Aktionsplanung und -steuerung unter Unsicherheit bei der Navigation eines Geometrischen Agenten mit Hilfe von Wegbeschreibungen

Diplomarbeit am Arbeitsbereich
Wissens- und Sprachverarbeitung
des Fachbereichs Informatik
der Universität Hamburg

von

Nils Bittkowski
Emanuel-Geibel-Weg 3
23858 Reinfeld / Holstein

Betreuerin: Dr. Carola Eschenbach
Arbeitsbereich Wissens- und Sprachverarbeitung

Zweitbetreuer: Prof. Dr. Jianwei Zhang
Arbeitsbereich Technische Aspekte Multimodaler Systeme

Mai 2005

Zusammenfassung

Das Projekt Geometrischer Agent widmet sich der Entwicklung einer Methode, mit Hilfe der sich die Interaktion zwischen räumlicher Information aus sprachlichen Wegbeschreibungen und räumlicher Information, die durch Perzeption der Umgebung gewonnen wird, untersuchen läßt. Für die formale Repräsentation des Wissens aus natürlichsprachlichen Routeninstruktionen wird die sogenannte *Conceptual Route Instruction Language* (CRIL) entwickelt. Im Rahmen des Projekts wird eine Anwendung geschaffen, in der ein künstlicher Agent in einer virtuellen planaren Umgebung durch Navigation unter Zuhilfenahme der CRIL-Repräsentation einer Wegbeschreibung sein vorgegebenes Ziel finden soll. Diese Simulation soll eine Evaluation verschiedener Theorien über die Interpretation natürlichsprachlicher Ausdrücke im Hinblick auf das Vorhandensein von perzipierter Information ermöglichen. Die Simulation umgeht dabei Probleme, die bei sog. *low-level-navigation tasks* auftreten, indem sie primitive Aktionen für die Perzeption und Bewegung des Agenten bereitstellt.

Welche Aktionen er während der Navigation in der virtuellen Umgebung ausführt, ermittelt der Geometrische Agent in einem Prozess der lokalen Planung, der sich sowohl am Aktionsplan, wie auch am internen Zustand des Agenten orientiert. Der Aktionsplan enthält eine Repräsentation der imperativen Anteile der Routeninstruktion. Er setzt sich aus Anweisungen zusammen, die die in der sprachlichen Routeninstruktion explizit genannten Handlungen repräsentieren. Der interne Zustand beschreibt das Wissen und die Annhamen des Geometrischen Agenten über die aktuelle Situation. Die Anweisungen aus dem Aktionsplan sind hinsichtlich der Ausführung in der simulierten Umgebung unterspezifiziert. Somit ergibt sich für den Agenten die Aufgabe, eine Abbildung dieser abstrakten Anweisungen auf primitive Aktionen, die er in der simulierten Umgebung ausführen kann, vorzunehmen.

In der Diplomarbeit wird ein Aktionsmodul entwickelt, das ein Verfahren der lokalen Planung bereitstellt. Diese Verfahren nimmt die Abarbeitung des Aktionsplans vor, d.h. es wählt mit Hilfe der formalen Repräsentation der Routeninstruktion und des internen Zustands des Agenten primitive Aktionen des Agenten aus und stößt deren Ausführung an.

Inhaltsverzeichnis

1 Einleitung **11**
- 1.1 Motivation . . . 11
- 1.2 Navigation . . . 12
- 1.3 Bezüge zur Robotik . . . 13
- 1.4 Vorgaben . . . 16
- 1.5 Was *nicht* behandelt wird . . . 17
- 1.6 Aufbau der Arbeit . . . 17

2 Der Geometrische Agent **19**
- 2.1 Agentensysteme . . . 19
- 2.2 Das Rationalitätskriterium . . . 21
- 2.3 Die Umgebung des Geometrischen Agenten . . . 22
- 2.4 Umgebungstypen . . . 25
- 2.5 Agententypen . . . 26
- 2.6 Ein Performanzmaß für den Geometrischen Agenten . . . 27

