließend an diesen Absatz wird ein Beispiel einer Instanz der Objektinformations-

struktur gegeben. [31] Sie zeigt neben dem natürlichsprachlichen Wissen über die CAB-Objektinstanz `lever:1` dessen Lageinformationen, die der Roboter KAMRO der Dialogschnittstelle RONTRA zur Verfügung stellt.

Variable	Beschreibung
name	Name der Objektrepräsentation
ra-name	Interne Referenzvariable
an-name	Interne Referenzvariable
holes	Objektrepräsentationen, die für die Objektlöcher stehen
nl-name	Natürlichsprachliche Objektbezeichnungen
det	Verschiedene Artikel für die Objektbezeichnungen
ppron	Personalpronomen
coord	dreidimensionale Koordinatenangabe
orient	Objektdrehung
pos	Objektlage
loc	Verweis auf das Objekt, das unter diesem Objekt liegt

STATISCH
STATISCH
DYNAMISCH

Abbildung 26: Aufbau der Objektinformationsstruktur.

- Pendel

```
:name      'pendel-1
:ra-name   "lever:1"
:an-name   "LEVER-1"
:holes     '(pendel-1-h1)
:nl-name   '("Pendel" "Schwinger" "Hebel")
:det       '(("das" "das" "das") ("den" "dem" "der") ("den" "dem" "der"))
:ppron     '("es" "ihn" "ihn")
:coord     '(10 0 0)
:orient    0.0
:pos       "s1"
:loc       "ARBEITSPLATTE"
```

4.3.2 Verwaltung der Diagnose- und Fehlerdaten

Tritt während der Abarbeitung eines Anweisungsplans durch einen Roboter eine Fehlersituation auf, können auf der Grundlage von Diagnosedaten natürlichsprachliche Informationen über den Hergang und die Beseitigung dieses Fehlers erzeugt werden. Da sie meist in codierter Form vorliegen, existiert eine sogenannte *Fehlertranslationstabelle*, die ähnlich wie die Objekttranslationstabelle der musterorientierten Planerklärung aufgebaut ist. Sie stellt in Abhängigkeit von den zugrundeliegenden Diagnose- oder Fehlerinformationen statische Situationsbeschreibungen oder Fehlersatzmasken (siehe Abschnitt 4.2.2) bereit, aus denen natürlichsprachliche Beschreibungen erzeugt werden können.

[31] Die Beispiele sind in Lisp-Syntax verfaßt.

In Abbildung 27 sind in einer Fehlertranslationstabelle mögliche Situationsbeschreibungen und Fehlersatzmasken für Diagnosemeldungen des Roboters KAMRO gezeigt. Ergibt es sich beispielsweise, daß KAMRO für das Hinlegen des CAB-Objekts `spacingpiece:1` keinen Platz hat, könnte folgende Diagnosemeldung (siehe Abschnitt 4.1.3) generiert werden:

```
diag  <Anweisung>
      not-freespace-placegoal
      not-freespace-placegoal
      <Unteranweisung>
      remove-disturbing-o
```

Diagnoseinformation	Situationsbeschreibung oder Fehlersatzmaske
crash	Es ereignete sich eine Kollision.
crash-coagent	Meine Manipulatoren sind zusammengestoßen.
not-servo	Ein Fehler in der Servomechanik liegt vor.
not-startposition	Die Startposition meiner Manipulatoren ist falsch.
not-endposition	Die Endposition meiner Manipulatoren ist falsch.
not-free-agent	Der Manipulator ist besetzt.
not-open-agent	Der Manipulator ist geschlossen.
not-jawpos	Die Anfangsabweichung ist zu groß.
not-clean-o	<OBJ-NOM> ist dreckig oder klebrig.
not-position-o	<OBJ-NOM> liegt nicht an der angegebenen Stelle.
not-position-bo	<OBJ-NOM> ist nicht an der richtigen Stelle.
not-grasped-not-o	<OBJ-NOM> ist nicht das richtige Objekt
not-grasped-o	<OBJ-NOM> wurde von mir falsch gegriffen.
not-freespace-placegoal	Zum Hinlegen von <OBJ-DAT> ist kein Platz.
not-freespace-picktarget	Zum Greifen von <OBJ-DAT> ist es zu eng.
not-o-picktarget	<OBJ-NOM> ist nicht an der angebenen Stelle.
coagent-grasped-not-o	<OBJ-NOM> ist nicht das richtige Objekt.
coagent-not-grasped-o	<OBJ-NOM> wurde von mir falsch gegriffen.
verify-world-model	Die Umweltdaten werden neu erfaßt.
remove-disturbing-o	Störende Objekte werden von mir entfernt.
repick-o	Das Objekt wird von mir erneut gegriffen.
ignore	Der Fehler ist unkritisch.

Abbildung 27: Beispiele für Fehlersatzmasken und Situationsbeschreibungen.

Die Angaben der `Anweisung` beziehungsweise `Unteranweisung` sind für dieses Beispiel irrelevant. Zudem wird hier nicht zwischen dem Fehler und dessen Ursache unterschieden. Dies hat zur Folge, daß in den Datenzellen für Fehler und Ursache dieselbe Information steht. Mit der Fehlertranslationstabelle wird die Angabe, wie der Fehler behoben wird, auf die natürlichsprachliche Situationsbeschreibung „`Störende Objekte werden von mir entfernt`“ abgebildet. Die codierte Fehlerbeschreibung `not-freespace-placegoal` wird auf die Fehlersatzmaske „`Zum Hinlegen von <OBJ-DAT> ist kein Platz`“ abgebildet. Sie muß noch um das aktuelle Handlungsobjekt ergänzt werden, damit eine korrekte natürlichsprachliche Fehlerbeschreibung vorliegt.

Die codierten Diagnosemeldungen des Roboters und deren natürlichsprachliche Pendants werden in der *Diagnoseinformationsstruktur* gespeichert. Die darin enthaltenen Beschreibungen des Fehlers (:**nlerr**) und der Ursache (:**nlcause**) sowie der Korrekturvorschlag (:**nlgoal**) werden mit Hilfe der Fehlertranslationstabelle generiert. Die erweiterte Fehlerbeschreibung (:**excause**) wird aus den in Abschnitt 4.2.2 vorgestellten Satzmustern erzeugt. Die natürlichsprachliche Information über die Anweisung (:**nlieo**) und Unteranweisung (:**nleeo**) wird aus der im Rahmenwissen (siehe Abschnitt 4.4.1) definierten Satzmaske

`Der Befehl <`**`Befehlsname`**`> ist zum Zeitpunkt des Fehlers abgearbeitet worden.`

erzeugt. Das Diagnosewissen ist während des gesamten Fehlebehebungsvorgangs abrufbar. Den Aufbau dieser Datenstruktur zeigt die Abbildung 28.

Variable	Beschreibung	Verwendung
ieo	Anweisung, die zum Zeitpunkt des Fehlers abgearbeitet wurde.	Verschieden informative Erklärungen von Anweisungen, die zum Zeitpunkt des Fehlers abgearbeitet wurden.
nlieo	Natürlichsprachliche Beschreibung der Anweisung, die zum Fehlerzeitpunkt ausgeführt wurde.	
eeo	Unteranweisung, die zur Zeit des Fehlers abgearbeitet wurde.	
nleeo	Natürlichsprachliche Beschreibung der Unteranweisung, die zum Fehlerzeitpunkt ausgeführt wurde.	
err	Vom Planungssystem bestimmter Fehler.	Erklärungen der Fehlersituation
nlerr	Natürlichsprachliche Beschreibung des Fehlers.	
cause	Vom Planungssystem bestimmte Fehlerursache.	Verschiede Erklärungen der Fehlerursache
nlcause	Natürlichsprachliche Beschreibung der Ursache.	
excause	In natürlicher Sprache formulierte Erklärung, in der über die Fehlerursache genauer berichtet wird.	
goal	Vom Planungsystem generierte Lösung.	Lösungserklärungen
nlgoal	Natürlichsprachliche Beschreibung der Lösung.	

Abbildung 28: Aufbau der Diagnoseinformationsstruktur.

Die Annahme, daß KAMRO keinen Platz zum Greifen eines Bolzens hat, dient als Grundlage des folgenden Beispiels einer Instanz der Diagnoseinformationsstruktur. Der Planschritt, in dem der Befehl, einen Bolzen zu greifen, modelliert ist, lautet:

```
eodef (pick, 3,
          [[agent_name, red],
           [gripobj_id, [workpiece, spacer:1]]
          ]
      )
```

Dessen Ausführung aktiviert verschiedene Unterbefehle, unter anderem muß der Manipulator zunächst an die „Greifposition“ bewegt werden. Dies geschieht mit dem Unter-

befehl **finemotion** ... . Auf der Basis obiger Annahme tritt dabei eine Fehlersituation ein. Das Robotersystem erzeugt daraufhin folgende Diagnosemeldung:

```
diag
      finemotion, 3, [[agent_name, red],
        [gripobj_pos, [10, 10, 0], [workcell, assembly_station]]]
      not-freespace-picktarget
      not-freespace-picktarget
      pick, 3, [[agent_name, red], [ gripobj_id, [workpiece, spacer:1]]]
      repick-o
```

Die daraus erzeugten Fehlerbeschreibungen werden in der Diagnoseinformationsstruktur gespeichert:

```
:ieo      '("finemotion, 3, [ [ agent_name, red],
                              [ gripobj_pos, [10, 10, 0],
                                [workcell, assembly_station]
                              ]
                            ]")
:nlieo    "Der Befehl finemotion."
:eeo      '("pick, 3, [ [ agent_name, red],
                        [ gripobj_id, [workpiece, spacer:1]]
                      ]")
:nleeo    "Der Befehl pick."
:err      '("not-freespace-picktarget")
:nlerr    '(" Der Bolzen kann nicht gegriffen werden, weil es zu
              eng ist.")
:cause    '("not-freespace-picktarget")
:nlcause  '(" Der Bolzen kann nicht gegriffen werden, weil es zu
                              eng ist.")
:excause  '(  "Als ich den Bolzen greifen wollte, ist ein
              Fehler aufgetaucht.")
:goal     repick-o
:nlgoal   '("Das Objekt wird von mir erneut gegriffen.")
```

4.3.3 Der Verbrahmen als flexible Satzgrundlage

Wie sich herausgestellt hat, kann die Satzmaske der musterorientierten Planerklärungsmethode als Vorstufe der natürlichsprachlichen Beschreibung eines Roboterbefehls den im Kapitel 3.2 beschriebenen Anforderungen an eine roboter- und sprachunabhängige Erklärungskomponente allgemein nicht gerecht werden. Ihr Aufbau ist für die Erzeugung von Fließtext und Satzvariationen bei Robotertätigkeitsbeschreibungen nicht geeignet. Zwar sind die in Kapitel 3.1 angesprochenen Probleme grundsätzlich mit der Satzmaske als Hauptwissensrepräsentation von Roboterplanschritt- und Fehlerdiagnoseinformationen lösbar, aber aufgrund ineffizienter Konfigurations- und Verwaltungsverfahren generell zu aufwendig. Dies wird insbesondere von der Tatsache unter-

stützt, daß die Methoden und die Wissensrepräsenation nicht, wie in ROPLEX, roboter- und sprachunabhängig sind und somit auf jeden Roboter einzeln angepaßt werden müssen. Der *Verbrahmen* ist aufgrund seiner Flexibilität hinsichtlich der Erzeugung von Sätzen besser für diese Aufgabe geeignet. Im Gegensatz zur Satzmaske ist der Verbrahmen als Datenstruktur gänzlich sprachunabhängig. Er enthält selbst keine syntaktischen oder semantischen Angaben zum Satzbau. Damit bestimmt alleine der Erklärungsalgorithmus die Struktur der natürlichsprachlichen Beschreibung. In einem Verbrahmen ist das Wissen repräsentiert, das zur Generierung einer Beschreibung eines Planschritts notwendig ist. Bevor ein Verbrahmen zur Wissensrepräsentation genutzt werden kann, müssen invariante natürlichsprachliche Informationen sowie interne Steuerkommandos angegeben werden. Dies gilt für alle an der Erklärung beteiligten Verbrahmen. Die Datenfelder, die statische Informationen beziehungsweise Steuerkommandos aufnehmen, sind im folgenden beschrieben:

- *Steuerkommandos*: Um den Artikel an den Kasus von Objektbezeichnungen anzupassen, gibt es die Kommandos `DETAKK` (für Akkusativ) und `DETDAT` (für Dativ). Dies gilt für die Datenfelder :**`objdet`**, :**`addobjdet`**, :**`wheredet`** und :**`connectdet`**. Daneben gibt es den Befehl `NONE`, der die Informationsaufname für ein Verbrahmen-Datenfeld sperrt.
- *Verbinformation*: Sie ist für die Bezeichnung der Tätigkeit des Roboters notwendig. Für die Erklärung werden neben der Grundform (:**`infinverb`**) auch Präfix (:**`frontverb`**) und Suffix (:**`backverb`**) des Verbs sowie gegebenenfalls ein Adverb (:**`verbadv`**) benötigt. Bei der Generierung von Tätigkeitsbeschreibungen werden sie an den entsprechenden Stellen des zugrundeliegenden Satzmusters (siehe Abschnitte 4.2.1 und 4.2.2) eingefügt.
- *Präpositionalphrasen*: Sie sind Satzergänzungen, die schon vor einer Erklärung feststehen. So beispielsweise in einem Verbrahmen, der die Informationen für die Beschreibung einer Greifen-Handlung aufnimmt, wie zum Beispiel die Lokalergänzung „`an der Position`“ (:**`posprep`**). Sie wird für die Koordinatenangabe, die angibt, wo das Objekt gegriffen wird, gebraucht. Die korrespondierenden Bereiche im für die Tätigkeitsbeschreibung verwendeten Satzmuster sind Präpositionalphrasen oder Lokal- und Modalergänzungen im jeweiligen Kasus (siehe Abschnitte 4.2.1 und 4.2.2).
- *Sonstige Bezeichnungen*: Damit sind die Angaben des Subjekt (:**`teller`**) und der Name des Greifapparates (:**`mani`**) gemeint.

Die eigentliche Aufgabe des Verbrahmens ist die Beherbergung der natürlichsprachlichen Pendants der zu beschreibenden Werte des aktuellen Roboterbefehls sowie die Aufnahme zusätzlicher Umwelt- oder Roboterinformationen, die für die Erzeugung zusätzlicher informativer Erklärungen notwendig sind. Dafür sind dynamische Datenfelder angelegt; für das Handlungsobjekt existieren zum Beispiel :**`obj`** für die Objektbezeichnung, :**`objinfo`** für zusätzliche Lagebeschreibungen, :**`ppron`**, um ein Personalpronomen anzugeben, sowie :**`pos`** für die Koordinatenangabe. Abbildung 29 zeigt eine Zusammenfassung aller Datenfelder, aus denen sich der Verbrahmen zusammensetzt.

Darüber hinaus informiert sie, welche der Daten während des Erklärungsvorgangs im Verbrahmen gespeichert werden (dynamische Datenfelder) beziehungsweise vorher angegeben werden müssen (statische Datenfelder).

Variable	Beschreibung	Art der Information
name	Der Name der Verbrahmeninstanz	STATISCH
teller	Angabe des Subjekts	STATISCH
infinverb frontverb backverb verbadv	Verbinformationen, bestehend aus Grundform, Präfix, Suffix und Adverb.	STATISCH
obj objdet ppron objinfo	Angaben über Aktions-, Referenz-, Berührungs- und Fügeobjekt.	DYNAMISCH
objinfoprep		STATISCH
pos		DYNAMISCH
posprep		STATISCH
addobj addobjdet addobjadj		DYNAMISCH
addobjprep		STATISCH
whereobj wheredet whereadj whereprep connectobj connectdet		DYNAMISCH
connectprep		STATISCH
mani maniprep manipreponly	Manipulatorangaben, bestehend aus technischen Details, wie zum Beispiel Öffnungsweite und natürlichsprachliche Bezeichnungen, die zur Identifikation und zur genauer Erklärung der Tätigkeit dienen.	STATISCH
maniadj manimove		DYNAMISCH
manimoveprep		STATISCH
manipos maniwidth maniadd		DYNAMISCH

Abbildung 29: Aufbau der Verbrahmen-Datenstruktur von ROPLEX.

Neben der Bedingung, daß für jeden Typ eines Roboterbefehls ein oder mehrere Verbrahmen existieren müssen, um Erklärungen in der Gegenwartsform generieren zu können, muß es Verbrahmen für Tätigkeitsbeschreibungen im Perfekt und Futur I geben, damit ganze Planpassagen dem Benutzer chronologisch korrekt beschrieben werden können.

Beispiele für Instanzen von Verbrahmen, die für Roboteranweisungen an KAMRO erzeugt worden sind, werden am Ende dieses Kapitels präsentiert. Dabei ist zu beachten, daß es sich hier um unausgefüllte Verbrahmen handelt, das heißt sie enthalten keine konkrete Planschrittinformation. Darin befinden sich lediglich die (statischen) Angaben, die unabhängig von der jeweiligen Roboteranweisungen sind, aber für die Generierung der natürlichsprachlichen Beschreibung gebraucht werden. Zur Verdeutlichung sind diese invarianten Angaben mit der Farbe rot markiert und unterstrichen.

- Leerer Verbrahmen[32] für die KAMRO-Anweisung **`place-placed`** im Präsens:

```
:name          'hinlegen
:teller        "Ich"
:infinverb     "hinlegen"
:frontverb     "lege"
:backverb      "hin"
:verbadv       "NONE"
:obj           ""
:objdet        "DETAKK"
:objinfo       ""
:objinfoprep   "mit der Seitenlage"
:ppron         ""
:addobj        ""
:addobjdet     "DETDAT"
:addobjadj     ""
:addobjprep    "auf"
:pos           ""
:posprep       "an der Stelle"
:mani          ("Greifarm")
:maniprep      ("mit meinem")
:manipreponly  "mit einem meiner Greifarme"
:maniadj       ""
:manimove      "NONE"
:maniprep      "NONE"
:manipos       "NONE"
:maniwidth     "NONE"
:maniadd       "NONE"
:whereobj      ""
:wheredet      "DETDAT"
:whereadj      ""
:whereprep     ""
:connectobj    "NONE"
:connectdet    "NONE"
:connectprep   "NONE"
```

[32] Die Angabe erfolgt in Lisp-Syntax.

- Leerer Verbrahmen für die explizite KAMRO-Anweisung **finemotion** im Perfekt:

```
:name          'habenvorsichtigbewegen
:teller        "Ich habe"
:infinverb     "bewegen"
:frontverb     "NONE "
:backverb      "bewegt"
:verbadv       "vorsichtig"
:obj           "NONE"
:objdet        "NONE"
:objinfo       "NONE"
:objinfoprep   "NONE"
:ppron         "NONE"
:addobj        "NONE"
:addobjdet     "NONE"
:addobjadj     "NONE"
:pos           "NONE"
:posprep       "NONE"
:mani          ("Greifarm")
:maniprep      ("meinem")
:manipreponly  "einem meiner Greifarme"
:maniadj       ""
:manimove      ""
:maniprep      "an die Stelle"
:manipos       ""
:maniwidth     "NONE"
:maniadd       ""
:whereobj      "NONE"
:wheredet      "NONE"
:whereadj      "NONE"
:whereprep     "NONE"
:connectobj    "NONE"
:connectdet    "NONE"
:connectprep   "NONE"
```

4.4 Grundlagen des Erklärungsprozesses

Jeder Tätigkeit, die vom Menschen oder von einer Maschine ausgeführt wird, liegen gewisse Voraussetzungen zu Grunde. Dabei muß der Akteur (in diesem Fall die Maschine) wissen, wie er die Tätigkeit auszuführen hat. Zudem muß er Angaben über den Raum haben, in dem er handelt. Auch für die Erklärung von Anweisungsplänen beziehungsweise von Fehler- und Lokalisationsbeschreibungen sind verschiedene Angaben notwendig. Sie sind je nach Gebiet in geeignete Wissensbasen aufgeteilt. Zusammengefaßt ergeben sie das für Erklärungen notwendige *Basiswissen* (siehe Abschnitt 4.4.1).

Das Wissen, wie Sachverhalte dem Menschen erklärt werden sollen, ist im *Konfigurationswissen* (siehe Abschnitt 4.4.2) zusammengefaßt.

Mit Hilfe dieser Wissensbasen und geeigneten Verfahren ist es möglich, ein Erklärungsmodul für autonome Roboter (Maschinen) zu realisieren, das die Aufgabe der Beschreibung der aktuell vom Roboter ausgeführten Tätigkeit in Echtzeit beherrscht. Darüber hinaus bietet das Modul dem Benutzer die Möglichkeit, zu jeder Zeit Auskunft über Sachverhalte der Umwelt sowie, falls präsent, Informationen über Fehlersituationen mitzuteilen. Für jede dieser Erklärungsaufgaben sind verschiedene Verarbeitungsabläufe notwendig. Einen kleinen Überblick hierüber gibt der hier grob skizzierte Algorithmus für die synchrone Tätigkeitsbeschreibung:

1) Statusinformationen analysieren. Sind darin Angaben über eine Fehlersituation, so werden diese Informationen in der Diagnoseinformationsstruktur gespeichert.
2) Szenariodaten abrufen und in der Objektinformationsstruktur ablegen.
3) Die Planschrittinformation, die aus den Steuerdaten ermittelt wurde, analysieren und die so gewonnenen Daten im Verbrahmen ablegen.
4) Auf der Basis der zugrundeliegenden Daten die natürlichsprachliche Beschreibung der Tätigkeit oder der Fehlersituation erzeugen und dem Menschen mitteilen.

Die Vorgehensweise für den parallel ablaufenden Antwortgenerierungsprozeß sieht folgendermaßen aus:

1) Eingabe des Menschen analysieren. Die Ergebnisse in der Eingabeinformation (siehe Kapitel 2.3) speichern.
2) Abhängig vom Inhalt der Eingabeinformation wird eine entsprechende natürlichsprachliche Ausgabe generiert. Die Informationen, die dieser Ausgabe zugrunde liegen, sind der Inhalt des Basiswissens, das zu dem Zeitpunkt des Stellens der Frage aktuell war.

Ein weiteres, parallel ablaufendes Verfahren ist die Verarbeitung von Konfigurationsbefehlen für die Erklärungskomponente. Es setzt sich aus der syntaktischen Analyse des Konfigurationsbefehls des Menschen und der sich anschließenden Interpretation (semantische Analyse) der Eingabeinformation zusammen.

4.4.1 Basiswissen

Die Informationen, die der Erklärungskomponente zur Beantwortung von Fragen und zur Generierung von Handlungsbeschreibungen vorliegen, sind als *Basiswissen* definiert. Es bedient sich unterschiedlicher Datenstrukturen, um notwendiges Wissen zu speichern. Für den Einsatz bei Erklärungen ist es notwendig, daß bestimmte Angaben schon vorher bekannt sind. Zudem werden während eines Erklärungslaufs dynamische Angaben erfaßt und in dafür vorgesehene Wissensbereiche abgelegt. Die Informationsarten und deren dazugehörige Datenstrukturen, die für die Generierung von natürlichsprachlichen Aussagen benötigt werden, sind im folgenden zusammengefaßt:

- *Umweltwissen*: Diese Angaben werden in der Objektinformationsstruktur gespeichert. In ihr sind Angaben über die Lage des Objektes sowie ihm zugehörige natürlichsprachliche Informationen, wie Bezeichnung und Artikel, abgelegt, siehe Abschnitt 4.3.1.
- *Rahmenwissen*: Dieses Wissen umfaßt Bezeichnungen für Plandateien und Beschreibungen für codierten Informationen sowie statische Erklärungen für verschiedene Situationen. Da diese Angaben weniger umfangreich ausfallen, sind für sie keine komplexen Datenstrukturen notwendig; sie werden in tabellarischer Form abgelegt. Für den Roboter KAMRO ist unter anderem folgendes Rahmenwissen[33] definiert:

```
'( ("select"                "")
  ("red"                    "rechten")
  ("blue"                   "linken")
  ("NOT-POSITION-O.PRO"     "Korrekturplan Nr.1")
  ("TESTPLAN.PRO"           "Testplan Nr.1")
  ("TESTPLAN1.PRO"          "Testplan von Patrick")
)
```

  Hier werden unter anderem die codierten Bezeichnungen für die Manipulatoren und Anweisungspläne auf natürlichsprachliche Bezeichnungen abgebildet. Wie man aus dem obigen Beispiel entnehmen kann, ist im System KAMRO der rechte Greifarm mit `red` bezeichnet, der linke mit `blue`.

- *Temporalinformationen*: Ähnlich wie beim Rahmenwissen sind in einer einfachen Liste Zeitadverbien der Gegenwart aufgelistet, die zur Auflockerung der Satzstruktur dienen. Beispiele dafür sind: `jetzt`, `gerade`, `im Augenblick`, `nun` und so weiter.
- *Satzaussage*: Bevor dieses System zur Erzeugung natürlichsprachlicher Beschreibungen einer Robotertätigkeit eingesetzt werden kann, wird festgelegt, welche Verben sie am besten beschreiben. Für jedes Verb muß ein leerer Verbrahmen generiert werden, in dem dann die natürlichsprachlichen Pendants der Roboteranweisungsvariablen abgelegt werden können. Beispiele für einen leeren Verbrahmen finden sich in Abschnitt 4.3.3.
- *Eingabewissen*: Um natürlichsprachliche Äußerungen des Menschen identifizieren zu können, müssen die darin vorkommenden Satzkomponenten, wie Verben, Präpositionen und Objekte, erkannt werden.[34] Daraus wird dann der Eingabetyp ermittelt. Im Augenblick wird zwischen Lokalisationsfragen, Eingaben bei Fehlersituationen und Konfigurationsbefehlen unterschieden. Für die Erkennung von Satzkomponenten kommen Schlüsselwörtermengen, die mit *Eingabeidentifizierer* bezeichnet werden, zum Einsatz. Sie enthalten alle für eine jeweilige Komponente

[33] Die Angabe erfolgt in LISP-Syntax, ist aber informell.

[34] Die Äußerung wurde zuvor von einem Parser syntaktisch analysiert. Dadurch sind die einzelnen Komponenten bekannt.

möglichen Wörter. Der Eingabeidentifizierer für Fehlerbezeichnungen (`ErrcauseNons`) enthält die Wörter: [35] `fehler`, `fehlerursache` und `ursache`. Um eine Äußerung typisieren zu können, bedarf es einer Kombination aus mehreren Eingabeidentifizierern, die mit *Identifizierungsmaske* bezeichnet ist. Die Maske, die zur eindeutigen Erkennung vom Befehl: „`Nenne mir den Fehler`!“ gebraucht wird, ist:

```
SayVerbs       '("nenn" "sag")
ErrcauseNons   '("fehler" "fehlerursache" "ursache")
```

Durch die dadurch erreichte Flexibilität, können auch Äußerungen, die nicht in deutscher Sprache formuliert sind, interpretiert werden. Die komplette Liste aller Eingabeidentifizierer und deren Kombinationen, die zur Unterscheidung deutschsprachiger Äußerungen benötigten werden, befinden sich in Kapitel 6.1.

- *Fehlerwissen*: Das Fehlerwissen besteht unter anderem aus der Fehlertranslationstabelle und Situationsbeschreibungen (siehe Abschnitt 4.3.2), die für die natürlichsprachliche Übersetzung der Fehlermeldungen des Roboters gebraucht werden, und speziell für Ausnahmesituationen vorbereiteten Datenstrukturen, in denen der Fehler beschrieben wird. Darüber hinaus werden die Ausnahmesituationen betrachtet. Treten infolge von Informationsdefiziten (zum Beispiel fehlerhafter oder fehlender Plan auf Seiten des Roboters) Anomalien beim Erzeugen von natürlichsprachlichen Tätigkeitsbeschreibungen auf, so wird automatisch ein an die Situation angepaßter, vordefinierter *Fehler-Verbrahmen* benutzt, der den Benutzer über die Art des internen Verarbeitungsfehlers benachrichtigt.
- *Roboterzustand:* Diese zentrale Variable spiegelt die Art der Robotertätigkeit wider. Hier wird vermerkt, ob der Roboter in einer Fehlersituation operiert oder nicht. Die dafür vorgesehenen Werte heißen `Planabarbeitung` und `Fehlerplanabarbeitung`. Diese Datenstruktur wird für das korrekte Beantworten von Anfragen des Menschen bezüglich einer Fehlersituation benötigt.
- *Satzaufbau*: Das Wissen über die Satzstruktur von Beschreibungen ist in der Generierungsmethode für Beschreibungen selbst enthalten. Die möglichen Satzformen, die bei einer generierten Fehler-, Tätigkeits- oder Lokalisationsbeschreibung benutzt werden können, sind bereits in Kapitel **Fehler! Verweisquelle konnte nicht gefunden werden.** beschrieben.

4.4.2 Konfigurationswissen

Dieses Wissen besteht zum einen aus Anleitungen, wie der Analysealgorithmus aus Plänen die benötigten Daten extrahieren kann. Zum anderen sind hier die Angaben zusammengefaßt, wie dem Menschen die natürlichsprachliche Beschreibung der Robotertätigkeit präsentiert wird. Das *Konfigurationswissen* ist im allgemeinen abhängig vom Roboter und muß, bevor ROPLEX dessen Ausgaben interpretieren kann, an ihn angepaßt werden.

[35] Dabei wird nicht zwischen Groß- und Kleinschreibung unterschieden.

4.4.2.1 Das Planschrittanalyseschema

Damit die Stellen des Planschritts (Befehl) angesprochen werden können, an denen sich die erklärungsrelevante Information befindet, muß zunächst dessen syntaktische Struktur bekannt sein. Dieses Wissen erhält das Analyseverfahren durch das *Planschrittanalyseschema*. Um alle Befehle eines bestimmten Roboters analysieren zu können, muß es für jeden Befehl ein solches Schema geben. Es definiert, welche Informationen aus dem Planschritt extrahiert und in welches Datenfeld diese im Verbrahmen gespeichert werden. Der Roboterbefehl, der im Planschritt modelliert ist, wird syntaktisch anhand der maximal zwei Befehlsnamen identifiziert (siehe dazu Abschnitt 4.1.2). Die des Fügen-Befehls für KAMRO sind beispielsweise **`place`** und **`connected`**. Sie sind zu Beginn des Plananalyseschemas angegeben. Daneben sind in einer Liste die Instanzen des Verbrahmens angeben, die als Grundlage für die Beschreibung dieses Roboterbefehls in Frage kommen. Darüber hinaus sind in einer weiteren Liste sogenannte *Füllbefehle* angegeben. In ihnen ist zum einen eine Suchsequenz, bestehend aus mehreren Zeichenfolgen, definiert, die angibt, an welcher Stelle im Planschritt erklärungsrelevante Informationen stehen. Zum anderen weisen sie den Analysealgorithmus an, in welches Verbrahmen-Datenfeld später das natürlichsprachliche Pendant der extrahierten Information gespeichert werden soll. Letztendlich können dort Konfigurationsbefehle an die Erklärungssynthese gegeben werden, die den Inhalt der Beschreibung manipulieren. Im Augenblick ist dies jedoch auf den Befehl `say_mani` beschränkt. Er weist an, die Information des Manipulators in der Tätigkeitsbeschreibung anzugeben.

Eine Zusammenfassung des Aufbaus und die Beschreibung der einzelnen Komponenten des Plananalyseschemas zeigt Abbildung 30.

Im folgenden werden Beispiele dieser Konfigurationsstruktur, die Roboterbefehlen von KAMRO Verbrahmen der Gegenwartsform zuweisen, vorgestellt:

- Plananalyseschema für den Greifen-Befehl:

```
('pick' ''
  ("GREIFEN" "AUFHEBEN" "PACKEN" "NEHMEN") (
    ('maniadj'     (',' ',' ', ') ']')
    ('obj'         ('[gripobj_id,' ', ') ']')
  )
)
```

  Der Füllbefehl **`maniadj`** gibt der Analysekomponente vor, die Information, welcher Manipulator das Objekt greift, hinter Leerzeichen nach dem dritten Komma und vor der nächsten schließenden eckigen Klammer zu suchen. Ähnliches geschieht bei dem Füllbefehl **`obj`**.

- Plananalyseschema für den Ablegen-Befehl:

```
('place' 'placed'
  ("LEGEN" "STELLEN" "SETZEN") (
     ('maniadj'    (', ' ', ' ',') '] ')
     ('obj'        ('[gripobj_id,' ', ') '] ')
     ('objinfo'    ('[gripobj_pos,' ',' ':') ',')
     ('objadd'     ('[gripobj_pos,' ',' ',' ',' ',' ', ') ']'))
     ('pos'        ('[gripobj_pos,' ',' ',' '[') ']')
  )
)
```

```
('1. Befehlsname' '2. Befehlsname'
   ('1. Verbrahmen' ... 'n. Verbrahmen')
    (('1. Füllbefehl' (1.Suchsequenz)) ... ('n. Füllbefehl' (n. Suchsequenz)))
   'Ausgabe-Konfigurationsbefehl')
)
```

Befehlsidentifizierer für KAMRO:

```
pick
place connected
place placed
detach
finemotion
grasp
setgripper
transfer
```

Füllbefehle und deren Beschreibung:

obj	Handlungsobjekt im Planschritt suchen und natürlichsprachliche Bezeichnung in den Verbrahmen eintragen.
objinfo	Im Planschritt nach zusätzlichen Objektinformationen suchen. Diese an die entsprechende Stelle im Verbrahmen ablegen.
objadd	Die Information des Referenzobjektes aus dem aktuellen Planschritt extrahieren und das nat. Pendant in den Verbrahmen einfüllen.
maniadj	Die Information über den benutzen Manipulator aus dem Planschritt entnehmen und dessen nat. Übersetzung in den Verbrahmen eintragen.
manipos	Zielkoordinate des Manipulators im aktuellen Planschritt suchen und an die im Verbrahmen dafür vorhergesehene Stelle schreiben.
maniwdth	Öffnungsweite des Manipulators aus dem Planschritt bestimmen und die Information an die dafür vorgesehene Stelle im Verbrahmen ablegen.
connectobj	Fügeobjekt im Planschritt suchen und nat. Bezeichnung in den Verbrahmen eintragen.
pos	Nach den Positionsangaben des Handlungsobjektes im Plan suchen und an der entsprechenden Stelle im Verbrahmen ablegen.

Ausgabe-Konfigurationsbefehl und dessen Bedeutung:

say_mani	Anweisung an die Planerklärung, daß die Manipulatorangaben in jedem Fall in der Erklärung genannt werden müssen.

Abbildung 30: Aufbau des Planschrittanalyseschemas.

4.4.2.2 Das Erklärungsprofil

Da der Roboter kein Wissen darüber hat, welche Information der menschliche Kommunikationspartner austauschen will, muß der Teil des Robotersystems, der die Kommunikation regelt, in diesem Fall die Dialogschnittstelle RONTRA, dahingehend konfi-

gurierbar sein. Je nach Situation beziehungsweise Aufgabe, die der Roboter zu erledigen hat, will der Mensch mehr oder weniger Handlungs- oder Umweltdetails in Erklärungen mitgeteilt bekommen. Er muß also die Möglichkeit haben, dem Roboter mitzuteilen, daß er über bestimmte Modalitäten informiert werden möchte. Die individuellen Einstellungen der natürlichsprachlichen Beschreibung einer Robotertätigkeit sind im *Erklärungsprofil* gespeichert. Das dem Menschen am nächsten gelegene Kommunikationsmedium ist die natürliche Sprache. Aus diesem Grund kann über Befehle in natürlicher Sprache das Erklärungsprofil manipuliert werden. Zu ihrer Interpretation steht das Wissen der Eingabeidentifizierer zur Verfügung. Änderungen am Erklärungsprofil können auch durch die graphische Benutzeroberfläche RONTRAGUI mittels eines Dialogmenüs vorgenommen werden, das sich im praktischen Einsatz bewährt hat. Die Einstellungen der natürlichsprachlichen Ausgabe gibt die Abbildung 31 wieder. Diese lassen sich in die drei Kategorien *globale Einstellungen*, *natürlichsprachliche Erklärungen von Planschritten* und *natürlichsprachliche Fehlererklärungen* einteilen.

Im folgenden werden die globalen Einstellungsmöglichkeiten erörtert und deren natürlichsprachlichen Pendants[36] gezeigt:

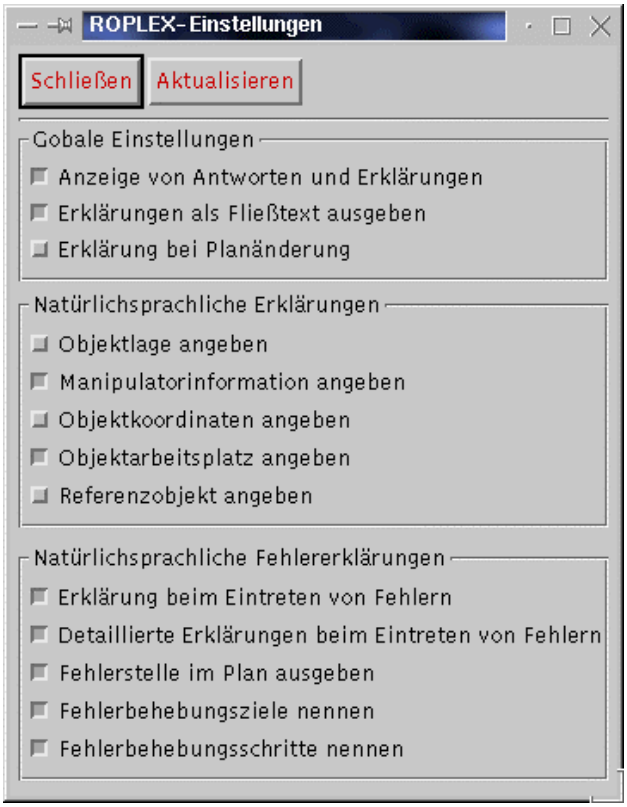

Abbildung 31: Beipielkonfiguration vom Erklärungsprofil.

- *Anzeige von Antworten und Erklärungen* aktiviert oder deaktiviert den natürlichsprachlichen Erklärungsprozeß. Die natürlichsprachlichen Kommandos für das Aktivieren sind: „`Sage etwas!`“ oder „`Erkläre mir etwas!`“ Ein möglicher Befehl für das Deaktivieren ist: „`Sei ruhig!`“.
- *Erklärungen als Fließtext ausgeben* aktiviert oder deaktiviert die natürlichsprachliche Erklärung in zusammenhängenden Sätzen, falls das Objekt der Handlung dasselbe geblieben ist. Das natürlichsprachliche Kommando dafür ist: „`Erkläre im Fließtext!`“. Das Kommando zum Abstellen dieser Eigenschaft ist: „`Erkläre ohne Fließtext!`“.
- *Erklärung bei Planänderung*: Ändert sich der Anweisungsplan für den Roboter, so kann durch Aktivieren dieses Schalters dieser Umstand dem Benutzer mitgeteilt werden. Der in natürlicher Sprache formulierte Befehl zu Einschalten dieser Option

[36] Weitere Beispiele für natürlichsprachlichen Konfigurationskommandos findet man im Abschnitt 6.1.3.

ist: „Berichte mir von Planänderungen!“. Der Befehl zum Ausschalten ist: „Berichte nicht von Planänderungen!“.

Die Optionen, die zur Variation der natürlichsprachlichen Planerklärung in ROPLEX existieren, werden im folgenden beschrieben:

- *Objektlage und Objektkoordinaten angeben*: Beim Aktivieren einer dieser Einstellungen tauchen gegebenenfalls Lageangaben und Koordinaten des Handlungsobjektes in der natürlichsprachlichen Beschreibung von Roboterhandlungen auf. Die Pendants zu dieser Einstellung sind die Befehle: „Nenne mir die Objektlage!“ oder „Nenne mir die Objektkoordinaten!“ und „Erkläre ohne Objektlage!“ oder „Erkläre ohne Objektkoordinate!“.
- *Manipulatorinformation angeben*: Durch das Auswählen dieser Option werden die Angaben, mit welchem Manipulator die Tätigkeit ausgeführt wird, eingeflochten. Ein Kommando zum Aktivieren ist: „Nenne mir die Manipulatoren!“. Ein Befehl zum Deaktivieren ist: „Nenne mir die Manipulatoren nicht!“.
- *Objektarbeitsplatz angeben*: Mit dieser Möglichkeit der Einstellung kann der natürlichsprachlichen Beschreibung die genaue Lokationsangabe, wo der Roboter mit dem Handlungsobjekt arbeitet, hinzugefügt oder weggenommen werden. Der Befehl „Erkläre mit Objektarbeitsplatz!“ oder „Erkläre ohne Objektarbeitsplatz!“ aktiviert beziehungsweise deaktiviert diese Option.
- *Referenzobjekt*: Mit dem Aktivieren dieser Einstellungsmöglichkeit wird eine räumliche Beschreibung des Handlungsobjekts eingefügt. Ein äquivalentes natürlichsprachliches Kommando lautet: „Erkläre mit Referenzobjekt!“. Ein natürlichsprachlicher Befehl zum Deaktivieren dieser Option ist: „Erkläre ohne Referenzobjekt!“.

Nachstehend werden die Optionen, die auf die Beschreibung von Fehlersituationen Einfluß nehmen, erörtert:

- Die Schalter *Erklärung beim Eintreten von Fehlern*, *detaillierte Erklärungen beim Eintreten von Fehlern* und *Fehlerstelle im Plan ausgeben* verändern den Informationsgehalt der Mitteilung einer Fehlersituation. Ist von Seiten des Menschen überhaupt keine Benachrichtigung erwünscht, so muß der Schalter *Erklärung bei Eintreten von Fehler* deaktiviert sein. Dies erreicht man auch mit dem Befehl: „Erkläre keine Fehler!“. Der Befehl „Erkläre mir Fehler!“ bewirkt das Gegenteil. Ist eine genauere Beschreibung erwünscht, so kann dies durch Aktivieren der Optionen *Detaillierte Erklärungen beim Eintreten von Fehlern* und *Fehlerstelle im Plan ausgeben* erreicht werden. Die korrespondierenden natürlichsprachlichen Kommandos dafür sind: „Erkläre den Fehler genau!“ oder „Erkläre den Fehler normal!“ und „Erkläre die Fehlerstelle des Plans!“ oder „Erkläre die Fehlerstelle im Plan nicht!“.
- *Fehlerbehebungsziele* und *Fehlerbehebungsschritte nennen* aktivieren oder deaktivieren die automatische Benachrichtigung des Menschen über einzelne Ziele beziehungsweise Aktionen zur Fehlerbehebung. Mögliche natürlichsprachliche

Kommandos dafür sind: „`Nenne die Fehlerbehebungsziele!`“ oder „`Nenne die Fehlerbehebungsziele nicht!`“ und „`Nenne die Fehlerbehebungsschritte!`“ oder „`Nenne die Fehlerbehebungsschritte nicht!`“.

Neben diesen Feineinstellungen gibt es natürlichsprachliche Kommandos, die mehrere Optionen aktivieren beziehungsweise deaktivieren. Diese Befehle und ihre Auswirkungen werden im folgenden beschreiben:

- „`Erkläre alles!`“: Mit diesem Befehl wird jede der oben präsentierten Optionen eingeschaltet.
- „`Erkläre einfach!`“: Diese Anweisung aktiviert folgende Optionen:
  - Anzeige von Antworten und Erklärungen
  - Erklärungen im Fließtext ausgeben
  - Manipulatorinformation angeben
  - Objektarbeitsplatz angeben
  - Erklärung beim Eintreten von Fehlern
  - Fehlerbehebungsziele nennen

  Alle anderen Optionen werden deaktiviert. Der Informationsgehalt der Erklärungen, der durch dieses Erklärungsprofil bestimmt ist, hat sich im praktischen Einsatz bewährt.

4.5 Der Erklärungsalgorithmus

Das Verfahren von ROPLEX, natürlichsprachliche Beschreibungen der Robotertätigkeit aus Planschrittinformationen zu generieren, baut sich aus den drei Arbeitsschritten auf, die im folgenden beschrieben werden (siehe auch Abbildung 32):

1) *Wissensauffrischung*: In dieser initialen Phase werden die roboterabhängigen Informationen, insbesondere die der Steuerdatei (siehe Abschnitt 4.1.3) interpretiert und die Ergebnisse in entsprechende Datenstrukturen abgelegt. Besteht der Inhalt der Steuerdatei aus einer Diagnoseinformation, dann werden daraus später natürlichsprachliche Fehlererklärungen generiert und in der Diagnoseinformationsstruktur gespeichert. Anschließend wird abhängig vom Inhalt der Wert der Roboterzustandsvariable (siehe Abschnitt 4.4.1) geändert. In dieser Phase werden zusätzlich der Inhalt des aktuellen Plans sowie die von ihm gelieferten Umweltdaten interpretiert. Das Ergebnis wird in dafür vorgesehene Wissensbasen abgelegt.
2) *Planschrittanalyse*: In dieser Phase wird der aktuelle Planschritt analysiert. Dadurch erfaßte relevante Daten, beziehungsweise deren natürlichsprachliche Pendants, werden in einem geeigneten Exemplar des Verbrahmens gespeichert.
3) *Erklärungssynthese*: In diesem Schritt werden die Informationen des Verbrahmens oder der Diagnoseinformation vervollständigt und daran anschließend zu einem natürlichsprachlichen Aussagesatz zusammengesetzt, der dann dem Benutzer präsentiert wird.