3 Verarbeitung der Routeninstruktion **31**
- 3.1 Routeninstruktionen . . . 31
- 3.2 Die Conceptual Route Description Language CRIL . . . 33
  - 3.2.1 Referentielle Netze . . . 34
  - 3.2.2 CRIL-Netze . . . 35
- 3.3 Die Instruktionsphase . . . 36
  - 3.3.1 Syntaktische und semantische Verarbeitung . . . 37
  - 3.3.2 Verarbeitung der Instruktion . . . 37
  - 3.3.3 Räumliche Anteile des Instruktionsmodells: der Instruktionsgraph . . . 38
  - 3.3.4 Imperative Anteile des Instruktionsmodells: der Aktionsplan . . . 40
- 3.4 Verwendung des Instruktionsmodells in der Navigationsphase . . . 41
- 3.5 Prozesse in der Navigationsphase . . . 42
  - 3.5.1 Verarbeitung der Perzeption . . . 42
  - 3.5.2 Auflösung von Koreferenzen . . . 42
  - 3.5.3 Selbstlokalisation . . . 44
- 3.6 Aufgaben des Aktionsmoduls . . . 45

4 Aktionsplanung und Ausführung unter Unsicherheit 47
4.1 Grundlagen der Planung . . . 47
4.2 Klassisches Planen . . . 48
4.2.1 Das klassische Planungsproblem . . . 48
4.2.2 Die Domäne klassischer Planungsverfahren . . . 49
4.2.3 Klassische Planungsverfahren . . . 50
4.3 Hierarchisches Planen . . . 50
4.4 Planen in unsicheren Umgebungen . . . 52
4.4.1 Ausführungsüberwachung und Neuplanung . . . 52
4.4.2 Kontinuierliches Planen . . . 54
4.4.3 Das RAP-System . . . 54
4.4.4 Die Plan-as-communication-Sichtweise . . . 55
4.4.5 Die Plan-as-behaviour-Sichtweise . . . 58
4.5 Verfahren für die instruktiongestützte Navigation eines Roboters . . . 60
4.5.1 Das Robot Navigation-Projekt . . . 60
4.5.2 Instruction-Based Learning . . . 64
4.6 Planen im Geometrischen Agenten . . . 69
4.6.1 Generierung und Nutzung des Aktionsplans . . . 70
4.6.2 Planungskonzepte im Geometrischen Agenten . . . 71

5 Das Aktionsmodul 75
5.1 Repräsentation des internen Zustands des Agenten . . . 75
5.1.1 Das Umgebungsmodell des Agenten . . . 75
5.1.2 Der Aktionsplan und die aktuell relevanten Instruktionsanweisungen . . . 77
5.1.3 Anweisungstypen . . . 77
5.1.4 Die Vorbedingung von Instruktionsanweisungen . . . 78
5.2 Die primitiven Aktionen des Agenten . . . 79
5.2.1 Perzeption . . . 80
5.2.2 Bewegung . . . 81
5.3 Prozesse während der Navigation . . . 83
5.3.1 Prozesse für die Zustandsbestimmung . . . 83
5.3.2 Prozesse für die Auswahl von Bewegungsaktionen . . . 84
5.3.3 Prozesse für die Problembehandlung . . . 84
5.4 Pragmatische Annahmen für die Hilfsprozesse . . . 85
5.4.1 Pragmatische Annahmen für die Selbstlokalisation . . . 85
5.4.2 Pragmatische Annahmen für die Auswahl eines Pfades . . . 86
5.5 Die Ablaufsteuerung . . . 90
5.6 Abarbeitung verschiedener Anweisungstypen . . . 91
5.6.1 Anweisungen vom Typ GO und CH_ORIENT . . . 91
5.6.2 Anweisungen vom Typ VIEW . . . 93
5.6.3 Anweisungen vom Typ BE_AT . . . 94
5.7 Folgeaktionen . . . 96
5.7.1 Typen von Folgeaktionen . . . 96
5.7.2 Verwendung der Folgeaktionen . . . 99
5.8 Algorithmen für die lokale Planung . . . 101