Abbildung 32 illustriert graphisch den Datenfluß bei Erklärungen. Zudem gibt sie einen genauen Überblick der Daten und Wissensbasen, die bei den jeweiligen Bearbeitungsschritten notwendig sind.

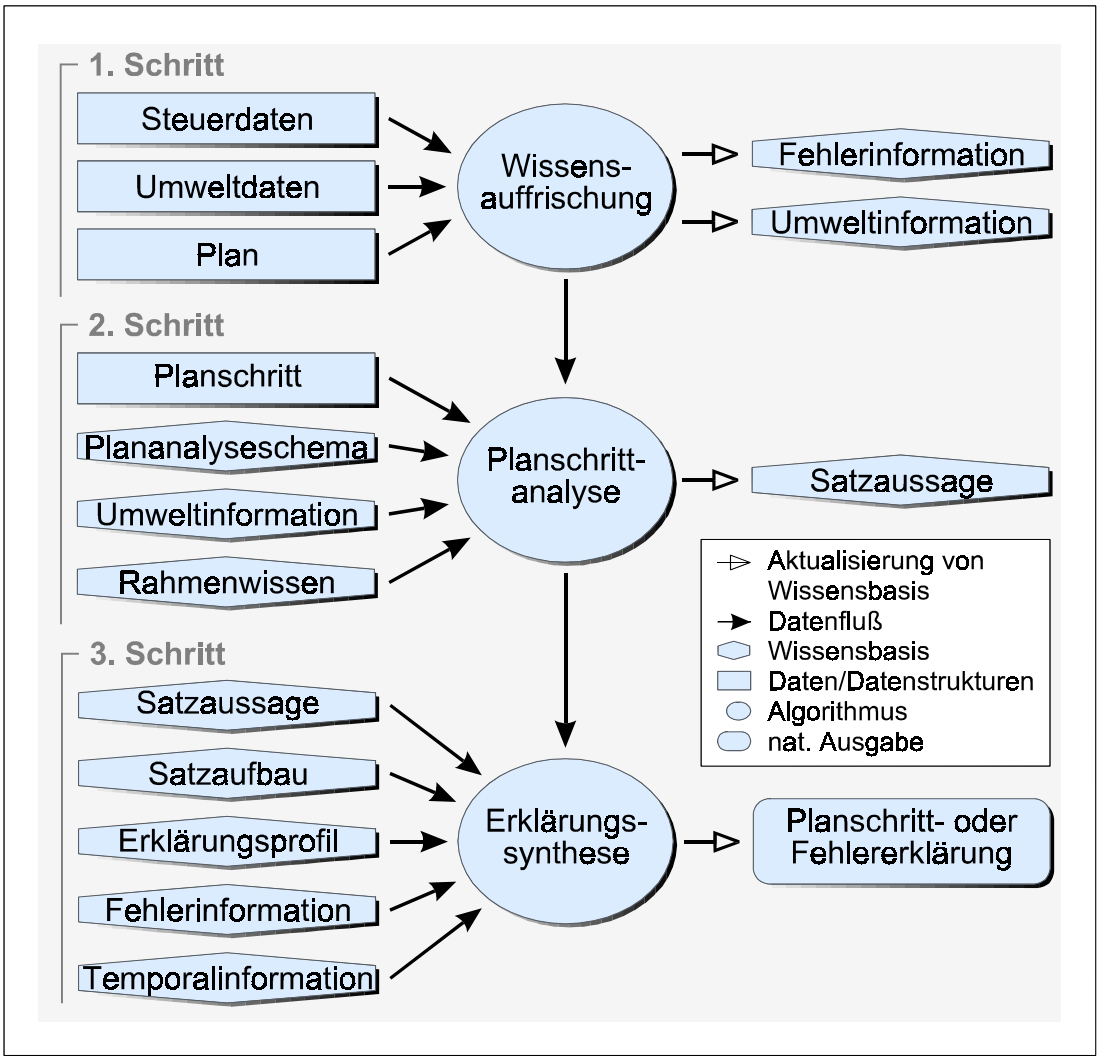

Abbildung 32: Stationen des Erklärungsalgorithmus.

In den sich anschließenden Kapiteln werden die einzelnen Teile des Erklärungsalgorithmus´ genau beschrieben.

4.5.1 Wissensauffrischung

Durch die Steuerdatei teilt das Robotersystem dem Planerklärungsalgorithmus mit, woran der Roboter gerade arbeitet. Da diese Datei nur dann erzeugt wird, wenn der Roboter etwas tut, wird regelmäßig nach einem bestimmen Zeitabschnitt (1 Sekunde) überprüft, ob sie vorhanden ist. Beim Erscheinen dieser Datei werden zeitgleich die Umweltdaten und die Planinformation aktualisiert. Handelt es sich beim Inhalt der Steuerdatei um eine Diagnoseinformation, wird die sich anschließende Arbeitsphase der Planschrittanalyse übersprungen und direkt mit der Erklärungssynthese fortgefahren. Ihr liegen dabei unter anderem die Befehlsinformationen des zuletzt vom Roboter abgearbeiteten Planschrittes vor, die in einem Verbrahmen gespeichert wurden. Ist der Inhalt der Steuerdatei ein Verweis auf einen Planschritt in einem Anweisungsplan,

dann wird diese Anweisungszeile aus dem Plan gelesen und in der Planschrittanalyse weiterverarbeitet. Wurden alle roboterabhängigen Informationen verarbeitet, wird zur Quittierung die Steuerdatei gelöscht. Damit keine unvorhergesehenen Zustände eintreten können, wird sie bis dahin als „nur lesbar“ gekennzeichnet.

Als letztes wird das Basiswissen um den Zustand des Roboters aktualisiert. Im Fehlerplanerklärungs- oder Diagnosefall wird der Wert der Roboterzustandsvariable mit dem Wert `Fehlerplanabarbeitung`, ansonsten mit dem Wert `Planabarbeitung` belegt.

4.5.2 Planschrittanalyse

Die sequenzenorientierte Plananalyse kann verschiedene Planstrukturen erkennen (siehe Abschnitt 4.1.2). Im Fall einer Fehlersituation wird zunächst die Sektionsinformation interpretiert. Dies ist der Fall, wenn der Anweisungszeiger auf den Anfang einer Sektion verweist. Die sich dort befindende Beschreibung wird mit Hilfe des Rahmenwissens interpretiert, um anschließend daraus eine natürlichsprachliche Erklärung zu erzeugen. Sie wird dem Menschen in der Arbeitsphase der Erklärungssynthese, sofern es das Erklärungsprofil vorsieht, mitgeteilt. Ansonsten wird versucht, mit Hilfe des Anweisungsidentifizierers im jeweiligen Planschritt eine Roboteranweisung zu finden.

Jede Roboteranweisung kann durch ihren Befehlsnamen genau bestimmt werden. Sie sind in den vor dem Systemlauf definierten Plananalyseschemen (siehe Abschnitt 4.4.2.1) hinterlegt. Der Analysealgorithmus versucht nun mit den darin angegebenen Befehlsnamen einen Roboterbefehl zu identifizieren. In dem Fall, daß die Befehlsnamen des Planschritts genau mit den korrespondierenden Angaben der Befehlsnamen eines Plananalyseschemas syntaktisch übereinstimmen, entnimmt die Analysemethode aus ihm, welcher Verbrahmen als Grundlage für die natürlichsprachliche Erklärung dient. Stehen dabei mehrere Verbrahmen zur Auswahl, so wird per Zufall einer ausgewählt.

In den nächsten Arbeitsschritten werden die für die Tätigkeitsbeschreibung gebrauchten Angaben aus dem Planschritt extrahiert. Die Füllbefehle des passenden Plananalyseschemas liefern die dafür benötigten Informationen. In ihnen ist durch Zeichenfolgen (Sequenzen) genau die Start- und Endposition im Planschritt angegeben, zwischen denen eine erklärungsrelevante Information gefunden werden kann. Die auf diese Art gefundenen Koordinaten und Lageinformationen der beteiligen Objekte werden direkt an die entsprechende Stelle des Verbrahmens eingetragen. Für die auf diese Weise ermittelte Objekt- und Manipulatorinformation muß erst das natürlichsprachliche Pendant gefunden werden. Dafür stehen die Objektinformationsstruktur und das Rahmenwissen zur Verfügung. Die natürlichsprachlichen Objektinformationen, wie Bezeichnung, Artikel, Personalpronomen und so weiter werden nun im zugehörigen Verbrahmen gespeichert.

Tritt bei der Bestimmung des Verbrahmens oder bei der Abarbeitung der Füllbefehle ein Fehler auf, dann wird der vordefinierte Fehler-Verbrahmen als Grundlage der natürlichsprachlichen Erklärung verwendet. Die daraus erzeugte natürlichsprachliche Erklärung informiert den Benutzer über den Fehler, der aufgetreten ist.

4.5.2.1 Beispiel einer Planschrittanalyse

Als Beispiel dient die Anweisung für den Roboter KAMRO, einen Bolzen (**spacer:1**) des CAB zu greifen. Der dabei zugrundeliegende Roboterbefehl beziehungsweise die Planzeile[37] wird im folgenden gezeigt:

```
eodef (pick, 3, [
                  [agent_name, red],
                  [gripobj_id, [workpiece, spacer:1]]
                ]
      )
```

Das Wissen über Roboterbefehle sei durch das folgende Plananalyseschema definiert:

```
('pick' ''
  ("GREIFEN" "AUFHEBEN" "PACKEN" "NEHMEN") (
    ('maniadj'      (',' ',' ', ') ']')
    ('obj'('[gripobj_id,' ', ') ']')
  )
)
```

Da die Planschrittanalyse nur die Information vom oben definierten Plananalyseschema hat, wird versucht, in der Planzeile den Befehlsidentifizierer **pick** zu finden. Da dies offentsichtlich zutrifft, wird anschließend der Verbrahmen AUFHEBEN für die weitere Verarbeitung zufällig ausgewählt. Daraufhin werden nacheinander die Füllbefehle abgearbeitet. Die Suchanweisung **maniadj** soll aus der Planzeile die Information extrahieren, mit welchem Greifer das Objekt gegriffen wird. Laut Suchvorgabe befindet sie sich zwischen dem Leerzeichen hinter dem dritten Komma und einer schließenden eckigen Klammer. Die extrahierte Information red wird mit dem Rahmenwissen in die natürlichsprachliche Information „rechten“ übersetzt und in die Datenzelle :**maniadj** im Verbrahmen eingefügt. Die Suchanweisung **obj** entnimmt die Information spacer:1 zwischen dem Leerzeichen hinter der Komma, nach der Zeichenfolge »[gripobj_id,« und einer schließenden eckigen Klammer. Das natürlichsprachliche Pendant wird unter Zuhilfenahme des Umweltwissens ermittelt. In der passenden Objektinformationsstruktur sind folgende natürlichsprachliche Informationen gegeben:

```
'(("Bolzen" "Stift")
  ("den" "dem" "der") ("den" "dem" "der"))
  ("ihn" "ihn")
 )
```

Da mehrere natürlichsprachliche Objektbezeichnungen existieren, wird zufällig Bolzen ausgewählt. Sie wird in die Datenzelle :obj, der dazugehörige Artikel im Akkusa-

[37] Der Übersicht halber ist sie nicht auf eine Zeile formatiert.

tiv in :objdet und das Personalpronomen in :ppron des aktuellen Verbrahmens eingetragen. Der gefüllte Verbrahmen sieht danach folgendermaßen aus:

:**name**	'aufheben
:**teller**	"Ich"
:**infinverb**	"aufheben"
:**frontverb**	"hebe"
:**backverb**	"auf"
:**verbadv**	"NONE"
:**obj**	"Bolzen"
:**objdet**	"der"
:**objinfo**	"NONE"
:**objinfodet**	"NONE"
:**ppron**	"ihn"
:**addobj**	"NONE"
:**addobjdet**	"NONE"
:**addobjadj**	"NONE"
:**addobjprep**	"NONE"
:**pos**	"NONE"
:**posprep**	"an der Position"
:**mani**	("Greifer")
:**manidet**	("mit meinem")
:**manidetonly**	"mit einem meiner Greifer"
:**maniadj**	"rechten"
:**manimove**	"NONE"
:**maniprep**	"NONE"
:**manipos**	"NONE"
:**maniwidth**	"NONE"
:**maniadd**	"NONE"
:**whereobj**	"NONE"
:**wheredet**	"NONE"
:**whereadj**	"NONE"
:**whereprep**	"NONE"
:**connectobj**	"NONE"
:**connectdet**	"NONE"
:**connectprep**	"NONE"

4.5.3 Erklärungssynthese

Die Erklärungssynthese verarbeitet im allgemeinen Fall sowohl die Informationen des aktuellen und vorangegangenen Verbrahmens als auch die des Erklärungsprofils. Tritt jedoch bei der Robotertätigkeit ein Fehler auf, so werden an dieser Stelle nach den Direktiven des Erklärungsprofils dem Menschen Diagnoseinformationen vermittelt. Abbildung 33 gibt einen Überblick über die Arbeitsweise der Erklärungssynthese. Insbesondere gibt sie Auskunft über die Wissensbasen, die bei den zwei Arbeitsschritten,

der Vervollständigung der Eingangsdaten und der Generierung der Tätigkeits- oder Fehlerbeschreibung, eingesetzt werden. Daneben zeigt sie den Datenfluß beziehungsweise den Einsatz beteiligter Datenstrukturen, die als Schnittstelle der einzelnen Synthesemodule fungieren.

4.5.3.1 Robotertätigkeit

Vor der Erzeugung der natürlichsprachlichen Erklärung wird überprüft, ob alle laut Erklärungsprofil gebrauchten Informationen im Verbrahmen enthalten sind. Fehlende Einträge, wie zum Beispiel Lage oder Positionsangaben werden ergänzt, falls sie vom Benutzer verlangt werden und in den entsprechenden Wissensrepräsentationen vorhanden sind. Wird laut Erklärungsprofil in der natürlichsprachlichen eine natürlichsprachliche räumliche Beschreibung des Handlungsobjektes verlangt, so muß sie auf der Grundlage des Umweltwissens bestimmt werden.

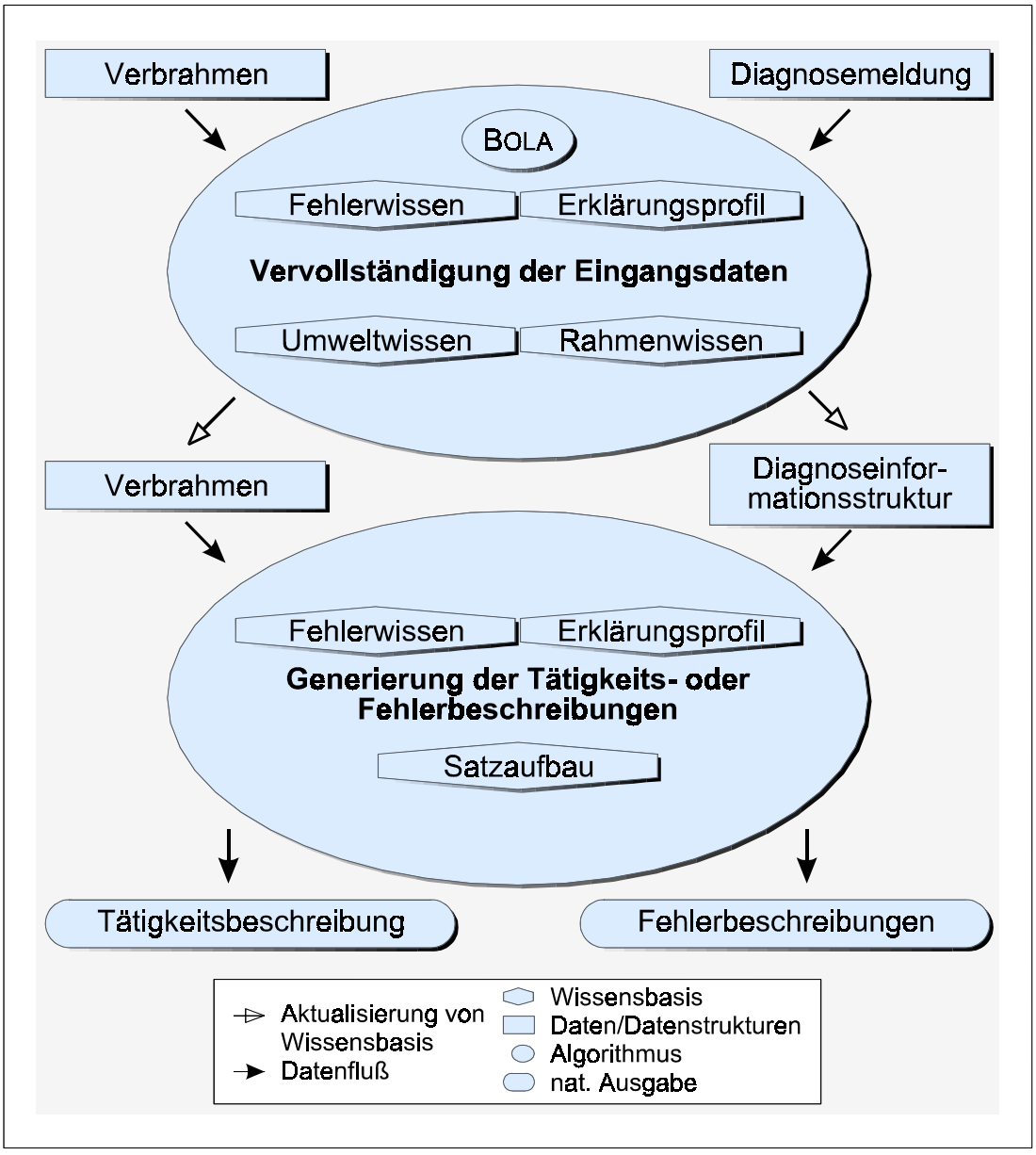

Abbildung 33: Arbeitsweise der Erklärungssynthese.

Das System BOLA liefert die dafür notwendigen Verfahren (siehe Kapitel 2.2). Das Berechnungsergebnis ist ein räumlicher Ausdruck, bestehend aus einer räumlichen Relation und einem Referenzobjekt. Die Bezeichnungen des ermittelten Referenzobjektes und der räumlichen Relation müssen auf natürlichsprachliche Bezeichnungen abgebildet werden, um dann im Verbrahmen gespeichert werden können. Die dafür notwendigen Angaben befinden sich in der dazugehörigen Objektinformationsstruktur des Umweltwissens und des Rahmenwissens.

Anschließend daran wird der vollständig gefüllte Verbrahmen durch die Erklärungssynthese interpretiert. Dabei wird eine Kontextanalyse durchgeführt, die feststellt, ob das Handlungsobjekt in bezug zur letzten Robotertätigkeit ein anderes ist. Falls dem nicht so ist, kann, sofern das Erklärungsprofil dies vorsieht, die Tätigkeitsbeschreibung des Roboterbefehls im Fließtext-Modus generiert werden. Dies erlaubt es, die aktuelle Erklärung mit dem Wort `und` an die vorangegangene anzuknüpfen und die Bezeichnung des Handlungsobjektes durch seinem Personalpronomen zu ersetzten. Bevor diese erzeugt werden kann, muß aus der Menge der Satzformen (siehe Abschnitt 4.2.1) zufällig eine ausgewählt werden, die für die Beschreibung der aktuellen Robotertätigkeit in Frage kommt. Welche Information des Verbrahmens zum Füllen dieser Satzform schließlich verwendet wird, ist durch das Erklärungsprofil bestimmt. Grundsätzlich werden jedoch das Verb und das Aktionsobjekt, sofern die Robotertätigkeit eine einfache Objektmanipulationshandlung ist, in diese Satzform übertragen. Handelt es sich um eine Bewegen-Handlung, wird das natürlichsprachliche Pendant des bewegten Roboterteils in die Satzform eingetragen.

Bei der komplexen Objektmanipulationshandlung besteht die minimale Bestückung dieser Satzform aus Verb, Aktionsobjekt und Berührungsobjekt. Alle anderen Angaben, wie Lage, Koordinaten, Objektarbeitsplatz und Referenzobjekt, werden nicht unbedingt zur minimalen Erklärung gebraucht. Ihr Auftreten in der Tätigkeitsbeschreibung kann der Benutzer frei wählen. Für die Ausschmückung der Erklärungen mit Adjektiven der Zeit steht das Wissen der Temporalinformation zur Verfügung.

4.5.3.2 Beispiel einer Erklärungssynthese

Es wird nun davon ausgegangen, daß die vorbereitenden Schritte der Wissensauffrischung und Planschrittanalyse beendet sind. Der zugrundeliegende Verbrahmen `AUFHEBEN` ist im Unterabschnitt 4.5.2.1 vorgestellt worden. Die Einstellungen des Erklärungsprofils, die den Inhalt der Tätigkeitsbeschreibung beeinflussen, zeigt Abbildung 34.

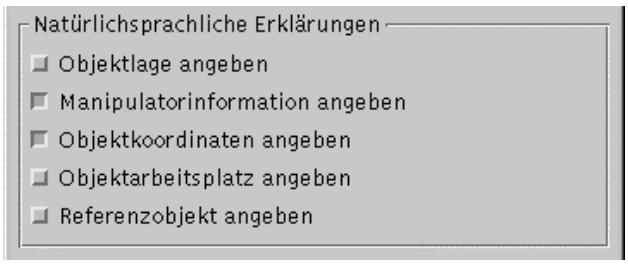

Abbildung 34: Abschnitt des Erklärungsprofils.

Betrachtet man den Verbrahmen in Unterabschnitt 4.5.2.1, so stellt man fest, daß die Angaben über *Objektlage*, *Objektkoordinaten* noch nicht eingetragen sind. Diese Informationen müssen mittels des Objektwissens gewonnen werden und in den aktuellen Verbrahmen eingetragen werden.

Daran anschließend wählt die Erklärungssynthese eine mögliche Satzform für die Tätigkeitsbeschreibung aus. Die in diesem Beispiel verwendete, zufällig ausgewählte Satzform ist:

<Adverb der Zeit> <Verbpräfix> ich <Akkusativobjekt> <Modalergänzung im Dativ> <Adverb> <Lokal- und Modalergänzung im Dativ oder im Akkusativ> <Verbsuffix>.

In den sich daran anschließenden Arbeitsschritten werden die Platzhalter dieser Satzform nacheinander mit natürlichsprachlichen Informationen aufgefüllt. Die dafür benötigten Angaben werden vom Basiswissen bereitgestellt. Nach dem Einfügen vom `<Adverb der Zeit>` und dem `<Verbpräfix>` (siehe :**frontverb** im Verbrahmen) ergibt sich folgendes:

Jetzt hebe ich <Akkusativobjekt> <Modalergänzung im Dativ> <Adverb> <Lokal- und Modalergänzung im Dativ oder im Akkusativ> <Verbsuffix>.