5.8.1 Die Ablaufsteuerung 101
5.8.2 Aktualisierung des internen Zustands 103
5.8.3 Auswahl und Ausführung von primitiven Bewegungsaktionen 103
5.8.4 Ausführung einer Folgeaktion 105
5.9 Ein Beispiel 107
5.10 Weitere Verfahren für die Ablaufsteuerung 110
5.10.1 Test auf Lücken im Aktionsplan 110
5.10.2 Problemfall: der Agent hat sich verlaufen 112

6 Schlussbetrachtung und Ausblick **115**
6.1 Schlussbetrachtung 115
6.2 Ausblick 116

Abbildungsverzeichnis

1.1 Hierarchische Steuerungsstruktur eines autonomen Robotersystems 14

2.1 Interaktion des Agenten mit seiner Umwelt 20
2.2 Der Informatikcampus Stellingen 23
2.3 Die Komponenten der simulierten Umgebung 24

3.1 Die Route, die durch die Instruktionen (1) und (2) beschrieben wird. 33
3.2 Ein referentielles Netz (Beispiel) 34
3.3 Die Instruktionsphase des Geometrischen Agenten 37
3.4 Repräsentation im CRIL-Netz nach Inferenz 40
3.5 Koreferenz zwischen Instruktions- und Perzeptionsmodell 44

4.1 Hierarchisches Planen 51
4.2 Beispiel für einen Algorithmus für die Abarbeitung einer primitiven Prozedur. 68
4.3 Die Prozesse zur Generierung und Nutzung des Aktionsplans. 71

5.1 Die Architektur des Geometrischen Agenten in der Navigationsphase. 76
5.2 Sortierung nach dem Kriterium ‚Enthaltensein'. 88
5.3 Abarbeitung einer Anweisung des Typs GO 92
5.4 Abarbeitung einer Anweisung des Typs VIEW 93
5.5 Abarbeitung einer Anweisung des Typs BE_AT. 94
5.6 Auswahl eines Fortsetzungspfades für die Folgeaktion 3. 98
5.7 Der komplexe Prozess ‚Ausführung der Folgeaktion' 100
5.8 Algorithmus für die Ablaufsteuerung. 102
5.9 Algorithmus für die Aktualisierung des internen Zustands. 103
5.10 Algorithmus für die Ausführung der Aktion 104
5.11 Algorithmus für die Auswahl und Ausführung einer Folgeaktion. 106
5.12 Die Route aus dem Beispiel. 108
5.13 Abarbeitung der Routeninstruktion aus dem Beispiel 109

Tabellenverzeichnis

3.1 Zwei Beispiele für Routeninstruktionen 32
3.2 Repräsentation als referentielles Netz und als CRIL-Netz 36
3.3 Einträge des Lexikons 38
3.4 CRIL-Repräsentation einer Wegbeschreibung nach semantischer Analyse . . . 39
3.5 Repräsentation als CRIL-Netz nach semantischer Analyse 39
3.6 Repräsentation als CRIL-Netz nach Verarbeitung mit dem GCS 40
3.7 Verarbeitung der Perzeption 43

5.1 Typen von Instruktionsanweisungen 78
5.2 Die primitiven Aktionen des Agenten 81
5.3 Vorbedingungen und Effekte von Bewegungsaktionen 82
5.4 Die Routeninstruktion aus dem Beispiel 107

Kapitel 1

Einleitung

1.1 Motivation

Das Projekt *Geometrischer Agent* beschäftigt sich mit der Verarbeitung von natürlichsprachlichen Routeninstruktionen. Im Fokus des Projekts liegen die kognitiven Prozesse, die bei der Generierung eines internen Weltmodells auf Basis der Routeninstruktion beteiligt sind, sowie die Prozesse, die die Informationen aus dieser Repräsentation bei der Navigation verwenden. Hierfür wird im Projekt ein formales Rahmenwerk geschaffen, das durch die Modellierung einer Testumgebung Methoden zur Evaluation und Validierung verschiedener Theorien über die Interpretation natürlichsprachlicher Ausdrücke und über kognitive Prozesse für die Konstruktion adäquater interner Repräsentationen ermöglicht. Die interne Repräsentation wird mit Hilfe eines Formalismus gebildet, in dem Ausdrücke mit Hilfe der sogenannten *Conceptual Route Instruction Language* (CRIL) formuliert werden. Diese Ausdrücke bilden die Basis für eine semantische und pragmatische Weiterverarbeitung. Ziel der Verabeitung ist eine adäquate formale Repräsentation sowohl der räumlichen als auch der imperativen Aspekte der Routeninstruktion.

Im Rahmen des Projekts wird eine Anwendung geschaffen (die ebenfalls den Namen *Geometrischer Agent* hat), in der ein künstlicher Agent die Aufgabe hat, in einer virtuellen planaren Umgebung unter Zuhilfenahme der CRIL-Repräsentation einer Wegbeschreibung zu navigieren. Für die Simulation wurde ein Modell des Informatikcampus Stellingen gewählt, das Probleme auf niedriger Stufe (sog. *low-level-navigation tasks*, wie z.B. Bild- bzw. Objekterkennung und Bahnplanung auf niedriger Stufe) umgeht, indem es primitive Aktionen für die Perzeption und Bewegung des Agenten bereitstellt.

Für die Navigation des Geometrischen Agenten werden Informationen aus zwei Ressourcen verwendet, die zusammen das interne Weltmodell des Agenten bilden: aus dem während der *Instruktionsphase* erzeugten *Instruktionsmodell* und dem während der *Navigationsphase* aus der Wahrnehmung des Agenten generierten *Perzeptionsmodell*. In der Repräsentation der Routeninstruktion im Instruktionsmodell kann die räumliche Information von den imperativen Anteilen unterschieden werden. In der Verarbeitung der sprachlichen Wegbeschreibung in der Instruktionsphase wird ein sogenannter CRIL-Graph erzeugt, der die räumliche Information der Routeninstruktion repräsentiert, sowie ein *Aktionsplan* für die imperativen Anteile. Der Aktionsplan repräsentiert die in der Routeninstruktion explizit genannten Aktionen, die der

Agent auszuführen hat, um sein Ziel zu erreichen. Die im Instruktionmodell repräsentierte Information ist *unsicher*, da zum einen die Pfade, auf denen sich der Agent bewegen soll, in der natürlichsprachlichen Routeninstruktion nur auf einer hohen Granularitätsebene spezifiziert werden, und somit hinsichtlich der Ausführung unterspezifiziert sind, und zum anderen Lücken in der Beschreibung und somit in dem Modell auftreten können.

Während der Navigation wird das Instruktionsmodell um ein Modell der wahrgenommenen Umgebung ergänzt. Um geeignete Repräsentationen zu erzeugen und das Problem der Unsicherheit zu handhaben, führt der Agent Inferenzen über seine internen Modelle durch, die es z.B. ermöglichen, bestimmte Pfade zu verknüpfen, Referenzsysteme auszuwählen oder Lücken in der Repräsentation der auszuführenden Aktionen zu schließen. Die Erzeugung der Repräsentationen, wie auch die Inferenzmechanismen bedienen sich dabei im hohen Maße geometrischer Konzepte. Herzstück des Geometrischen Agenten ist ein Modul, welches diese Konzepte in einer Wissensbasis bereitstellt: das GCS (Geometric Concept Specification).