Danach werden `<Akkusativobjekt>` und `<Modalergänzung im Dativ>` (siehe :**objdet**, :**obj**, :**objinfodet** und :**objinfo** im Verbrahmen) eingesetzt. Da kein `<Adverb>` (siehe :**verbadv** im Verbrahmen) vorhanden ist, wird dieser Platzhalter ersatzlos gestrichen. Damit ergibt sich folgendes Bild der Satzform:

Jetzt hebe ich den Bolzen mit der Seitenlage s1 <Lokal- und Modalergänzung im Dativ oder im Akkusativ> <Verbsuffix>.

In den beiden letzten Arbeitsvorgängen werden nacheinander `<Lokal- und Modalergänzung im Dativ oder im Akkusativ>` und `<Verbsuffix>` (siehe :**posprep**, :**pos**, :**manidet**, :**maniadj**, :**mani** und :**backverb** im Verbrahmen) in die Satzform eingefügt. Die dadurch fertiggestellte Handlungsbeschreibung wird im folgenden gezeigt:

Jetzt hebe ich den Bolzen mit der Seitenlage s1 an der Position 10, -10,0 mit meinem rechten Greifer auf.

4.5.3.3 Fehlerbeschreibung

Im Falle, daß der Wissensauffrischung eine Diagnoseinformation geschickt wird, werden relevante Informationen, die Hinweise über Hergang des Fehlers enthalten können, in der Diagnoseinformationsstruktur abgelegt. Welche das in einzelnen sein können, kann in Abschnitt 4.3.2 eingesehen werden. Zudem wird der zuletzt gebrauchte Verbrahmen, sofern er vorhanden ist, gesichert. Darin sind Informationen über die Robotertätigkeit enthalten, bei der ein Fehler festgestellt wurde. Daran anschließend folgt die Interpretation des Erklärungsprofils, worin Angaben gemacht sind, wie der Mensch über Fehler informiert werden will. Ist die Option des Erklärungsprofils *Erklärung beim Eintreten von Fehlern* aktiviert, so wird der Benutzer mit der Aussage „`Es ist ein Fehler eingetreten`!“ über das Vorhandensein einer Anomalie informiert. Die Op-

tion *Fehlerstelle im Plan ausgeben* erweitert die Nachricht um den Planschritt, in dem sich der fehlerhaft ausgeführte Roboterbefehl befindet. Ein mögliches Beispiel dafür ist: „Im Testplan Nr.1 ist bei der Abarbeitung des 5. Schrittes ein Fehler aufgetaucht!“. Diese Meldungen sind im Fehlerwissen definiert. Durch die Aktivierung der Option *detaillierte Erklärungen beim Eintreten von Fehler* wird mit den Angaben des letzten Verbrahmens und einer dazu passenden Fehlersatzstruktur eine genauere Erklärung durch die Erklärungssynthese erzeugt. Um ein Beispiel dafür zu geben, wie eine solche Erklärung aussieht, stelle man sich eine Situation vor, in der bei der Abarbeitung der im Unterabschnitt 2.4.2.3 vorgestellten Roboteranweisung (**pick**...) ein Fehler aufgetreten ist. Das dafür verwendete Erklärungsprofil sei dasselbe wie in vorigem Abschnitt. Eine dazu erzeugte Fehlersituationsbeschreibung sieht folgendermaßen aus:

„Es ist ein Fehler aufgetaucht, als ich den Bolzen mit der Seitenlage s1 an der Position 10, -10, 0 mit meinem rechten Greifer aufheben wollte.“

In den sich anschließenden Schritten wird die Diagnoseinformationsstruktur um die natürlichsprachlichen Pendants der Diagnoseinformation ergänzt. Die Angaben über Anweisung und Unteranweisung liegen zu diesem Zeitpunkt nur in der Roboteranweisungssyntax vor. Daraus müssen noch natürlichsprachliche Erklärungen mit Hilfe des Rahmenwissens generiert werden. Da alle für die Erklärungssynthese benötigten Daten bereits vorhanden sind, bedarf es dafür keiner erneuten Wissensauffrischung. Die fertigen Beschreibungen der Anweisung und der Unteranweisung werden in die dafür vorgesehenen Datenzellen der Diagnoseinformationsstruktur eingetragen.

Im nächsten Arbeitsgang werden aus den vom Robotersystem codierten Meldungen über Fehler, Ursache und Fehlerbehebung mittels der Fehlertranslationstabelle und den vorliegenden Objektinformationsstrukturen natürlichsprachliche Erklärungen erzeugt, die in die entsprechenden Datenzellen der Diagnoseinformationsstruktur gespeichert werden. Dies erst erlaubt die Beantwortung von Fragen des Menschen bezüglich der eingetretenen Fehlersituation.

4.6 Natürlichsprachliche Fragen und Befehle

Um verstehen zu können, welche Eingaben für ROPLEX von Bedeutung sind, bedarf es einiger Erklärungen der Arbeitsweise des Steueragenten. Er entscheidet, ob am Ende ROPLEX oder RIACS die Eingabe weiterverarbeitet.

Der Steueragent läßt die Aussagen des Menschen parallel zu möglichen Prozessen durch RIACS interpretieren. Das Ergebnis wird in der Eingabeinformation abgelegt. Darin befinden sich syntaktische Informationen der Eingabe, die zu deren Identifizierung gebraucht werden. Konkret sind dies detaillierte Fakten über Verb, Subjekt, Objekt, Fragewörter und Satzendezeichen. Anhand dieser Informationen kann eindeutig bestimmt werden, um welchen Typ von Eingabe es sich dabei handelt. Generell werden

in RONTRA nur Bestimmungsfragen [Dreyer & Schmitt 96] erlaubt, das heißt Fragen mit Fragewort.[38]

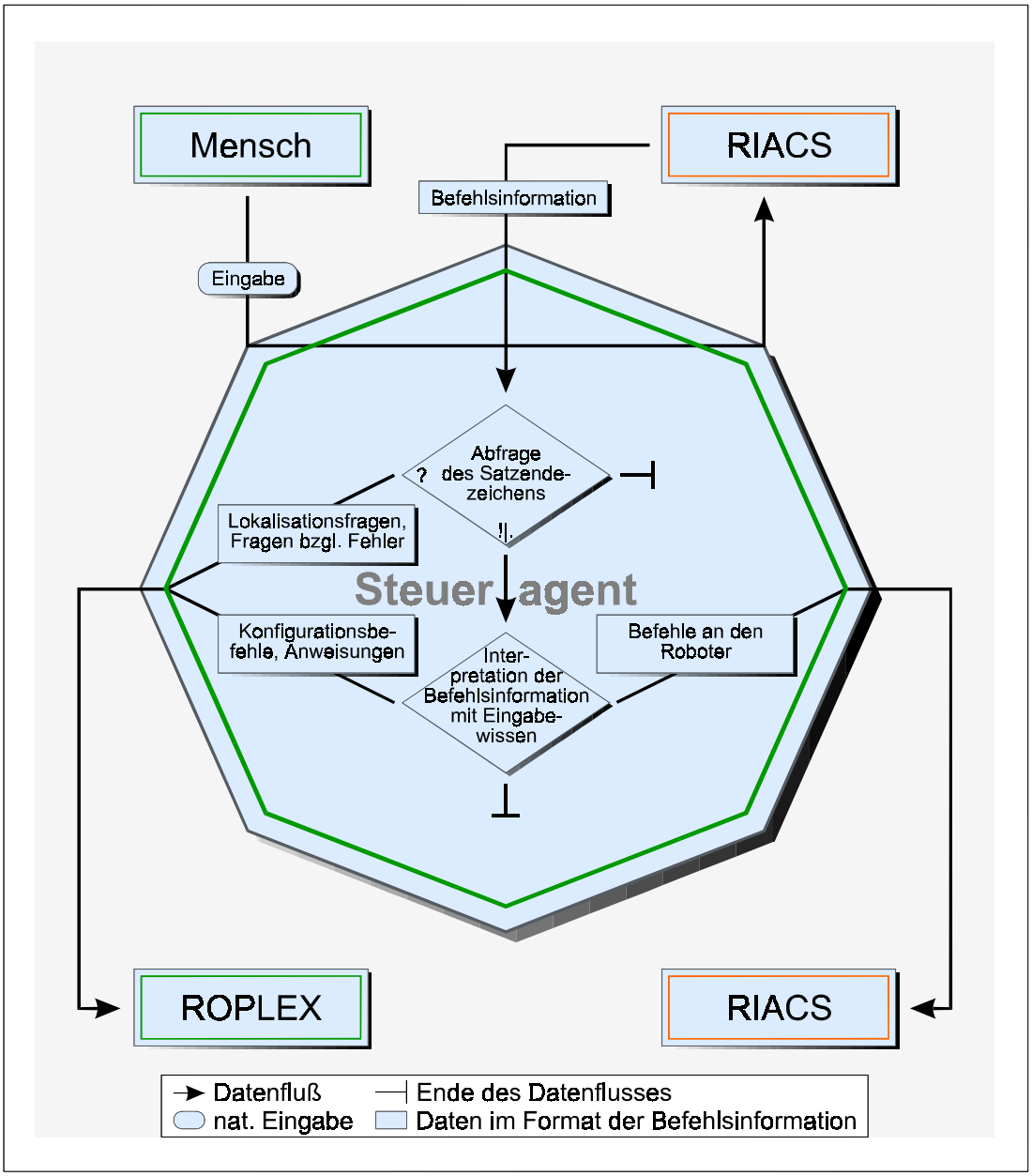

Abbildung 35: Verarbeitung von natürlichsprachlichen Eingaben vom Steueragent.

Zur Typisierung der Eingabe wird zuerst das Satzendezeichen untersucht. Ist es ein Fragezeichen, wird die Eingabeinformation an ROPLEX weitergeleitet und auf das Fragewort `Wo` untersucht. Trifft dies zu, wird die Eingabeinformation nach dem Objekt untersucht, nach dem gefragt wird. Andernfalls wird die Eingabe so behandelt, als wäre sie eine Frage bezüglich einer Fehlersituation. Ist das Satzendezeichen ein Ausrufezeichen, erlaubt RONTRA mehrere Eingabetypen, die anhand des Verbs und des Objektes unterschieden werden. Befehle an den Roboter werden durch die in [Jung 97] beschriebenen Verben identifiziert. Diese Art der Eingabe wird von RIACS weiterverarbeitet. Andernfalls kann es sich entweder um Konfigurationsbefehle oder um Anweisungen in

[38] Beispiele für Fragewörter sind: Wo, Wie, Weshalb oder Warum.

einer Fehlersituation[39] handeln, die von ROPLEX weiter verarbeitet werden. Zur genauen Identifizierung beziehungsweise Unterscheidung der Eingabe steht ROPLEX das Eingabewissen (siehe Abschnitt 4.4.1) zur Verfügung. In Abbildung 35 wird das Modell dieses Datenflusses angezeigt. Wie daraus zu entnehmen ist, kann die Bearbeitung der Eingabe in bestimmten Fällen nicht fortgeführt werden. Dies ist zum einem der Fall, wenn die Abfrage, ob das Satzendezeichen ein Frage- oder Ausrufezeichen ist, keine eindeutigen Ergebnisse liefert. Zum anderen, wenn die Interpretation der Eingabeinformation mit dem Eingabewissen fehlschlägt. In diesen Fällen wird eine Systemnachricht generiert, die der Benutzer im Tracefenster von RONTRAGUI (siehe Abschnitt 6.2.1) einsehen kann.

4.6.1 Eingaben bei Fehlersituationen

Korrigiert der Roboter einen Fehler, kann der Benutzer in Fragen oder Anweisungen Informationen über die Ursache oder die Lösung erlangen. Je nach Fragekontext sind passende Erklärungen im Fehlerwissen (siehe Abschnitt 4.4.1) gespeichert.

Fordert der Benutzer außerhalb einer Fehlersituation Angaben über einen vermeintlichen Fehler, wird er durch im Rahmenwissen definierte Hilfsmeldungen[40] darauf hingewiesen, daß kein Fehler vorliegt.

Ist eine Fehlersituation gegeben, wird die Eingabeinformation mittels der Eingabeidentifizierer überprüft. Kann die Eingabe interpretiert werden, wird die dazu entsprechende Information aus der Diagnoseinformationsstruktur ausgelesen und dem Benutzer präsentiert. Die gültigen Eingaben bei Fehlersituationen können im Abschnitt 6.1.3 in Erfahrung gebracht werden.

4.6.2 Lokalisationsfragen

Für die Beantwortung von Lokalisationsfragen steht ROPLEX das Wissen über den Satzaufbau zur Verfügung. Darin finden sich die benötigten Antwort-Satzmasken (siehe Abschnitt 4.4.1). Die akzeptierte Form von Lokalisationsfragen ist:

`Wo` <**`Verb`**> <**`Nominativobjektphrase`**>?

Dabei ist das **`Verb`** ein Element der Menge {`ist`, `steht`, `befindet sich`, `liegt`}. Die **`Nominativobjektphrase`** besteht aus dem gesuchten Objekt und dem dazugehörigen Artikel.

Die Objektinformation der zugrundeliegenden Eingabeinformation wird mit den Objekten des Szenarios, die in der Objektinformationsstruktur gespeichert sind, verglichen. Wird eine Instanz des fraglichen Objektes gefunden, so wird dem System BOLA der Auftrag erteilt, für dieses Objekt eine räumliche Beschreibung, bestehend aus einem Referenzobjekt und Relationen zu berechnen. Andernfalls wird der Benutzer davon in Kenntnis gesetzt, daß sich das fragliche Objekt nicht in dieser Umgebung befin-

[39] Eingaben mit der Satzform des Imperativs, wie zum Beispiel: „`Nenne mir die Fehlerusache`!“.

[40] Zum Beispiel: „`Es liegt kein Fehler vor`!“

det. Die Antworten von BOLA müssen mit Hilfe des Objekt- und Rahmenwissens mit natürlichsprachlichen Bezeichnungen assoziiert werden. Die auf diesem Wege ermittelte natürlichsprachliche Objektbezeichnung und Präposition werden in eine Antwort-Satzmaske eingefügt, die dann dem Benutzer als Antwort auf die Lokalisationsfrage gegeben wird. Ihr Aufbau wird nachstehend präsentiert:

<**Nominativobjektphrase**> <**Verb**> <**Prep**> <**Dativobjektphrase**>.

Die **Dativ-** und **Nominativobjektphrase** bestehen jeweils aus dem gesuchten Objekt und dem dazugehörigen Artikel. Das **Verb** ist ein Element der Menge {`ist`, `steht`, `liegt`, `befindet sich`}. Die Präposition **Prep** ist ein Element der Menge {`vor`, `hinter`, `links von`, `rechts von`, `bei` , `nahe an`, ...}.

4.6.3 Konfigurationsbefehle

Konfigurationsbefehle geben dem Menschen die Möglichkeit, das Erklärungsprofil zu verändern. Dies kann er auch mit einem dafür vorgesehenen Dialogmenü (siehe Abschnitt 6.2.2).

Nachdem ausgeschlossen wurde, daß es sich um keine Eingabe bei Fehlersituationen handelt, wird sie auf Kombinationen von Eingabeidentifizierern untersucht, die für Konfigurationsbefehle im Eingabewissen definiert sind. Die in ROPLEX gültigen Eingaben können im Abschnitt 6.1.3 eingesehen werden.

Wurde die natürlichsprachliche Eingabe als gültiger Konfigurationsbefehl identifiziert, wird der betroffene Teil des Eingabeprofils dementsprechend geändert. So würde beispielsweise durch die vorher gegebene Konfigurationsbefehle: „`Erkläre ohne Objektkoordinate`!“ und „`Erkläre ohne Objektlage`!“ die am Ende von Unterabschnitt 4.5.3.2 gezeigte Erklärung folgendermaßen aussehen:

`Jetzt hebe ich den Bolzen mit meinem rechten Greifer auf.`

4.7 Systemimplementation und Integration

In diesem Abschnitt werden die implementatorischen Details der von mir entworfenen und realisierten Module ROPLEX, Steueragent und RONTRAGUI genannt.

Für den Entwurf des Gesamtsystems RONTRA wurde die objektorientierten Analyse- und Designmethoden ([Sommerville 96], Seite 215) eingesetzt. Sie basieren auf der Idee des Zurückhaltens von Informationen [Parnas 72]. Dadurch wurde eine bessere Übersicht hinsichtlich der Funktionsweise der einzelnen Agenten erreicht. Für den Entwurf der einzelnen Agenten wurden Datenflußdiagramme ([Sommerville 96], Seite 215) angewandt, da es mit ihnen einfacher ist, Datenverarbeitungsprozesse darzustellen.

Der Kern von RONTRA spiegelt sich im Steueragenten wieder. Er regelt den eingehenden und ausgehenden Datenstrom. Die für zusätzliche Analysen und Berechnungen benötigten Sekundärsysteme, wie zum Beispiel BOLA oder SB-PATR, sind in der Sprache

COMMON LISP [Steele 90] verfaßt. Es liegt daher nahe, daß der Steueragent auch in dieser Sprache realisiert ist. Damit ist gewährleistet, daß Probleme hinsichtlich berechnungsintensiver Datenkonvertierung weitgehend ausgeräumt sind. Zur besseren Gliederung der Informationsrepräsentationen wurden objektorientierte Datenstrukturen verwandt, die im LISP-Sprachstandard CLOS definiert sind. Für die Erfassung und Ausgabe von Daten wurde der Steueragent mit zweierlei Schnittstellen ausgestattet. Er ist in der Lage, Daten über eine Datei- oder TCP-/IP-Schnittstelle mit Robotersystemen auszutauschen. Dies erweist sich gerade dann als Vorteil, wenn Serviceroboter aufgrund ihres Einsatzgebietes über wenig Platzreserven verfügen, um rechenstarke Hardware mitzuführen. Mit solchen Robotern kann der Steueragent mit einer Funk-TCP-/IP-Verbindung kommunizieren.

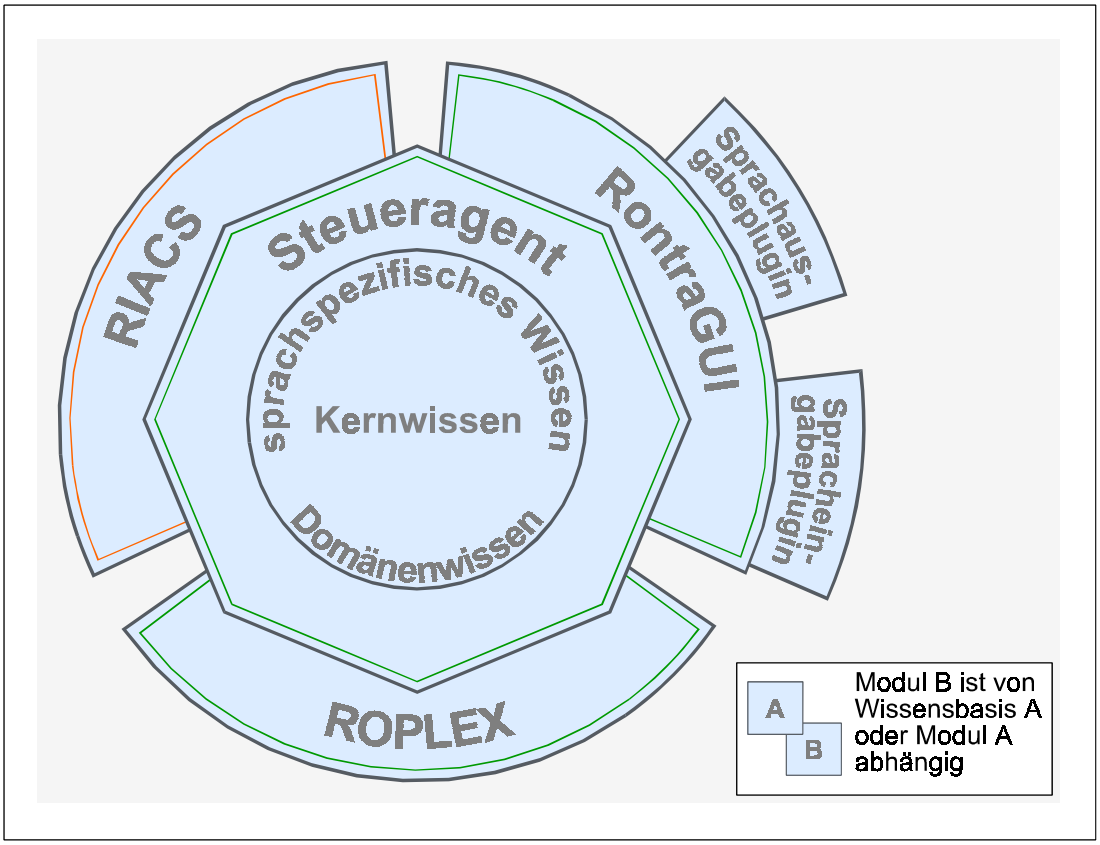

Abbildung 36: Abhängigkeiten zwischen den einzelnen Modulen und Wissensbasen von RONTRA.

Die verschiedenen Funktionen von RONTRA sind auf die beiden Agenten RIACS und ROPLEX aufgeteilt. Da sie möglichst schnell Daten verarbeiten sollen, wurden sie ebenfalls in der Sprache LISP verfaßt, um unnötige Datenkonvertierungen, die Zeit beanspruchen würden, auszuschließen. Ein Vorteil, der sich unmittelbar aus dieser Tatsache ergibt, ist, daß man für die Agenten des Kernsystems ein dediziertes Rechnersystem verwenden könnte, das besonders für die Verarbeitung von LISP-Prozessen geeignet ist, um somit deren Verarbeitungsgeschwindigkeit zu erhöhen. Bei der Planung von RONTRA wurde darüber hinaus darauf Wert gelegt, daß jeder einzelne Agent autonom, das heißt unabhängig von den anderen, seine spezielle Aufgabe erfüllen kann. Dies hilft bei der Einsparung von Ressourcen[41], da so RONTRA individuell an die Gegebenheiten

[41] Damit sind Speicherbelegung und Recherauslastung gemeint.

des Robotersystems angepaßt werden kann. Es reicht beispielsweise aus, den Steueragenten mit dem Plan- und Umwelterklärungsmodul zu verbinden, um das Robotersystem dahingehend zu erweitern, daß es seine Umwelt und seine Tätigkeiten erklären kann. Abbildung 36 zeigt den hierarchischen Aufbau des Gesamtsystems und visualisiert die Abhängigkeiten zwischen den einzelnen Modulen und den globalen Wissensbasen. Eine Erweiterung der Kommunikationsfähigkeiten des Kernsystems ist durch die Dialogschnittstelle RONTRAGUI realisiert.