In der Diplomarbeit wird ein Aktionsmodul entwickelt, welches die primitiven Aktionen des Geometrischen Agenten während der Navigation auswählt. Diese Auswahl orientiert sich am Aktionsplan und am internen Zustand des Agenten, der sein Wissen und seine Annahmen über die aktuelle Situation repräsentiert. Da die Handlungsanweisungen im Aktionsplan unterspezifiziert sind, können sie nicht wie bei klassischen Planverfahren starr ausgeführt oder statisch im voraus verfeinert werden. Vielmehr werden diese abstrakten Anweisungen im Kontext der aktuellen Situation interpretiert und durch ein dynamisches Verfahren der lokalen Planung auf konkrete, vom Agenten in der Simulation ausführbare Aktionen abgebildet. Ein derartiges Verfahren ist notwendig, da während der Navigation Informationen aus der Perzeption miteinbezogen werden, die während der Instruktionsphase noch nicht vorliegen.

1.2 Navigation

Die Kompetenz der Navigation setzt die Kombination der drei folgenden Fähigkeiten voraus ([Nehmzow 2002]):

- Kartenerstellung und Karteninterpretation (Kartenanwendung)
- Selbstlokalisierung
- Routenplanung.

Die *Karte* des Navigators ist in diesem Zusammenhang eine Abbildung der relevanten Aspekte der Umwelt auf eine interne Repräsentation (das interne *Umgebungsmodell*). Die Karte muss dabei nicht die Struktur einer typischen Landkarte besitzen; sie kann auf vielfältige Weise repräsentiert sein, z.B. als Erregungsmuster in einem neuronalen Netzwerk oder symbolisch in einer graphen- bzw. netzartigen Struktur (wie bei der CRIL-Repräsentation im Geometrischen Agenten). Die Nutzung von Karten schließt die Erzeugung bzw. Aktualisierung und Interpretation durch den Agenten mit ein. Zusätzlich muss der Navigator zur *Selbstlokalisation* fähig sein, d.h. seine aktuelle Position im Umgebungsmodell bestimmen können. Wenn zusätzlich zur eigenen Position noch der Zielpunkt repräsentiert wird, liegen die Grundsteine für die *Routenplanung* vor. Um navigieren zu können, muss ein Agent diese drei Fähigkeiten

beherrschen. In [Nehmzow 2002] werden verschiedene Strategien für die Bereitstellung dieser Kompetenzen vorgestellt, die auch schon bei Tieren beobachtet werden können:

Bei der **landmarkenbasierten Navigation** (piloting) dienen Referenzlandmarken, die anhand ihrer perzeptuellen Eigenschaften vom Navigator identifiziert werden, zur Lokalisation von Plätzen oder zur Richtungsorientierung. Dabei kann zwischen *globalen* und *lokalen* Landmarken unterschieden werden. Globale Landmarken befinden sich in relativ großer Entfernung zum Naviagtor und ihre relative Position verändert sich währen der Navigation nicht oder nur kaum. Beispiele für globale Landmarken sind Berge, die Sonne oder der nächtliche Sternenhimmel. Lokale Landmarken hingegen befinden sich relativ nah zum Navigator, so dass sich ihre relative Position während der Bewegung des Navigators zu diesem verändern kann. Eine Konstellation von lokalen Landmarken wird von manchen Tieren als ein ‚Schnappschuss' der wahrgenommenen Szene gespeichert. Wird dann eine Szene wahrgenommen, die hinreichend ähnlich zur gespeicherten Szene ist, kann der momentane Aufenthaltsort identifiziert werden ([Wehner und Räber 1979], [Trullier et al. 1997]).

Durch **Odometrie** (Wegintegration / Koppelnavigation) wird eine Navigation auch ohne die Verwendung von externen Orientierungspunkten ermöglicht. Der Navigator berechnet hierbei aus dem Startpunkt und seiner Bewegung (Richtung und Geschwindigkeit) seinen aktuellen Standort. Da diese Art der Navigation tendenziell fehleranfällig ist, da eine Einschätzung der eigenen Bewegung selten genau möglich ist, wird diese Art der Navigation oft nur in Verbindung mit anderen Strategien verwendet.