Sie erlaubt dem Benutzer sämtliche Parameter zu manipulieren, die den Arbeitsverlauf der Kernsystemmodule beeinflussen. Darüber hinaus gibt sie eine Übersicht über die Ausgabe der einzelnen Agenten. Weiterhin bietet sie einerseits die Möglichkeit, Erklärungen des Roboters an ein externes Modul weiterzuleiten, um sie beispielsweise in gesprochene Sprache umzuwandeln und wiederzugeben, andererseits können gesprochene Befehle und Fragen, die von einen externen Modul in ein akzeptiertes Format umgewandelt worden sind, an die jeweiligen Systeme weitergeleitet werden. Da diese Benutzeroberfläche möglichst flexibel eingesetzt werden soll, wurde sie in der Programmiersprache JAVA [Gosling et al. 97] implementiert, die ein Einsatz auf verschiedenen Rechnerplattformen garantiert. Der Datenaustausch zwischen RONTRAGUI und RONTRA ist durch eine TCP-/IP-Schnittstelle realisiert.

5 Zusammenfassung und Ausblick

5.1 Zusammenfassung

Mit dem Modul ROPLEX von RONTRA ist es erstmalig prinzipiell möglich, in verschiedenen Sprachen Erklärungen von Umweltmodalitäten und Tätigkeiten aus Anweisungsplänen für Roboter und den dazugehörigen Umweltinformationen zu erzeugen. Die Schnittstelle zum Robotersystem, die für den Wissensabgleich von Umwelt- und Tätigkeitsdaten zuständig ist, kann auf die Datensyntax von verschiedenen Robotern angepaßt werden. Die so gewonnenen Informationen werden in einer sprach- und roboterunabhängigen Datenstruktur gespeichert. Diese Tatsache erlaubt es, ROPLEX ohne großen Aufwand in schon bestehende Robotersysteme (Anwendungdomänen), wie zum Beispiel KAMRO oder CHEW, zu integrieren. Dies wird insbesondere durch das modulare Konzept unterstützt, in dem der Algorithmus, der die Umwandlung von Umwelt- und Tätigkeitsdaten in natürlichsprachliche Erklärungen realisiert, von jeglichen roboter- und sprachabhängigen Informationen getrennt ist. Für die Generierung von natürlichsprachlichen Erklärungen existieren, abhängig vom Typ der Erklärung, gesonderte Wissensbasen, die im folgenden präsentiert sind:

- *Umweltwissen*: Es beinhaltet einerseits die statischen natürlichsprachlichen Bezeichnungen für die in der Roboterwelt vorkommenden Objekte. Andererseits sind hier dynamische Angaben, wie zum Beispiel Position, Drehung und Lage des Objekte gespeichert.
- *Rahmen- und Fehlerwissen*: Hauptsächlich sind hier feststehende natürlichsprachliche Erklärungen zu Ereignissen und feststehenden Fakten des Roboters zusammengefaßt. Darüber hinaus werden je nach Kontext aktuelle Fehlerinformationen für Erklärungen bereitgestellt.
- *Grammatikwissen*: Es weist Tätigkeiten des Roboters eine Informationsgrundlage zu, die neben statischen Angaben Informationen über aktuelle Gegebenheiten aufnimmt. Diese stellen die Grundlage für die Tätigkeitserklärung dar.

Dieses Wissen von ROPLEX ist Bestandteil der Wissensbasen von RONTRA. Darauf können andere Module, wie zum Beispiel RIACS, zugreifen. Für die Interpretation von Plan- oder Umweltdaten, existiert eine gesonderte Wissensbasis, die auch Bestandteil der Wissensbasen von RONTRA ist, in der die syntaktischen Eigenheiten der für die Erklärung benötigten Daten angegeben sind. Die sich anschließende Aufzählung zeigt die

Möglichkeiten, welche Formen der Erklärung ROPLEX aus Plan- und Umweltdaten von Robotern erlaubt:

- *Tätigkeitsbeschreibungen*: Sie sind die Verbalisierung des vom Roboter aktuell ausgeführten Planbefehls. Je nach Roboter können hier mehr oder weniger Informationen über die Tätigkeit enthalten sein. Form und Inhalt kann der Benutzer frei konfigurieren.
- *Fehlerbeschreibungen*: Immer dann, wenn der Roboter mit einer Fehlersituation konfrontiert ist, kann sich der Benutzer über dessen Ursache und Lösung genauer informieren lassen. Je nach Wahl werden solche Beschreibungen automatisch beim Eintreten einer Fehlersituation oder durch Erfragen dem Benutzer mitgeteilt.
- *Antworten auf Lokalisationsfragen*: Durch eine Lokalisationsfrage, wie zum Beispiel: „`Wo ist das Objekt`[42]?“, wird der Benutzer über Position und Lage des Objektes genau informiert.

Dem Benutzer ist eine Konfigurationsschnittstelle bereitgestellt, über die er mit natürlichsprachlichen Befehlen Form und Inhalt von Erklärungen verändert kann.

5.2 Ausblick

Die Realisierung eines flexiblen, modularen Bausteins eines Roboterdialogsystems, der sich um die Erklärung von Umweltmodalitäten und darin stattfindenden Handlungen kümmert und dabei benötigtes Wissen mit anderen Modulen teilen muß, bringt unter anderem organisatorische Probleme hinsichtlich der Implementation mit sich. Da dabei der Schwerpunkt sowohl auf dem roboterunabhängigen Analyse- und Auswertungsverfahren von Informationen, als auch in der Realisierung einer sprachunabhängigen Zwischenrepräsentation von Informationen lag, sind nicht alle möglichen Methoden der Informationsverarbeitung realisiert worden. Die sich daraus ergebenden Einschränkungen betreffen zwar nicht die Stabilität des Systems, beschränken aber dessen flexiblen Einsatz. Sie werden im folgenden kurz angesprochen:

- *Genauere Objektidentifikation*: Wie in Abschnitt 4.6.2 beschrieben, können Lokalisationsfragen zu Objekten des Roboterszenarios gestellt werden. Verweist die Bezeichnung des gefragten Objekts auf mehrere Szenarioobjekte, ist nicht gewährleistet, daß mit der Antwort die Lokation der Objektinstanz beschrieben wird, nach der gefragt wurde. Da das System generell nicht wissen kann, welche Instanz (bei Objekten mit derselben Bezeichnung) der Benutzer meint. Für diese Fälle müssen zusätzliche Angaben, wie zum Beispiel Farbe, Oberfläche oder räumliche Ausdrücke in Lokalisationsfragen erlaubt sein, beziehungsweise Rückfragen vom System generiert werden, um die Instanz genauer bestimmen zu können.
- *Lernfähigkeit des Moduls*: Kann ROPLEX während des Erklärungsvorgangs die der Robotertätigkeit zugrundeliegende Eingabeinformation nicht erkennen, so wird ei-

[42] Die Bezeichnung ist abhängig vom Roboterszenario.

ne Fehlermeldung (siehe Abschnitte 4.4.1 und 4.5.2) generiert, die auf dieses Problem hinweist. Besser wäre es, wenn diese Wissenslücke durch einen Dialog mit dem Benutzer direkt geschlossen würde. Auf diese Weise könnte man einem Roboter die Erklärung seiner Tätigkeiten interaktiv beibringen.

- *Multimediale Erweiterungen*: Die textuelle oder gesprochene Erklärungen von Umweltmodalitäten und Tätigkeiten könnten durch geschickte Kombination mit graphischen Elementen verbessert werden.
- *Metaerklärungen*: Zur Zeit ist durch ROPLEX nur die schrittweise Erklärung von Plänen realisiert. Wünschenswert wäre es, wenn zu Beginn einer Planabarbeitung dem Benutzer grob mitgeteilt werden könnte, was der Roboter als nächstes tut. Dieses Konzept ist zwar schon durch die Plansektionen teilweise realisiert, basiert aber auf den vom Robotersystem generierten Kurzbeschreibungen. Durch die Verwendung eines Expertensystems mit speziellen Regeln könnten solche Erklärungen auf der Basis von Plänen erzeugt werden.
- *Benutzermodell*: Durch die Analyse der vom Benutzer gemachten Anfragen und Einstellungsänderungen kann ein Benutzermodell generiert werden. Dadurch kann das System aktiv in Abhängigkeit von der Situation dem Benutzer Erklärungen generieren, wie er sie wünscht. Die Qualität von Erklärungen könnte durch die Verwendung einer Antizipationsrückkopplungsschleife verbessert werden [Schirra 93].

6 Anhang

6.1 Eingabewissen

In diesem Kapitel werden die Wissensbasen vorgestellt, auf die sich in dieser Arbeit häufig bezogen wurde, insbesondere diejenigen, die für die Identifizierung der natürlichsprachlichen Eingabe gebraucht werden.

6.1.1 Eingabeidentifizierer

Mit dem Eingabewissen ist ROPLEX in der Lage, die von RIACS analysierten natürlichsprachlichen Eingaben zu typisieren. Die Eingabeidentifizierer stellen die Grundlage des Eingabewissens dar. In ihnen sind Verben, Adjektive, Objekte und andere Satzteile mit ähnlicher semantischer Bedeutung zusammengefaßt. Um eine im Format der Eingabeinformation vorliegende Eingabe zu interpretieren, ist eine Kombination aus verschiedenen Eingabeidentifizierern, die sogenannte Identifizierungsmaske, notwendig. Nachstehend werden die verschiedenen Eingabeidentifizierer und deren Bedeutung in tabellarischer Form vorgestellt. Beispielsätze oder Wörter wurden in der deutschen Sprache formuliert. Das heißt aber nicht, daß der Inhalt der Eingabeidentifizierer auf sie beschränkt ist.

	Name des Eingabeidentifizierers	Bedeutung
Verben	BeingVerbs (`BV`)	Menge von Verben, die eigentlich in Form von selbständigen Prädikaten in natürlichsprachlichen Eingaben vorkommen. Beispiel: „`Was ist passiert?`“
	CauseVerbs (`CV`)	Menge von Verben, die eine ähnliche Semantik wie das Verb `verursachen` haben.
	DoingVerbs (`DV`)	Menge von Verben, die eine ähnliche Semantik wie das Verb `tun` haben.
	RepairVerbs (`RV`)	Menge von Verben, die eine ähnliche Semantik wie das Verb `reparieren` haben.
	SayVerbs (`SV`)	Menge von Verben, die eine ähnliche Semantik wie das Verb `sagen` haben.
	KnowVerbs (`KV`)	Menge von Verben, die eine ähnliche Semantik wie das Verb `wissen` haben.

	Name des Eingabeidentifizierers	Bedeutung
Verben	BehaviorVerbs (BhV)	Menge von Verben, die für die Identifizierung von Verhaltensanweisungen für ROPLEX gebraucht werden. Zum Beispiel das Verb sein, daß für die Erkennung der Anweisung „Sei ruhig!“ gebraucht wird.
	ExpconfigVerbs (EV)	Menge von Verben, die in einer Konfigurationsanweisung für ROPLEX auftreten können. Beispielsweise das Verb erkläre, daß in der Anweisung „Erkläre mit Koordinatenangaben!“ gebraucht wird.

	Name des Eingabeidentifizierers	Bedeutung
Hauptwörter	ErrcauseObj (EcO)	Hauptwörter, deren Semantik paradigmatisch mit Fehlerwörtern zu tun haben.
	ErrcauseSubj (EcS)	Hauptwörter, deren Semantik paradigmatisch mit Fehlerwörtern zu tun haben.
	ErrgoalSubj (EgS)	Hauptwörter, die für die Erkennung von Anweisungen das Fehlerbehebungsziel zu nennen, gebraucht werden.
	PlanSubj (PS)	Planbezeichnende Hauptwörter.
	PlanChange (PC)	Wörter, die für Planänderung stehen.
	BehaviorNons (BhN)	Wörter, die für die Identifizierung von Verhaltensanweisungen für Roplex gebraucht werden.
	ExpconfigNons (EN)	Allgemeine Wörter, die in einer Konfigurationsanweisung auftreten können.
	ExpconfigNons-flowtext (ENf)	Wörter, die den Umstand der Ausgabe im Fließtextes beschreiben.
	ExpconfigNons-mani (ENm)	Menge von Wörter, die eine Bezeichnung der manipulativen Geräte des Roboters beinhalten. Zum Beispiel: Hand oder Greifer.
	ExpconfigNons-whereobj (ENw)	Bezeichnungen des Bezugsobjektes.
	ExpconfigNons-workplace (ENwp)	Wörter, die eine Bezeichnung des Arbeitsplatzes darstellen.
	ExpconfigNons-coord (ENc)	Wörter, die eine Bezeichnung der Koordinatenangabe darstellen.
	ExpconfigNons-pos (ENp)	Menge von Wörter, die eine ähnliche Semantik wie das Wort Lage haben.
	ExpconfigNons-info (ENi)	Menge von Wörter, die eine ähnliche Semantik wie das Wort Informationen haben.
	ExpconfigNons-error (ENe)	Menge von Wörter, die eine ähnliche Semantik wie das Wort Fehler haben.

	Name des Eingabeidentifizierers	Bedeutung
Adjektive, Präpositionen und Pronomen	BehaviorElem-quiet (`BhEq`)	Pronomen, die eine ähnliche Semantik wie `ruhig` haben.
	BehaviorElem-loud (`BhEl`)	Pronomen, die eine ähnliche Semantik wie `laut` haben.
	ExpconfigElem-pos (`EEp`)	Satzkomponenten, die eine ähnliche Semantik wie das Wort `mehr` haben.
	ExpconfigPrep-pos (`EPp`)	Präpositionen, die eine ähnliche Semantik wie das Wort `mit` haben.
	ExpconfigElem-neg (`EEn`)	Satzkomponenten, die eine ähnliche Semantik wie das Wort `weniger` haben.
	ExpconfigPrep-neg (`EPn`)	Präpositionen, die eine ähnliche Semantik wie das Wort `ohne` haben.
	ExpconfigElem-base (`EEb`)	Satzkomponenten, die eine ähnliche Semantik wie das Wort `normal` haben.
	ExpconfigElem-all (`EEa`)	Pronomen, die eine ähnliche Semantik wie `alles` haben.
	ExpconfigElem-none (`EEno`)	Pronomen, die eine ähnliche Semantik wie `nichts` haben.

	Name des Eingabeidentifizierers	Bedeutung
Fragepartikel	ObjQuest (`OQ`)	Menge von Fragepartikel, die zur Identifizierung von Lokalisationsfragen dienen. Beispiel: `Wo`.
	Questionparticles (`Qp`)	Menge von Fragepartikel, die bei Fragen bezüglich Fehler auftreten können. Beispiele hierfür sind: `Wie`, `Weshalb` und `Warum`.

6.1.2 In ROPLEX definierte Eingabeidentifizierer

Welche Werte die verwendeten Eingabeidentifizierer haben können, ist in der folgenden Auflistung gezeigt. Da die Werte mit eigenen der Eingabeinformation verglichen werden, sind sie abhängig von dem syntaktischen Analysealgorithmus (dem Parser), der in RIACS verwendet wird. So kann es zum Beispiel sein, daß der Parser anstatt der Grundform des erkannten Verbs nur den Wortstamm an RIACS übermittelt. Dahingehend muß auch die verendete Form der Verben oder anderer Satzpartikel auf die Parserausgabe angepaßt werden.

Dieses Verfahren, erlaubt es, Eingaben, die in anderen als der deutschen Sprache formuliert sind und von einem Parser syntaktisch analysiert wurden, zu verarbeiten. In diesem Fall müssen alle Eingabeidentifizierer an die neuen Werte angeglichen werden.

BeingVerbs '("`sei`")
CauseVerbs '("`verursach`" "`tret`" "`komm`")
DoingVerbs '("`tu`" "`mach`" "`anstell`")

RepairVerbs '("beheb" "reparier")
SayVerbs '("nenn" "sag")
KnowVerbs '("laut" "heiss")
BehaviorVerbs '("sei" "sag" "halt")
ExpconfigVerbs '("erklaer" "erlaeuter" "sag")

ErrcauseObj '("fehler" "fehlerursache" "ursache")
ErrcauseSubj '("befehl" "du" "fehler" "es")
ErrgoalSubj '("ziel" "du")
PlanSubj '("plan")
PlanChange '("planaenderung")
BehaviorNons '("maul" "mund")
ExpconfigNons '("angabe" "es")
ExpconfigNons-flowtext '("fliesstext")
ExpconfigNons-mani '("manipulatorangabe")
ExpconfigNons-whereobj '("bezugsobjekt")
ExpconfigNons-workplace '("arbeitsplatz")
ExpconfigNons-coord '("koordinate")
ExpconfigNons-pos '("positionsangabe")
ExpconfigNons-info '("information")
ExpconfigNons-error '("fehler" "fehlerstelle")

BehaviorElem-quiet '("ruhig" "still")
BehaviorElem-loud '("laut" "was")
ExpconfigElem-pos '("mehr" "besser" "genauer")
ExpconfigPrep-pos '("mit" "im")
ExpconfigElem-neg '("weniger" "schlechter" "ungenau")
ExpconfigPrep-neg '("ohne")
ExpconfigElem-base '("normal")
ExpconfigElem-all '("ungenau")
ExpconfigElem-none '("nicht")

ObjQuest '("wo")
Questionparticles '("wie" "was" "warum" "wodurch" "weshalb" "welch" "w-")

6.1.3 Identifizierungsmasken

Eingaben des Menschen, die mittels RIACS syntaktisch untersucht worden sind, können mit Eingabeidentifizierern semantisch typisiert werden. Dazu wird versucht die Werte von relevanten Datenzellen der Eingabeinformation mit Werten aus Eingabeidentifizierern, die zu Identifizierungsmasken zusammengefaßt sind, zu vergleichen. In der sich anschließenden Tabelle werden die Identifizierungsmasken gezeigt, die von ROPLEX für die Eingabeuntersuchung verwendet werden. Darüber hinaus sind darin die mit der Erkennung verbundenen Aktionen vermerkt. Um zu kennzeichnen, daß Werte von Eingabeidentifizierern nicht in der natürlichsprachlichen Eingabe auftauchen dürfen, werden diese Eingabeidentifizierer durchgestrichen dargestellt.

Eingabetyp		Identifizierungsmaske	Beispiel
Anweisung (!)	Fehlerursache	SV, Ec0, ~~EcS~~	Nenne die Fehlerursache!
	Fehleranweisung	SV, EcO, EcS	Sage den Befehl, der den Fehler verursacht hat!
	Fehlerbehebung	SV, ~~EcO~~, ~~EcS~~, EgS	Sage das Fehlerbehebungsziel!
	Fehlerplansektion	SV, PS	Sage mir Dein Plan!

Eingabetyp		Identifizierungsmaske	Beispiel
Frage (?)	Fehlerbeschreibung	Qp, DV, EcS	Was hast Du getan, als der Fehler auftrat?
	Fehlerursache	Qp, CV, EcO, EcS	Was ist die Fehlerursache?
		Qp, CV, EcO, EcS	Warum trat der Fehler auf?
	Fehleranweisung	Qp, CV, EcO, ~~EcS~~	Welcher Befehl hat den Fehler verursacht?
	Fehlerbehebung	Qp, CV, EcO, EgS	Wie wirst Du den Fehler beheben?
	Fehlerplansektion	Qp, KV, PS	Wie lautet Dein Plan?
	Lokalisation	OQ, EcO, ~~BhN~~, ~~EN~~, ~~ENm~~, ~~ENw~~, ~~ENp~~	Wo liegt das Pendel?

Neben diesen Fragen und Anweisungen gibt es noch die Klasse von natürlichsprachlichen Konfigurationsanweisungen. Damit kann das Aussehen der Robotertätigkeitsbeschreibungen verändert werden. Diese speziellen Anweisungen werden in der folgenden Tabelle präsentiert:

Eingabetyp		Identifizierungsmaske	Beispiel (Kamro)
Konfigurationsanweisung (!)	Abschaltung	BhV, BhEq	Sei ruhig!
	Anschaltung	BhV, BhEl	Sei laut!
		EV, BhEl	Sag was!
	Standardeinstellung	EV, EEb	Erkläre normal!
	Maximaleinstellung	EV, EEa	Sag alles!
	Informationen erhöhen	EV, EPp, EEp, ENi	Erkläre mit mehr Informationen!
	Informationen erniedrigen	EV, EPp, EEn, ENi	Erkläre mit weniger Informationen!
	Benachrichtigung bei Planänderung	EV, PC	Sage mir Planänderungen!
	Keine Benachrichtigung bei Planänderung	EV, PC, EEno	Sage mir Planänderungen nicht!

Eingabetyp		Identifizierungsmaske	Beispiel (Kamro)
Konfigurationsanweisung (!)	Erklärungen im Fließtextmodus erzeugen	EV, EPp, ENf	Erkläre mit Fliesstext!
	Kein Fließtextmodus für Erklärungen	EV, EPn, ENf	Erkläre ohne Fliesstext!
	Manipulator	EV, EPp, ENm	Erkläre mit Manipulator!
	Manipulator nicht angeben	EV, EPn, ENm	Erkläre ohne Manipulatorangabe!
		EV, ENm, EEno	
	Arbeitsplatz angeben	EV, EPp, ENwp	Nenne den Arbeitsplatz!
	Arbeitsplatz nicht angeben	EV, EPn, ENwp	Erkläre ohne Arbeitsplatzangabe!
		EV, ENwp, EEno	
	Bezugsobjekt angeben	EV, EPp, ENw	Erkläre mit Bezugsobjekt! Nenne das Bezugsobjekt!
		EV, ENw	
	Bezugsobjekt nicht angeben	EV, EPn, ENw	Erkläre ohne Bezugsobjekt!
		EV, ENw, EEno	
	Objektlage angeben	EV, EPp, ENc	Erkläre mit Lageangaben!
		EV, ENc	
	Objektlage nicht angeben	EV, EPn, ENc	Erkläre ohne Lageangaben!
		EV, ENc, EEno	
	Koordinaten des Aktionsobjektes angeben	EV, EPp, ENp	Erkläre mit Koordinatenangaben!
		EV, ENp	
	Keine Koordinatenangabe des Aktionsobjektes	EV, EPn, ENp	Erkläre ohne Koordinatenangaben!
		EV, ENp, EEno	
	Fehlerangabe	EV, EPp, ENe	Berichte ein Fehlereintritt!
	Keine Fehlerangabe	EV, ENp, ENe, EEno	Berichte eine Fehlersituation nicht!
	Genaue Fehlerangabe	EV, EPp, EEp, ENe	Erkläre den Fehler genau!
	Keine genaue Fehlerangabe	EV, EPp, EEb, ENe	Erkläre den Fehler normal!
	Angabe der Fehlerstelle im Plan	EV, EPp, ENe, PS	Sage die Fehlerstelle im Plan!
	Fehlerstelle im Plan nicht angeben	EV, ENp, ENe, PS	Sage die Fehlerstelle im Plan nicht!

6.2 Die graphische Benutzeroberfläche RONTRAGUI

Um eine große Flexibilität für den Einsatz von RONTRA zu erreichen, sollte das Kommunikationsmodul, das die Schnittstelle zwischen Mensch und Roboter realisiert, möglichst universell einsetzbar sein. Die Ausführbarkeit auf vielen unterschiedlichen Rechnerplattformen, die durch die Verwendung der Programmiersprache JAVA erreicht wurde, sowie die Kommunikation mittels dem TCP/IP – Protokoll sind die Grundlage für den flexiblen Einsatz der Benutzeroberfläche von RONTRA.

Insbesondere wurde hoher Wert auf variable Datenein-/ausgabe gelegt, dadurch wurde unter anderem erreicht, daß Spracherzeugungsprogramme die erzeugten textuellen Erklärungen und Antworten benutzen können, um sie in gesprochene Sprache umzuwandeln. Zudem bietet dieses Konzept an, Aussagen eines Benutzers die mittels eines Spracherkennungsprogramms in eine textuelle Form umgewandelt wurden, an RONTRA weiterzuleiten.

6.2.1 Eigenschaften und Funktionalität

In Abbildung 37 ist die Benutzerschnittstelle in ihrem Ausgangszustand gezeigt. Im

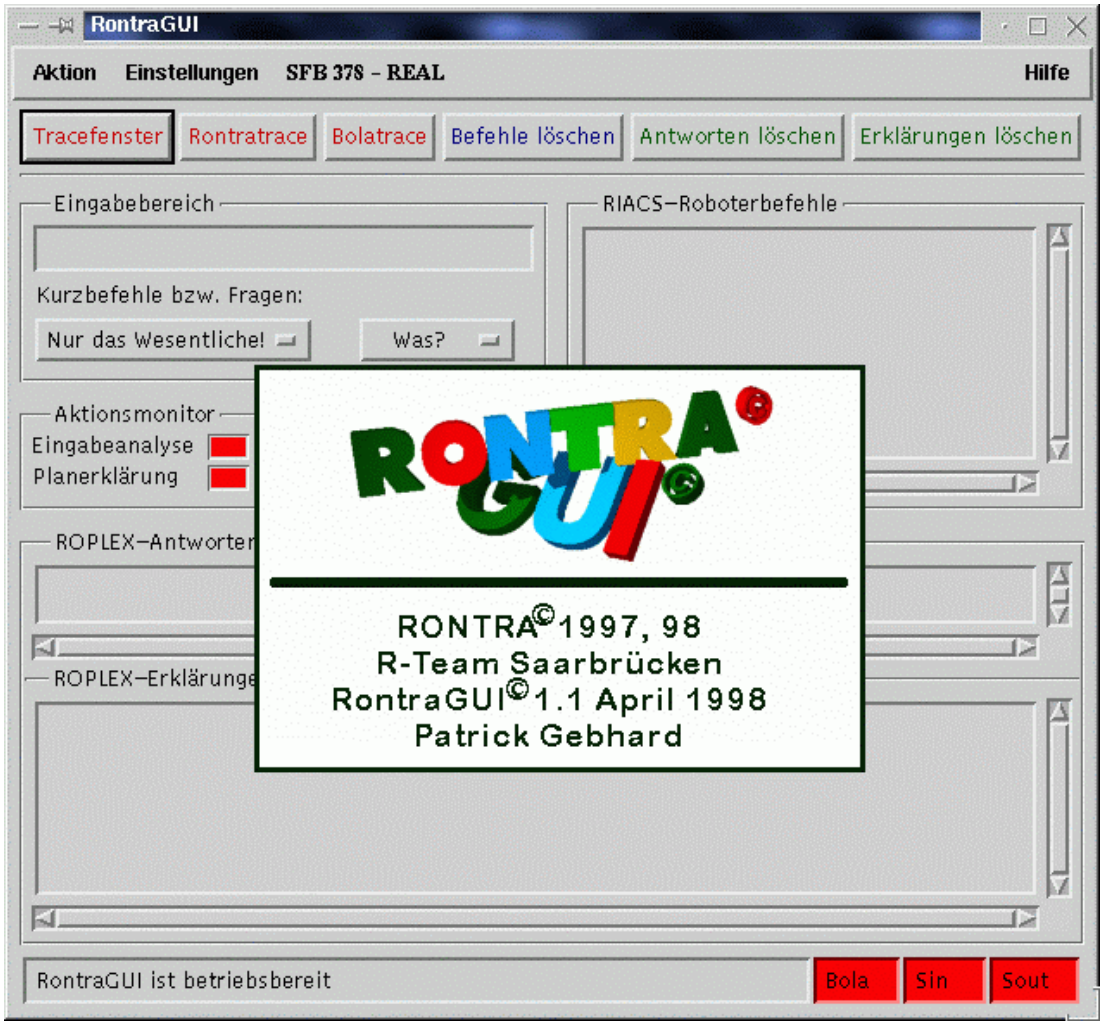

Abbildung 37: Die graphische Benutzerschnittstelle RONTRAGUI.

oberen Bereich befindet sich die Menüzeile, die Einstellungsdialoge, Aktivierungsvorgänge und einen Hilfedialog anbietet. Direkt unterhalb sind elementare Aktionen in einer Buttonleiste zusammengefaßt. Der mittlere Teil beherbergt unter anderem den Eingabebereich, der es dem Benutzer ermöglicht, Anweisungen und Fragen über die Tastatur an den Roboter zu stellen. Im Ausgangszustand ist dieser Bereich grau dargestellt, da diese Option erst aktiviert wird, wenn sich die Benutzeroberfläche mit dem Kernsystem verbunden hat. Daneben ist ein Bereich angesiedelt, in dem die von RICAS generierten Roboterbefehle angezeigt werden. Zudem sind hier zwei Ausgabebereiche von ROPLEX vertreten, worin einerseits Anworten auf Fragen und andererseits Erklärungen von Robotertätigkeiten anzeigt werden. Zwischen dem Eingabebereich und den Ausgabebereichen liegt die Sektion des Aktionsmonitors. Er zeigt die Aktivitäten der einzelnen Agenten von RONTRA an. Dies ist durch Statusleuchten realisiert, die je nach Art der Aktivität eine andere Farbe annehmen. Rot steht für einen inaktiven Zustand. Orange signalisiert, das sich der Agent in einer Wartestellung befindet. Grün wird bei aktiven Agenten benutzt. Am unteren Rand des Hauptfensters von RONTRAGUI befindet sich eine Statuszeile, in der mit Nachrichten über den augenblicklichen Status der Benutzeroberfläche informiert wird. Zudem wird angezeigt, welche Zusatzsysteme gerade aktiv (Grün hinterlegt), inaktiv (Orange hinterlegt) oder gar nicht vorhanden sind (Rot hinterlegt). Im Augenblick sind das die Sprachein-/ausgabeprogramme (Sin, Sout) sowie das System BOLA, das zur Berechnung von räumlichen Ausdrücken benutzt wird. Zusätzliche Kontrolle bietet das RONTRAGUI-Trace-Fenster (siehe Abbildung 38), das durch Anklicken des Tracefenster-Buttons in der Buttonleiste aktiviert werden kann.

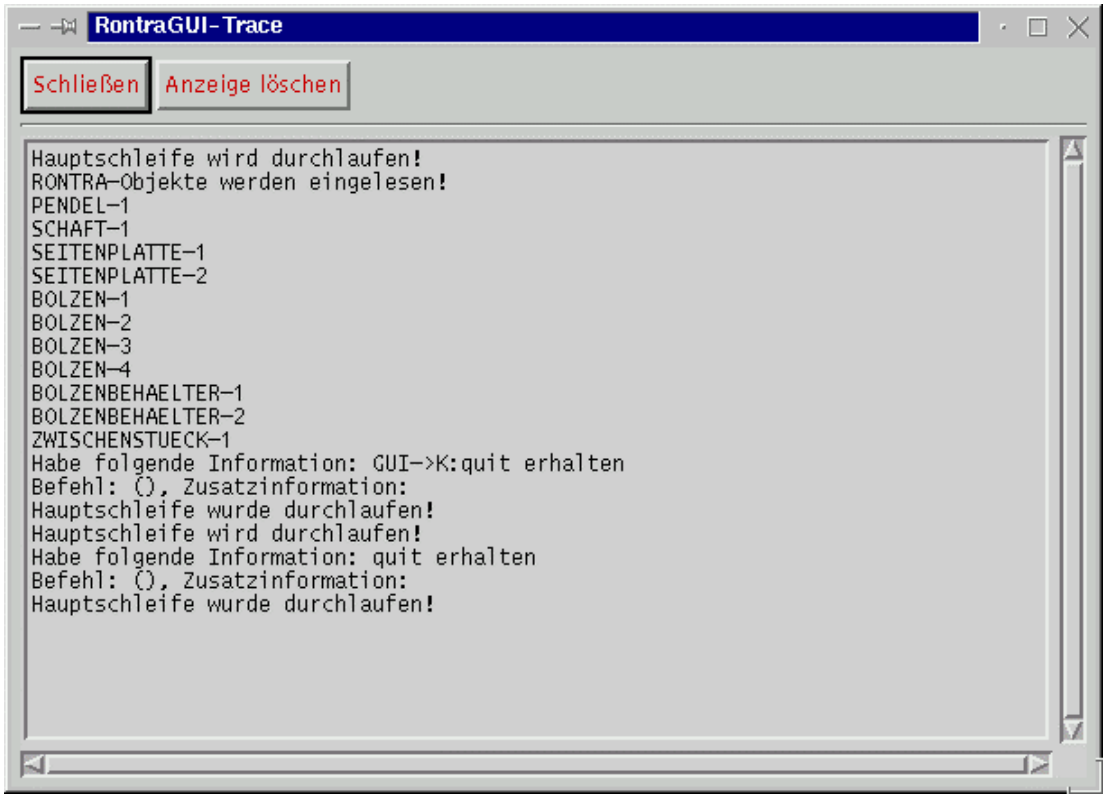

Abbildung 38: Trace-Fenster von RONTRAGUI.

Hier wird der Benutzer primär über die internen Berechnungen des Kernsystems informiert. Darüber hinaus werden auch interne Nachrichten der Module und die der Zu-

satzsysteme angezeigt. Insbesondere wenn Fehler auftauchen und erkannt werden, deren Ursachen nicht direkt nachvollziehbar sind (zum Beispiel die Angabe von falsche Koordinaten für das aktuelle Handlungsobjekt in der Tätigkeitsbeschreibung) bietet diese Darstellung einen chronologischen Überblick über die gesamten Berechnungen, so daß Zeitpunkt und die Berechnungsmethode und somit auch die Fehlerursache genau lokalisiert werden kann. Einen tieferen Einblick und auch Eingriff in die Arbeitsvorgänge von dem RONTRA-System ist mit dem RONTRAGUI-Lisp-Fenster (siehe Abbildung 39) realisiert. Die Aktivierung erfolgt durch das Anklicken des RONTRA-Trace-Buttons. In diesem Dialogfenster werden alle Daten und Ausgaben von RONTRA dargestellt. Zusätzlich gibt es eine Eingabezeile am unteren Rand, die ein direktes Interagieren mit dem LISP-System erlaubt. Diese Eingabezeile wird erst dann aktiviert, wenn der Verbindungsvorgang mit dem Kernsystem vollständig abgeschlossen ist.

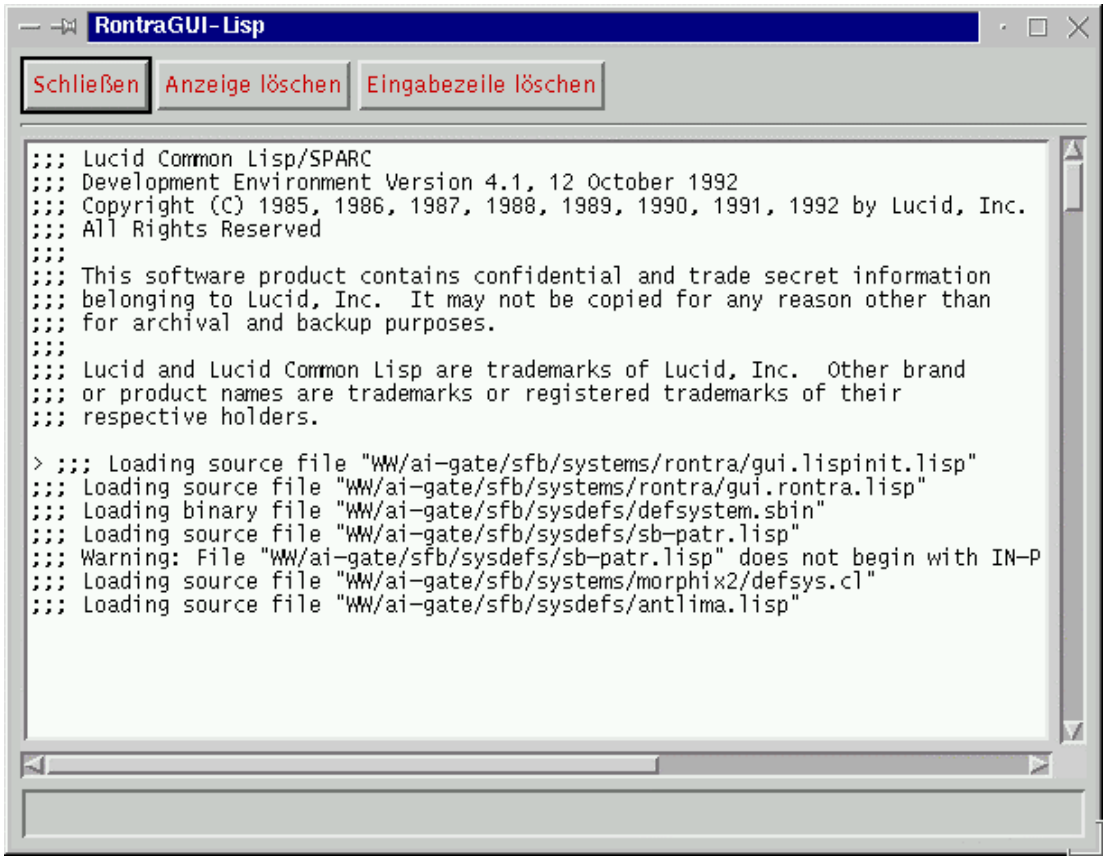

Abbildung 39: Lisp-Fenster von RONTRAGUI.

Das BOLA-Lisp-Fenster gibt Auskunft über die Arbeitsvorgänge bei der Berechnung räumlicher Ausdrücke. Außerdem kann hier durch Eingabe von LISP-Befehlen die Berechnung manipuliert werden. Dieses Fenster wird durch Anklicken des BOLA-Trace-Buttons aktiviert werden.

6.2.2 Konfiguration

Damit RONTRAGUI überhaupt eine Verbindung zu dem Kernsystem herstellen kann, müssen die für die Verbindung notwendigen Variablen mit Werten belegt werden. Ein dafür geschaffenes Dialogfenster kann über das Untermenü *RONTRAGUI-Einstellungen* im Menü *Einstellungen* erreicht werden. Das Format der hier benötigten Angaben wird im folgenden gezeigt:

<**RGUIV**>:<**Variable**>:<**Wert**>

Einträge, die nicht in diesem Format vorliegen, werden ignoriert. Die Variablen, die für eine erfolgreiche Verbindung mit dem Kernsystem gebraucht werden, zeigt die nachstehende Auflistung:

- **HOST** muß den vollständigen Rechnernamen[43] beinhalten, auf dem das Kernsystem arbeitet. Zum Beispiel: `lisp.cs.uni-sb.de`.
- **PORT**. Die Variable muß den Wert enthalten, der für die Kommunikationsadresse des Kernsystems auf dem **HOST**-Rechner steht. Meist sind dies Werte zwischen `200` und `6000`.
- **BOLAHOST** muß den vollständigen Rechnernamen beinhalten, auf dem das BOLA-System arbeitet ist. Zum Beispiel: `bola.cs.uni-sb.de`.
- **BOLAPORT**. Die Variable den Wert enthalten, der für die Kommunikationsadresse des BOLA-Systems auf dem **BOLAHOST**-Rechner steht. Wie bei der **Port**-Variablen können hier Werte zwischen `200` und `6000` angegeben werden.
- **SINPORT** bezeichnet den Port, der für die Eingabe von Befehlen an den Roboter über die TCP-/IP-Schnittstelle eingerichtet ist. Dafür werden Zahlen über `9000` verwendet. Ein externes Programm, das sich für die Umwandlung von gesprochener Sprache in geschriebene Sprache kümmert, kann so RONTRA um die Eigenschaft erweitern, daß Befehle in gesprochener Sprache an den Roboter gestellt werden können. Dies ist zur Zeit mit einem Programm realisiert, das auf Spracherkennungsroutinen und Wissensbasen von IBM-Voicetype zurückgreift.
- **SOUTPORT** repräsentiert den Port, an dem ein externes Programm die erzeugten Erklärungen von Tätigkeiten oder Lagebezeichnungen abgreifen kann. Somit können diese auch in gesprochene Sprache umgewandelt werden.
- **RONTRAINIFILE** muß mit der Angabe der RONTRA-Initialisierungsdatei belegt werden. Sie muß das Format (`load "<Verzeichnis/LISP-Datei>"`) haben.
- **BOLAINIFILE** muß mit der Angabe der BOLA-Initialisierungsdatei belegt werden. Sie muß das Format (`load "<Verzeichnis/LISP-Datei>"`) haben.

[43] Dieser besteht aus dem Rechnernamen und der vollständigen Bezeichnung der Internet-Domäne.

- **LOADANTLIMACOMMAND** beherbergt die LISP-Funktion, mit der das ANTLIMA-System gestartet wird. Beispiel: (`antlima::run-antlima "lisp.cs.uni-sb.de" :set-up "kamro-2d"`).
- **SETDOMAIN** muß mit der LISP-Anweisungsfolge belegt werden, worin die aktuelle Anwendungsdomäne in RONTRA gesetzt wird. Beispiel: (`setf RONTRADOMAIN "kamro"`).
- **RONTRAGUICONTROLLFILE**. In dieser Variablen muß die Ladeanweisung für den Kontrollagenten abgelegt werden. Format: (`load "<Verzeichnis/LISP-Datei>"`).
- **RONTRACONTROLGUIFUNCTION** sollte mit der LISP-Funktion belegt sein, die das Kernsystem startet. Beispiel: (`rontra`).

Neben diesen Einstellungen können die Kurzfragen beziehungsweise Befehle, die im Eingabebereich von RONTRAGUI erscheinen, definiert werden. Die dafür vorgesehenen Variablen werden im folgenden beschrieben:

- **Q_CHOICEITEM**. Die hier gemachte Angabe erscheint im Auswahlmenü für Kurzfragen. Für weitere Angaben muß jeweils eine neue Zeile mit dieser Variable angelegt werden. Beispiele: „`Was?`“ oder „`Warum?`“.
- **C_CHOICEITEM**. Die hier gemachte Angabe erscheint im Auswahlmenü für Kurzbefehle. Für weitere Angaben muß jeweils eine neue Zeile mit dieser Variable angelegt werden. Beispiele: „`Nochmal!`“ oder „`Nur das Wesentliche!`“.

In Abbildung 40 ist das Einstellungsdialogfenster von RONTRAGUI gezeigt, in dem die eben beschrieben Variablen belegt werden können. Die darin präsentierten Variablenbelegungen dienen nur der Anschauung.

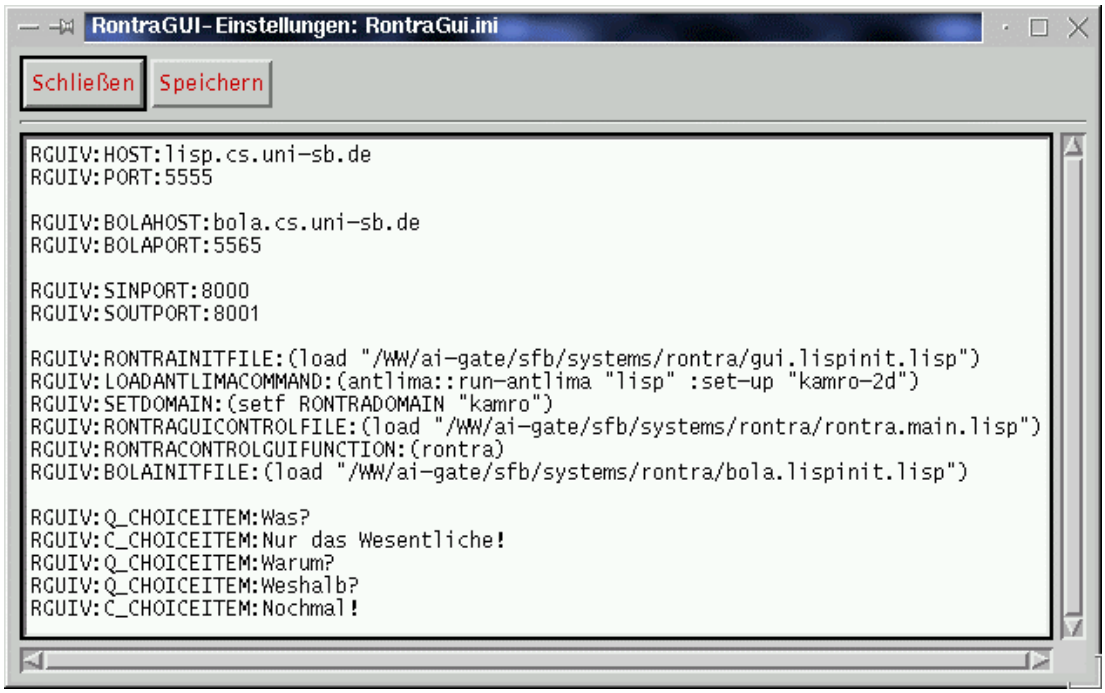

Abbildung 40: Der Einstellungsdialog von RONTRAGUI.

6.3 Die Anwendungsdomäne CHEW

CHEW[44] ist der definierte Aufbau eines Zwischenlagers für chemische Stoffe. Es kann in seiner Art für verschiedene Projekte und Abteilungen genutzt werden. Es wurde von der Abteilung **H**azardous **W**aste **M**anagement **D**ivision (HWMD) des **L**awrence **L**ivermore **N**ational **L**aboratory (LLNL)), einer Forschungsanstalt, die dem Ministerium für Energie der USA unterstellt ist, entwickelt. Das LLNL beschäftigt sich mit Fragen der globalen Sicherheit und Ökologie sowie mit Problemen im Bereich der Biowissenschaften. Bei Forschungsarbeiten auf diesen Gebieten fallen häufig Chemikalien an, die als Sondermüll gelten und geeigneten Endlagern zugeführt werden müssen. Um Kosten zu minimieren und um Chemikalien wieder zu verwenden, wurde CHEW entwickelt. Damit gelang es die Mengen an Sondermüll weitestgehend zu reduzieren.

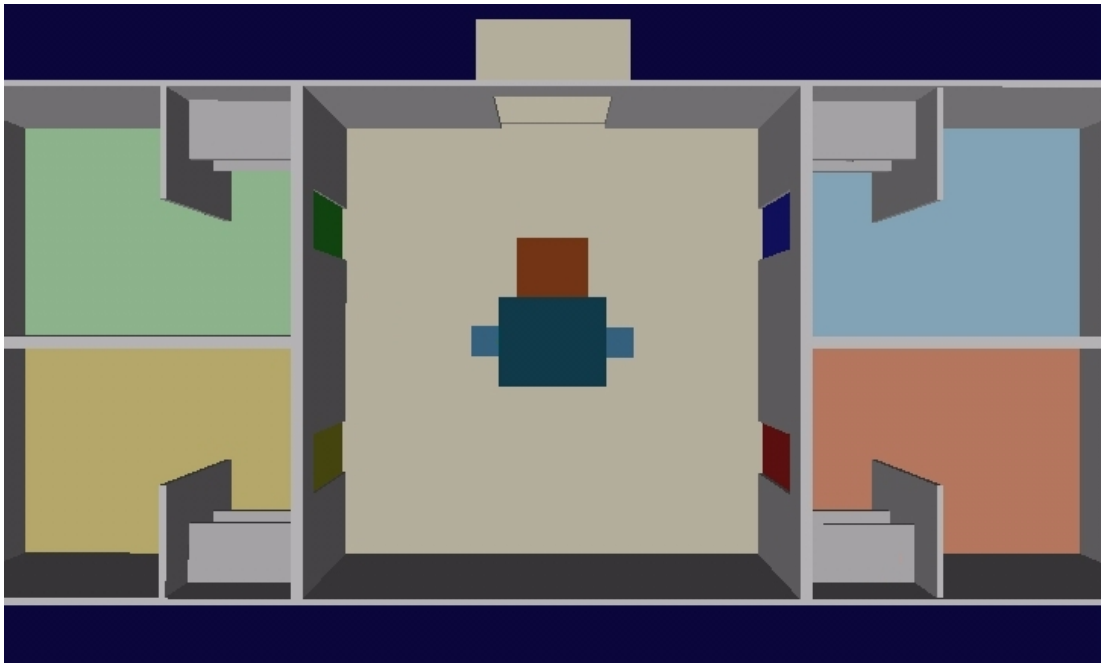

Abbildung 41: Grundriß des CHEW-Lagers.

Die eigentliche Ausstattung des Lagers und die vom Personal ausgeführten Tätigkeiten unterliegen strengen Sicherheitsauflagen. Im Laufe der Zeit ereigneten sich trotz dieser Auflagen Unfälle im Zusammenhang mit den eingelagerten Chemikalien, die auf menschliches Versagen oder auf unvorhergesehene Ereignisse zurückzuführen sind. Der Einsatz von Robotern erscheint nach [Feibel 97] sinnvoll, da somit dem Personal risikobehaftete Tätigkeiten, wie Transport und Einlagerung von gefährlichen Chemikalien, abgenommen werden. Er entwarf innerhalb dieses Szenarios ein Konzept eines Planungssystems zur Steuerung eines Roboters. Es erlaubt, für gegebene Aufgaben Handlungspläne zu generieren und sicher auszuführen.

Das CHEW-Lager besteht aus fünf Räumen und einer Rampe, über die Chemikalien an- und ausgeliefert werden (siehe Abbildung 41). Der zentrale Bereich des Lagers, der

[44] **Ch**eminal **e**xchange **w**arehouse

auch Halle genannt wird, beherbergt die Analysestation, die für die Bestimmung des Inhaltes von Chemikalienbehältern zuständig ist. Behälter in Form von Fässern und Dosen, deren Inhalt durch die Analysestation ermittelt worden ist, erhalten eine, dem Inhalt entsprechende, Farbe (beispielsweise Blau für Laugen oder Grün für Gift). Zudem befindet sich hier ein Stapel von Paletten, die unter anderem zum Zwischenlagern von Fässern verwendet werden. Für die Lagerung der verschiedenen Stoffe existieren vier Räume, die jeweils durch eine Tür von der Halle abgetrennt sind. Aus sicherheitstechnischen Gründen ist auch die Rampe durch eine Tür von der Halle getrennt. Der mobile Roboter soll die Ein- und Auslagerungsarbeiten übernehmen. Dabei gelten die folgenden Sicherheitsauflagen:

- Je Lagerraum dürfen jeweils nur Chemikalien einer bestimmten Stoffklasse gelagert werden. Die Stoffklassen teilen sich in Säuren, Laugen, toxische und explosive Stoffe auf.
- Dosen dürfen nur auf Regalen abgestellt werden.
- In einem Raum dürfen Fässer nur dann direkt auf den Boden gestellt werden, wenn dort kein weiteres steht. Somit ergibt sich, daß alle Fässer eines Raumes bis auf maximal eines auf Paletten stehen müssen. Auf einer Palette dürfen nur Fässer stehen, deren Inhalte jeweils die gleiche chemische Zusammensetzung haben.
- Der Roboter kann immer nur einen Gegenstand transportieren: eine Dose, ein Faß oder eine Palette.

Die Bewegungen des Roboters sind durch einen festen Graphen definiert, dessen Knoten die erlaubten Positionen im Lager repräsentieren [Feibel 97]. Alle Handlungen des Roboters werden durch das 3D-Visualisierungsprogramm GEOMVIEW dargestellt. Die aus Plänen vom Planungssystem RAP[45] generierten Kommandos an den Roboter werden parallel von einem Kommandointerpreter in GCL[46]-konforme Anweisungen übersetzt. Diese steuern den GEOMVIEW-Prozeß, der für die Darstellung der Roboterhandlungen zuständig ist. Auf umgekehrtem Wege erhält das RAP-System Meldungen über eine korrekte Ausführung des Befehls oder über eventuelle Fehlersituationen.

6.3.1 Schnittstelle zum CHEW-Roboter

Damit natürlichsprachliche Erklärungen erzeugt werden können, müssen Umwelt-, Plan- und Statusinformationen vorliegen (siehe Kapitel 4.1). Für den Austausch der Umweltinformationen und der Statusinformationen konnte die bereits vorhandene Schnittstelle[47] genutzt werden. Jedoch kann das Planungssystem keine Diagnosemeldungen erzeugen, die für natürlichsprachliche Fehlerbeschreibungen benötigt werden. Das Format der Anweisungspläne für den CHEW-Roboter wird generell von ROPLEX akzeptiert. Für die Erkennung der einzelnen Befehle in den Planschritten müssen die

[45] **R**easoning **a**bout **p**lans

[46] **G**eomview **c**ommand **l**anguage

[47] Die Schnittstelle bietet sowohl eine Ansteuerung über PROLOG als auch über C. Im Datenaustausch mit ROPLEX wird die C-Schnittstelle verwendet.

Plananalyseschemen angepaßt werden (siehe Unterabschnitt 4.4.2.1). Die Syntax der Planschritt-Befehle, die als Grundlage für die Erklärungen dienen, sind im folgenden präsentiert:

1. `Befehl(`**`move`**`, RobName, StartLok, EndLok)`
2. `Befehl(`**`passdoor`**`, RobName, StartLok, EndLok, TuerName)`
3. `Befehl(`**`gotoramp`**`, RobName, StartLok, EndLok)`
4. `Befehl(`**`gotomain`**`, RobName, StartLok, EndLok)`
5. `Befehl(`**`open_door`**`, TuerName)`
6. `Befehl(`**`close_door`**`, TuerName)`
7. `Befehl(`**`pickpallet`**`, RobName, Lokation, Pal)`
8. `Befehl(`**`pickup`**`, RobName, ContName, Lokation, RobName-ContTyp+Farbe)`
9. `Befehl(`**`putdown`**`, RobName, Lokation, NeueContLokation, ContName, ContTyp+Farbe)`
10. `Befehl(`**`analyse`**`, ContName, EingangsLokation, AusgangsLokation, ContTyp+Farbe)`

Dabei steht `RobName` für den Namen des Roboters (`robo`). Die Variablen `StartLok`, `EndLok`, `Lokation`, `NeueCont-`, `Eingangs-` und `AusgangsLokation` können mit Bezeichnungen für Knoten des Bewegungsgraph belegt werden. Der Inhalt von `Tuer-Name` ist die Bezeichnung des Knoten, der die jeweilige Tür repräsentiert. Die Variable `ContName` nimmt Bezeichnungen für einen Behälter auf. In der Variablen `ContTyp` wird dessen Typbezeichner aufgenommen. Eine weitere Spezifizierung der Behälter ist deren Farbe. Dabei steht `grün` für Gift, `blau` für Laugen, `rot` für Säuren, `gelb` für Explosivstoffe und `grau` für unbekannt. Für die Bezeichnung von Paletten steht die Variable `Pal` zur Verfügung. Die Kommandos kann man in Bewegungskommandos 1. – 4., Befehle zum Öffnen und Schließen der Türen 5. – 6., zum Aufnehmen und Abstellen von Objekten 7. – 9. und zur Analyse 10. gliedern. Zudem existieren noch einige Steuerbefehle, die hier jedoch nicht betrachtet werden.

Im Gegensatz zu KAMRO bewegt sich dieser Roboter nicht frei im Raum. Er ist an ein Graphen gebunden, der ihm vorgibt, wie er sich von Punkt A zu Punkt B zu bewegen hat. Diese Tatsache spiegelt sich auch in den Befehlen wider. Anstelle von Koordinaten, die angeben, wohin er sich begeben soll, muß der Name eines Knotens angegeben werden. Es existiert eine endliche Menge von Knoten, deren Elemente für die Angabe von Lokationen verwendet werden müssen. Abhängig von der Befehlsart stehen Teilmengen der Knotenmenge zur Verfügung. Für den **`move`**-Befehl können nur die Knoten verwendet werden, die in dem Raum sind, in dem sich der Roboter befindet. Beim **`passdoor`**-Befehl ist zu beachten, daß der Ausgangskonten, der Türknoten und der Endknoten angegeben sind. Sie beschreiben die Knoten, die der Roboter beim Passieren einer Tür abfahren muß. Die Befehle **`gotoramp`** und **`gotomain`** sind spezielle Befehle für die Bewegung auf die Rampe beziehungsweise in den Hauptraum. **`Pickpal-`**

let ist ein Spezialbefehl zum Aufnehmen einer Palette vom Palettenstapel in der Halle. Als Lokation muß der Knoten angegeben werden, der direkt davor liegt und für diesen Zweck vorgesehen ist. Der allgemeinere **pickup**-Befehl erfordert neben der Angabe der Objektbezeichner für den Roboter und den aufzunehmenden Container die Roboterlokation sowie die genaue Typisierung des entsprechenden Behälters. Das Kommando **putdown** wird um die Lokation ergänzt, an der das Gefäß abgestellt werden soll. Die Analyse des Inhalts eines Gefäßes geschieht mittels des **analyse**-Befehls. Er bewirkt, daß ein - normalerweise graues - Gefäß analysiert wird, wobei `Cont-Typ+Farbe` dem Ergebnisses entspricht. Die Knoten `Eingangs-` und `AusgangsLokation` der Analysestation, auf die zu untersuchende Behälter gelegt beziehungsweise ausgegeben werden, sind eindeutig definiert.

6.3.2 Ein Beispiellauf

In diesem Abschnitt wird die Zusammenarbeit von ROPLEX mit dem CHEW-Roboter anhand einer Serie von Bildschirmabzügen präsentiert. Sie zeigen die Handlungen des Roboters beim Einlagern einer neuen Dose und die dazugehörigen Erklärungen. Zudem wird hier eine Anwendung der Lokalisationsfrage gezeigt. Der Vorgang des Einlagerns beginnt mit dem Abholen der Dose auf der Rampe. Dazu muß der Roboter von seiner Ausgangsposition (siehe Abbildung 42) auf die Rampe fahren und die Dose aufheben.

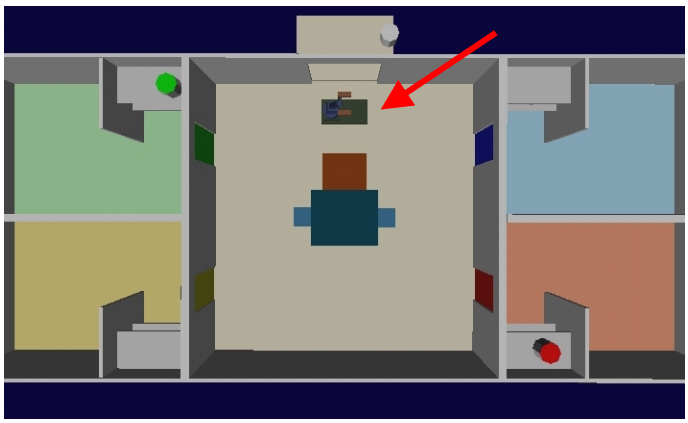

Abbildung 42: Ausgangsposition des CHEW-Roboters.

Die zugrundeliegenden Planschritte sind:

1. `Befehl(`**`gotoramp`**`, robo, door_ramp, ramp_robo_pos)`
2. `Befehl(`**`pickup`**`, robo, contsmall1, ramp_robo_pos, robocontsmall)`

Danach ergibt sich das in Abbildung 43 gezeigte Szenario. Zeitgleich neben der Abarbeitung der Anweisungen werden die dazugehörigen Erklärungen (1. „`Ich fahre zu der Rampe`“ und 2. „`Jetzt greife ich das Fass`“) erzeugt, siehe dazu Abbildung 44.

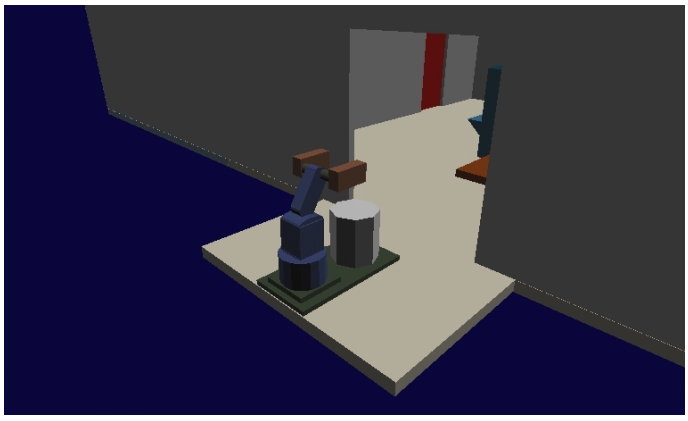

Abbildung 43: Chew hat das Faß auf der Rampe gegriffen.

Um herauszufinden, wo sich

der Roboter gerade aufhält, kann dem System die Frage: „`Wo befindest Du Dich gerade?`“ gestellt werden. Dazu wird das aktuelle Umweltwissen befragt, um daraus eine Antwort zu generieren (siehe dazu Abschnitt 4.6.2). Sie wird anschließend im Antwortfenster der Benutzerschnittstelle angezeigt (siehe Abbildung 44).

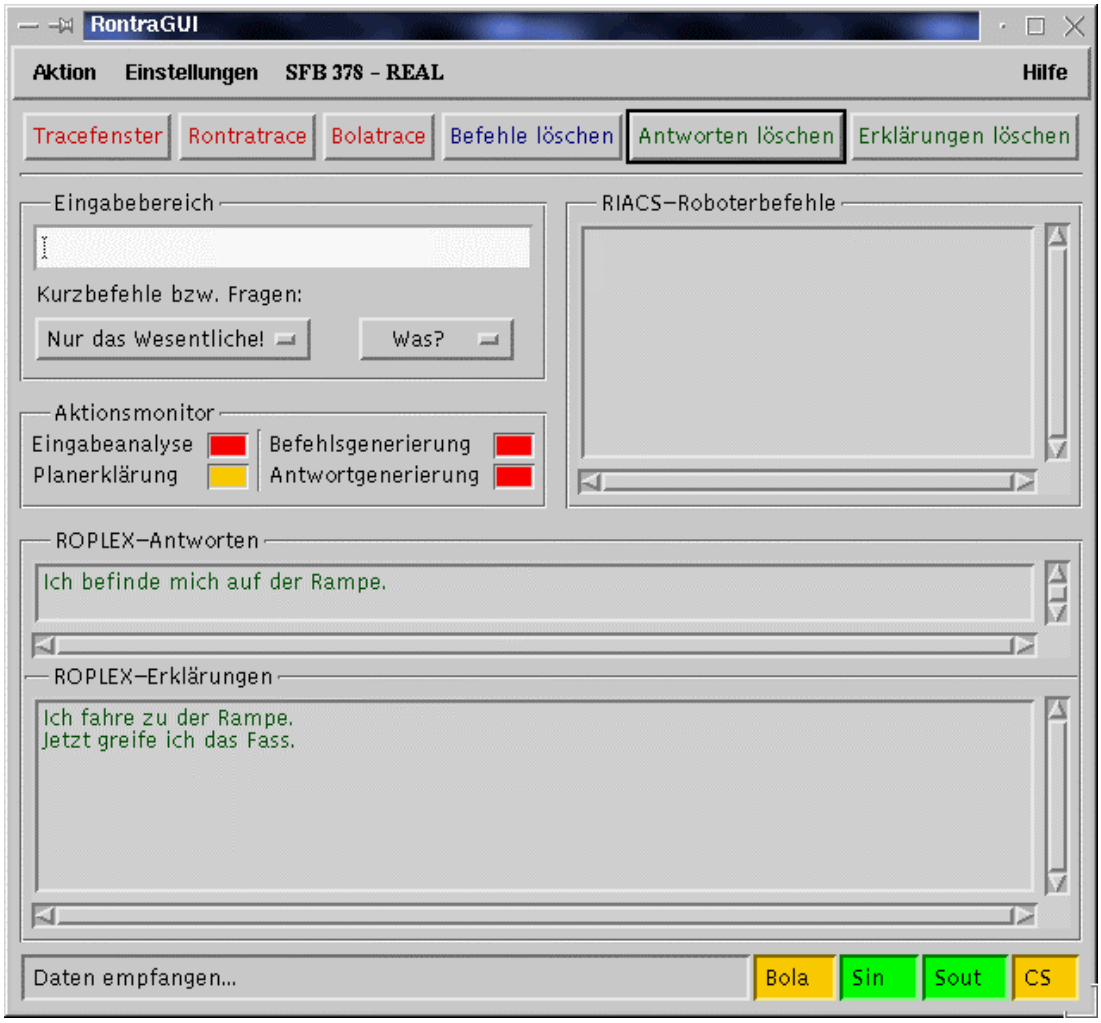

Abbildung 44: Anzeige von Antworten und Erklärungen nach dem Greifen des Fasses auf der Rampe.

Nun muß die Dose, damit sie ordnungsgemäß eingelagert werden kann, analysiert werden. Dazu wird sie an die Analysestation gebracht und von ihr untersucht. Die dazu notwendigen Kommandos (Planschritte) sind:

3. `Befehl(`**`passdoor`**`, robo, ramp_robo_pos, robo_dev_in, ramp_door)`
4. `Befehl(`**`putdown`**`, robo, robo_dev_in, dev_in, contsmall1, contsmall)`
5. `Befehl(`**`analyse`**`, contsmall1, dev_in, dev_out, contsmallgelb)`

Die resultierende Bewegungssequenz des Roboters ist in Abbildung 45 und Abbildung 46 gezeigt. Die dazugehörigen Handlungsbeschreibungen lauten:

3. „`Nun gehe ich durch die Rampentuer zu der Analysestation.`“
4. „`Ich lege das Fass in die Eingabestelle von der Analysestation.`“
5. „`Und lasse es analysieren.`“

Die Analysestation erkennt, daß es sich bei dem Inhalt der Dose um einen Explosivstoff handelt. Dies hat zur Folge, das die Dose mit der Farbe Gelb gekennzeichnet wird. Das Lager für solche Stoffe befindet sich hinter der gelben Tür. Zuerst muß die Dose jedoch von der anderen Seite der Analysestation abgeholt werden.

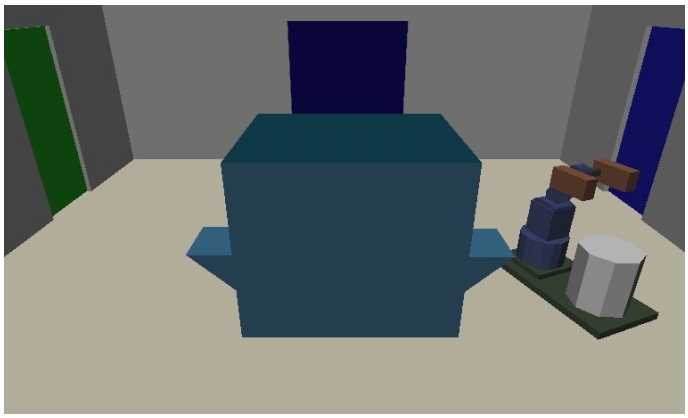

Abbildung 45: Der CHEW-Roboter vor der Analysestation.

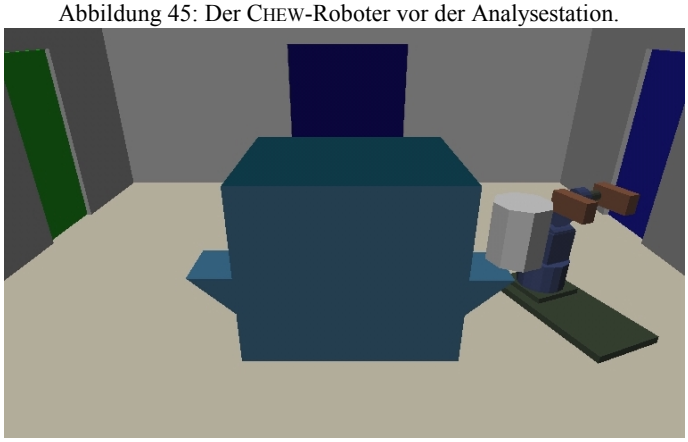

Abbildung 46: Einlegen des Fasses in die Analysestation.

Anschließend daran wird die Dose an einen geeigneten Stellplatz ins Regal des Explosivstofflagers gebracht. Der Roboter öffnet zunächst die Tür zu dem Lagerraum für Explosivstoffe, um dann anschließend das untersuchte Faß an eine geeignete Stelle im Regal abzulegen. Im folgenden ist die dazugehörige Plansequenz gezeigt:

6. `Befehl(`**`move`**`, robo, robo_dev_in, robo_dev_out)`
7. `Befehl(`**`pickup`**`, robo, contsmall1, ramp_robo_pos, robocontsmall)`
8. `Befehl(`**`open_door`**`, tuer_gelb)`
9. `Befehl(`**`passdoor`**`, robo, robo_dev_out, yellow, tuer_gelb)`
10. `Befehl(`**`putdown`**`, robo, yellow, cont_regal_gelb, contsmall1, contsmallgelb)`

Die korrespondierende Erklärungssequenz dazu ist:

6. „`Jetzt fahre ich zu der Ausgabestelle von der Analysestation.`“
7. „`Ich hebe das Fass auf.`“
8. „`Ich öffne just die gelbe Tuer.`“
9. „`Gerade fahre ich durch die gelbe Tuer zu dem Regal`“
10. „`Im Augenblick lege ich das gelbe Fass in das Regal ab.`“

Danach soll sich der Roboter zurück in die Halle begeben, um für die nächste Aufgabe bereit zu sein (siehe Abbildung 47). Dies geschieht durch das Kommando **`passdoor`**. Anschließend muß noch die Tür geschlossen werden, um eventuellen Gefahren vorzubeugen. Der Befehl dazu ist **`close_door`**.

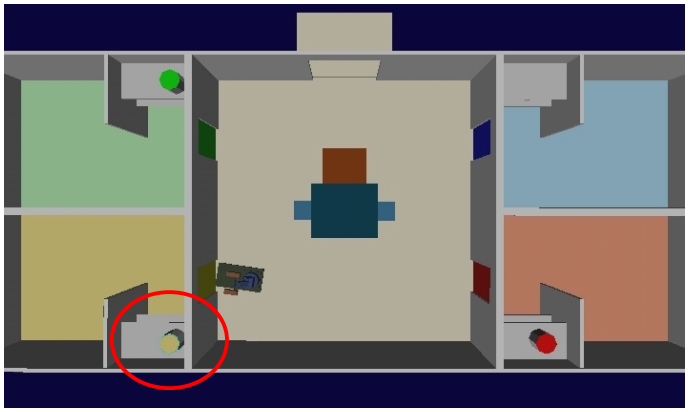

Abbildung 47: Das Faß ist in dem Regal abgestellt worden.

Die korrespondierenden Erklärungen der letzten Handlungen des Roboters sind im entsprechenden Bereich von RontraGUI zu sehen (siehe Abbildung 48).

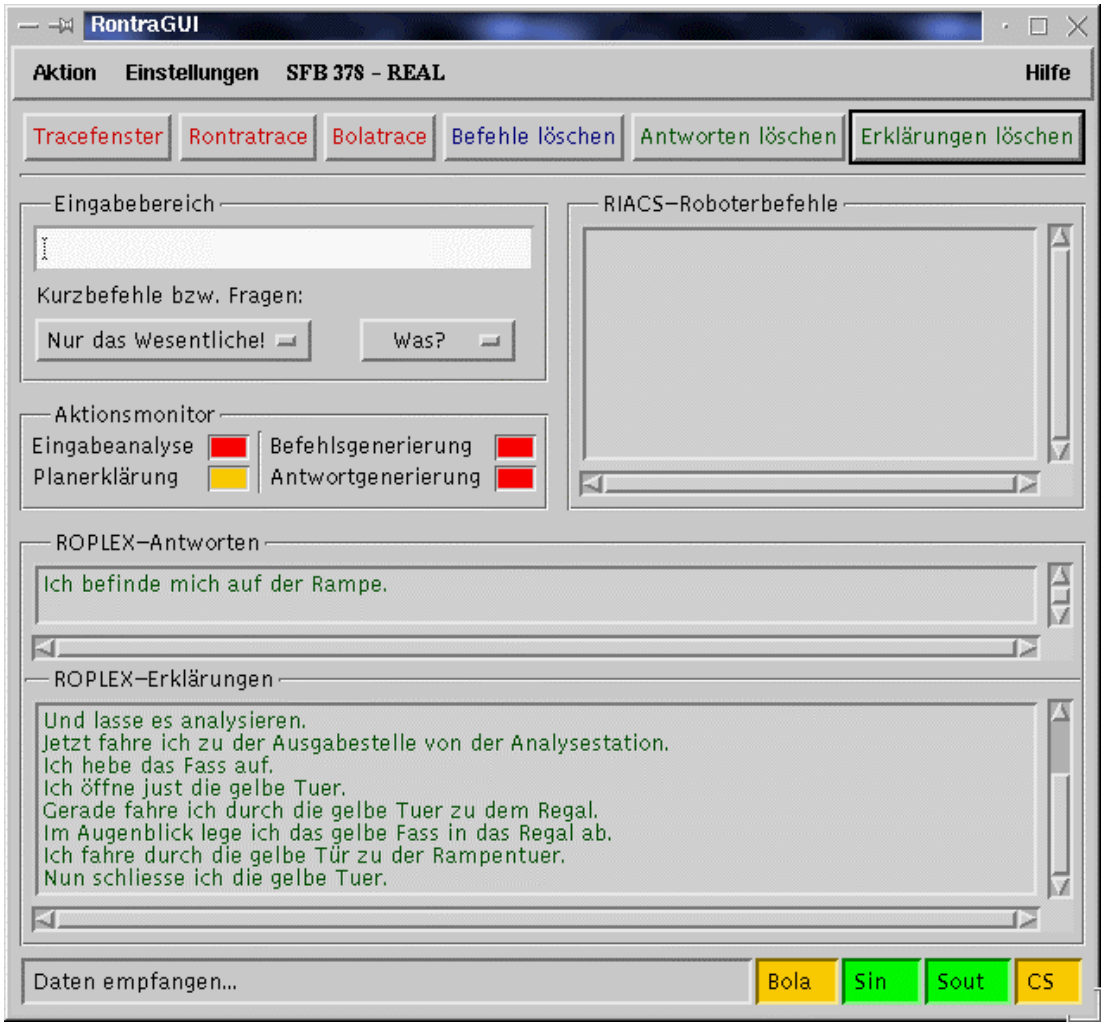

Abbildung 48: Anzeige der Erklärungen nach der Planabarbeitung.

Literaturverzeichnis

[Blocher et al. 92] A. **Blocher**, E. **Stopp**, T. **Weis**. ANTLIMA-1: Ein System zur Generierung von Bildvorstellungen ausgehend von Propositionen. Memo Nr. 50, Fachbereich 14, Universität des Saarbrücken, Saarbrücken, 1992.

[Buchanan & Shortliffe 84] B. G. **Buchanan** und E. H. **Shortliffe**. Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project. Reading, Addison-Wesley Publishing Company, Massachusetts, 1984.

[Crangle & Suppes 95] C. **Crangle** und P. **Suppes**. Language and Learning for Robots. Cambridge University Press, 1995.

[Dean & Boddy 88] T. **Dean** und M. **Boddy**. An Analysis of Time-Dependent Planning, Seiten 49-54. In Proceedings of AAAI-88. St. Paul, MN, 1988.

[Dreyer & Schmitt 96] H. **Dreyer** und P. **Schmitt**. Lehr- und Übungsbuch der deutschen Grammatik, Seiten 93 - 99. 1. Auflage 1996. Ismaning, Verlag für Deutsch.

[Drosdowski & Eisenberg 95] G. **Drosdowski** und P. **Eisenberg**. Duden »Grammatik der deutschen Gegenwartssprache«. 5. Auflage 1995. Mannheim, Leipzig, Wien, Zürich. Dudenverlag.

[Eichler & Bünting 96] W. **Eichler** und K.-D. **Bünting**. Deutsche Grammatik – Form, Leistung und Gebrauch der Gegenwartssprache. 6. Auflage 1996. Weinheim, Beltz Athenäum Verlag.

[Feiner & Keown 90] S. **Feiner** und K. R. Mc **Keown**. Generating Coordinated Multimedia Explanations, Proceedings of the Sixth Conference on Artificial Intelligence Applications CAIA-90,Seiten 290 – 296, 1990.

[Finkler & Neumann 88] W. **Finkler** und G. **Neumann**. MORPHIX. A Fast Realization of a Classification-Based Approach to Morphologie, Seiten 11-19. In: H. Trost (ed.): 4. Österreichische Artificial-Intelligence-Tagung. Wiener Workshop - Wissensbasierte Sprachverarbeitung. Proceedings, Berlin, Springer, 1988.

[Feibel 97] H. **Feibel**. Ein Konfigurationssystem für Planungsaufgaben in sicherheitskritischen Anwendungen. Dissertation, Technische Fakultät, Universität des Saarlandes, Saarbrücken, 1997.

[Gebhard & Jung 96] P. **Gebhard** und C. **Jung**. KANTRA-1: Analyse natürlichsprachlicher Befehle an einen autonomen, mobilen Roboter unter besonderer Berücksichtigung räumlicher Informationen. Sonderforschungsbereich 314 – Universität Saarbrücken, Bericht zum Fortgeschrittenenpraktikum, 1996.

[Geist et al. 94] A. **Geist**, A. **Beguelin**, J. **Dongarra**, W. **Jiang**, R. **Manchek** und V. **Sunderam**. PVM. Parallel Virtual Machine. A User's Guide and Tutorial for Networked Parallel Computing. MIT Press, Cambridge, MA. London, England, 1994.

[Gosling et al. 97] J. **Gosling**, F. **Yellin**, **Java Team**. Java™ API. 1. Auflage 1997. Bonn, Addison-Wesley-Longman.

[Grice 75] H. P. **Grice**. Logic and Conversation. In P. Cole and J. L. Morgan (eds.), Syntax and Semantics: 3. Auflage: Speech Acts, Seiten 41-58. San Diego, CA: Academic Press, 1975.

[Herzog et al. 96] G. **Herzog**, A. **Blocher**, K.-P. **Gapp**, E. **Stopp** und W. **Wahlster**. VITRA: Verbalisierung visuelle Information. Informatik – Forschung und Entwicklung, 11(1) Seiten 12 - 19, 1996.

[Herzog & Wazinski 94] G. **Herzog** und P. **Wazinski**. VIsual TRAnstlator: Linking Perceptions and Natural Language Descriptions. Artificial Intelligence Review Journal, Band 8, Nummer 2, Seiten 175-187, 1994. Special Volume on the Integration of Natural Language and Vision Processing, edited by P. Mc Kevitt.

[HWMD 98] HWMD: http://www.llnl.gov/es_and_h/hwm_home/CHEW.html, Hazardous Waste Mangement Division, 1998.

[Jung 97] C. **Jung**. Analyse zur Ansteuerung natürlichsprachlicher Äußerungen zur Ansteuerung autonomer Roboter. Diplomarbeit, Fachbereich 14, Universität des Saarlandes, Saarbrücken, 1997.

[Karttunen 86] L. **Karttunen**. D-PATR. A Development Enviroment for Unification-Based Grammers S. 74-80. Coling 86.

[Konolige et al. 96] K. **Konolige**, K. **Myers**, E. **Ruspini**. The Saphira Architecture: A Design for Automony. Journal of Experimental and Theoretical Artificial Intelligence (JETAI), Special Issue on Architectures for Physical Agents, 1996.

[Laengle et al. 95] T. **Laengle**, T. C. **Lueth**, E. **Stopp**, G. **Herzog**, G. **Kamstrup**. KANTRA – A Natural Language Interface for Intelligent Robots. In U. Rembold, R. Dillmann, L. O. Hertzberger & T. Kanade (Hrsg.), Proc. Of the 4th International Conference on Intelligent Autonomous Systems, Seiten 357-364, Karlsruhe 1995.

[Laengle et al. 96] T. **Laengle**, T. C. **Lueth**, E. **Stopp**, G. **Herzog**. Natural Access to Intelligent Assembly Robots: Explaining Automatic Error Recovery. In A. Ramsay (Hrsg.), Artificial Intelligence: Methodology, Systems, Applications, Seiten 357-267. Amsterdam: IOS Press, 1996.

[Lewis & Papadimitriou 81] H. R. **Lewis** und C. H. **Papadimitriou**. Elements of the theory of computation, Seiten 29-38. Prentice-Hall International, Inc., a division of Simon & Schuster, Eaglewood Cliffs, NJ 07632, 1981.

[Lüth et al. 94] T. C. **Lüth**, T. **Laengle**, G. **Herzog**, E. **Stopp**, U. **Rembold**. KANTRA: Human-Maschine Interaction for Intelligent Robots Using Natural Language. In 3rd IEEE Int. Workshop on Robot and Human Communication, ROMAN ´94, Seiten 106-111. Nagoya, Japan, 1994.

[Maaß 94] W. **Maaß**. From Visual Perception to Multimodal Communication: Incremental Route Descriptions. Artifical Intelligence Review Journal, 8(5/6).

(Special Volume on Integration of Natural Language and Vision Processing), 1994.

[Maaß 96] W. **Maaß**. Von visuellen Daten zu inkrementellen Wegbeschreibungen in dreidimensionalen Umgebungen: Das Modell eines kognitiven Agenten, Seiten 159-174. Dissertation. Technische Fakultät, Universität des Saarlandes, Saarbrücken, 1996.

[Maaß et al. 95] W. **Maaß**, J. **Baus** und J. **Paul**. Visual Grounding of Route Descriptions in Dynamic Enviroments. In AAAI Fall Symposium an Computational Models for Integrating Language and Vision. MIT, Cambridge, MA, 1995.

[Nilson 84] N. J. **Nilson**. Shakey the Robot. Technical Note 323. Artificial Intelligence Center, SRI International, Menlo Park, CA, 1984.

[Parnas 72] D. **Parnas**. On the criteria to be used in decomposing systems into modules. Comm. ACM, 15(2), 1053-8 [215], 1972.

[Schirra & Stopp 93] J. R. J. **Schirra**, E. **Stopp**. ANTLIMA – A Listener Model with Mental Images. Proceedings of the 13th IJCAI, Seiten 175-180, Chamréry, Frankreich, 1993.

[Sommerville 95] I. **Sommerville**. Software Engineering. 5. Auflage 1995. Addison-Wesley Publishing Company, Wokinghamm – England.

[Srinivas 77] S. **Srinivas**. Error recovery in robot systems. PhD Theses, California Institute of Technology, 1977.

[Steele 90] G. **Steele**. COMMON LISP, 2. Auflage,1990.

[Stopp 98] E. **Stopp**. Ein Modell für natürlichsprachlichen Zugang zu autonomen Robotern. Dissertation, Saarbrücken: Universität des Saarlandes, 1998.

[Stopp et al. 94] E. **Stopp**, K.-P. **Gapp**, G. **Herzog**, T. **Laengle**, T. C. **Lueth**. Utilizing Spatial Relations for Natural Language Access to an Autonomous Mobile Robot. In B. Nebel und L. Drescher-Fischer (Hrsg.), KI-94: Advances in Artificial Intelligence, Seiten 39-50. Berlin, Heidelberg: Springer 1994.

[Stopp & Laengle 95] E. **Stopp**, T. **Laengle**. Natürlichsprachliche Instruktionen an einen autonomen Serviceroboter. In R. Dillmann, U. Rembold & T. Lüth (Hrsg.), Autonome Mobile Systeme, Seiten 299-308. Berlin, Heidelberg: Springer, 1995.

[Trevelyan & Nelson 87] J. P. **Trevelyan** und M. **Nelson**. Adaptive robot control incorporating automatic error recovery. Proceedings of International Conference an Advanced Robotics, 1987.

[Torrance 94] M. C. **Torrance**. Natural Communication with Robots. Master Thesis, MIT, Department of Electrical Engineering and Computer Science, Cambridge, MA, 1994.

[Vere & Bickmore 90] S. **Vere** und T. **Bickmore**. A Basic Agent. Computational Intelligence 6(1), Seiten 41-60, 1990.

[Wahlster et al. 98] W. **Wahlster**, A. **Blocher**, J. **Baus**, E. **Stopp**, H. **Speiser**. Ressourcenadaptive Objektlokalisation: Sprachliche Raumbeschreibung unter Zeitdruck. Kognitionswissenschaft 7/3, Seiten 120-126,1998.

Abbildungsverzeichnis

Abbildung 1: Arbeitsweise von COMET. 14
Abbildung 2: Im Projekt REAL realisierte Systeme. 17
Abbildung 3: Modell der Dialogschnittstelle RONTRA. 22
Abbildung 4: Mehrstufige Referenzsemantik bei natürlichsprachlichem Zugang zu Robotern. 23
Abbildung 5: Multiagentenmodell von RONTRA. 28
Abbildung 6:Beispiel für die Arbeitsweise von BOLA. 30
Abbildung 7: Arbeitsweise von RIACS. 31
Abbildung 8: Der Roboter KAMRO. 32
Abbildung 9: Arbeitsweise von FATE und RT-RCS. 33
Abbildung 10: CAB-Teile und fertiger CAB. 33
Abbildung 11: Aufbau des Objektbeschreibungsformats von KAMRO. 34
Abbildung 12: Beispielszenario mit Objekten des CAB. 35
Abbildung 13: Syntax der verschiedenen Planformen für KAMRO. 37
Abbildung 14: Beispieloperationen für den Roboter KAMRO. 38
Abbildung 15: Fehlerbeschreibungen von KAMRO. 41
Abbildung 16: Arbeitsweise von KANTRA-1. 43
Abbildung 17: Einfacher Ansatz einer Planschritterklärungskomponente. 50
Abbildung 18: Beispiel einer Befehlsmustertabelle. 51
Abbildung 19: Beispiel einer Objekttranslationstabelle 51
Abbildung 20: Arbeitsweise von ROPLEX. 58
Abbildung 21: Format der Objektbeschreibung. 60
Abbildung 22: Akzeptierte Sprache von Plänen. 61
Abbildung 23: Akzeptierte Sprache der Planschritte. 62
Abbildung 24: Inhalte der Steuerdatei. 63
Abbildung 25: Beispielszenario mit einfachen Objekten 66
Abbildung 26: Aufbau der Objektinformationsstruktur. 71
Abbildung 27: Beispiele für Fehlersatzmasken und Situationsbeschreibungen. 72
Abbildung 28: Aufbau der Diagnoseinformationsstruktur. 73
Abbildung 29: Aufbau der Verbrahmen-Datenstruktur von ROPLEX. 76
Abbildung 30: Aufbau des Planschrittanalyseschemas. 83
Abbildung 31: Beipielkonfiguration vom Erklärungsprofil. 84
Abbildung 32: Stationen des Erklärungsalgorithmus. 87
Abbildung 33: Arbeitsweise der Erklärungssynthese. 91
Abbildung 34: Abschnitt des Erklärungsprofils. 92
Abbildung 35: Verarbeitung von natürlichsprachlichen Eingaben vom Steueragent 95

Abbildung 36: Abhängigkeiten zwischen den einzelnen Modulen und Wissensbasen von RONTRA.....98
Abbildung 37: Die graphische Benutzerschnittstelle RONTRAGUI.....111
Abbildung 38: Trace-Fenster von RONTRAGUI.....112
Abbildung 39: Lisp-Fenster von RONTRAGUI.....113
Abbildung 40: Der Einstellungsdialog von RONTRAGUI.....115
Abbildung 41: Grundriß des CHEW-Lagers.....116
Abbildung 42: Ausgangsposition des CHEW-Roboters.....119
Abbildung 43: Chew hat das Faß auf der Rampe gegriffen.....119
Abbildung 44: Anzeige von Antworten und Erklärungen nach dem Greifen des Fasses auf der Rampe.....120
Abbildung 45: Der CHEW-Roboter vor der Analysestation.....121
Abbildung 46: Einlegen des Fasses in die Analysestation.....121
Abbildung 47: Das Faß ist in dem Regal abgestellt worden.....122
Abbildung 48: Anzeige der Erklärungen nach der Planabarbeitung.....122

Index

A

Aktionsobjekt **46**, 62
ANTLIMA 42
Anweisungsidentifizierer 60
Anweisungsplan 46
Anweisungszeiger 63
Anwendungdomäne
  CHEW 116
  KAMRO 32
Anwendungsdomäne 32

B

Basiswissen 78, **79**
  Eingabeidentifizierer 80
  Eingabewissen 80
  Fehler-Verbrahmen 81
  Fehlerwissen 81
  Identifizierungsmaske 81
  Rahmenwissen 80
  Roboterzustand 81
  Satzaufbau 81
  Satzaussage 80
  Temporalinformation 80
  Umweltwissen 80
Befehlsmuster 49
Befehlsmustertabelle 49
Befehlsname 62
Befehlsplan 46
Berührungsobjekt 62
BOLA 17, **28**

C

CHEW 116
Cranfield-Assembly-Benchmark (CAB) 33
  Objekte 40

D

Diagnoseinformation 63
Diagnoseinformationsstruktur 73

E

Eingabeidentifizierer 80
Eingabeinformation 31
Eingabewissen 80
Elementaroperation 36
Erklärungsprofil 84

F

Fehlersatzmaske 68
Fehlersituation 46
  Fehlerbeschreibung 47
  Fehlerlösung 47
  Fehlerursache 47
Fehlertranslationstabelle 71
Fehler-Verbrahmen 81
Fehlerwissen 81
Fließtext 48
Füllbefehl 82

H

Handlungsobjekt 62
HWMD 116

I

Identifizierungsmaske 81
IPR 18

K

KAMRO 32
KAMRO 32
KANTRA-1 42
Konfigurationswissen 79, **81**
  Erklärungsprofil 84
  Füllbefehl 82
  Planschrittanalyseschema 82

L

LLNL 116
Lokalisationsfrage **25**, 96
Lokalisationssatzmaske 69
Lokationsausdruck 28
Lokationsbeschreibung 48

O

Objektbeschreibung 60
Objektinformationsstruktur 70
Objekttranslationstabelle 50

P

Plan 46
Planschritt 46
Planschrittanalyse

musterorientiert 49
sequenzenorientiert 58, **88**
Planschrittanalyseschema 82
Planschrittdaten 61
Prädikatsverband 64
Projekt REAL 16
Projekt VITRA 18

R

Rahmenwissen 80
räumliche Beschreibung **29**, 48
räumliche Relation 29
räumlicher Ausdruck 30
Referenzobjekt 46
Roboteranweisung 46
Unteranweisung 63
Roboterbefehle 46
Roboterszenario 34
Roboterzustand 81
RONTRA 21
Kernsystem 28
ROPLEX 10
RT-RCS 33

S

Sachaspekt 64
Satzaufbau 81
Satzaussage 80
Satzform 64
Satzmaske 49
Sektionsinformation 36
Situationsaspekt 64
Situationsbeschreibung 68
Statusinformation 63
Steuerdatei 63
Steuerungsprotokoll 70
Szenariodaten 59

T

Tätigkeitsbeschreibung 47
Temporalinformation 80

U

Umweltwissen 80

V

Verbrahmen 75
Verlaufsbeschreibung 47