Die Verwendung von **Routen** ist eine Möglichkeit, die Komplexität der Navigationsaufgabe zu verringern. Wo manche Tiere zwangsweise an bestimmte Routen gebunden sind (z.B. Lachse an den Fluss beim Aufsuchen ihrer Laichstätte), nutzen andere Tiere gut erkennbare Routen als Orientierungshilfe (wie Tafelenten, die dem Mississippital folgen, oder Bienen, die entlang von Waldrändern fliegen ([Waterman 1989])). Da Routen eine effektive Möglichkeit bereitstellen, Schwierigkeiten bei der Navigation zu minimieren, ist es nicht verwunderlich, dass Menschen ihre Besiedlung oft entlang fester Routen (Flüsse, Straßen) angeordnet haben.

Exploration ist eine Möglichkeit, ohne strukturiertes Vorwissen nach einem Ziel zu suchen. Durch Exploration wird zusätzliches räumliches Wissen angesammelt.

Oftmals findet sich bei Tieren eine Kombination der unterschiedlichen Strategien, je nachdem, welche Möglichkeiten die Umgebung bereitstellt. Durch die Kombination wird auch eine höhere Robustheit erreicht ([Nehmzow 2002]).

Für den Geometrischen Agenten ist die Verwendung von Landmarken, Routen und das Prinzip der Exploration vorgesehen. Die Routen, auf denen sich der Agent bewegen kann, entsprechen den gepflasterten Wegen, die den Informatikcampus vernetzen (Siehe Abb. 2.2). Diese Wege können auch die Funktion von Landmarken übernehmen. Als weitere Landmarken dienen in der aktuellen Version Gebäude auf dem Campus. Zusätzlich kann Exploration für den Agenten sinnvoll sein, wenn er weitere Informationen über die Umgebung benötigt.

1.3 Bezüge zur Robotik

Da das Thema dieser Arbeit mit typischen Problemstellungen der Robotik verwandt ist, soll hier gezeigt werden, wie das Thema in das umfassende (und nicht klar abgegrenzte) Gebiet der Robotik eingeordnet werden kann. Dabei gibt es Ähnlichkeiten, aber auch Unterschiede

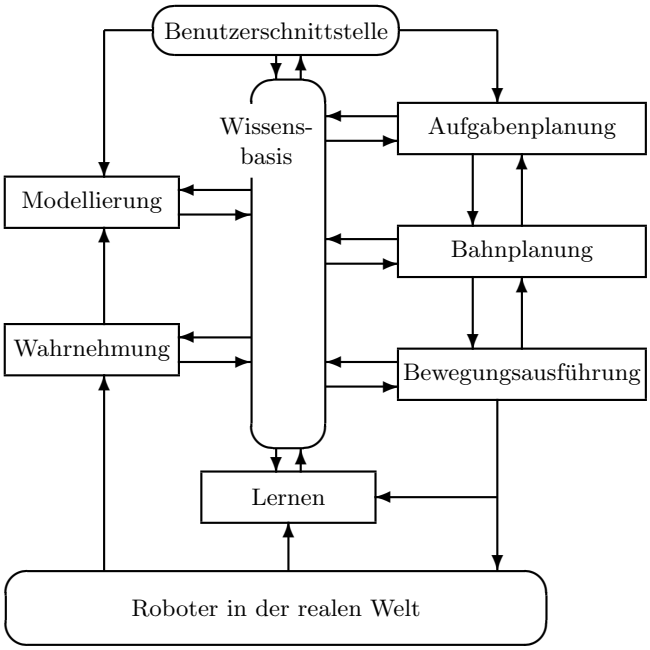


Abbildung 1.1: Hierarchische Steuerungsstruktur eines autonomen Robotersystems. Abbildung nach [Zhang 1995], S. 24.

zu Aspekten, die typischerweise im Fokus der Robotikforschung liegen.

Die *Robotik* ist ein breites interdisziplinäres Forschungsgebiet, welches das Ziel besitzt, intelligente Maschinen zu entwickeln, die ihre Umgebung wahrnehmen, aufgrund der Wahrnehmung autonome Entscheidungen treffen und schließlich in der Umgebung Handlungen ausführen. Ihren Beitrag dazu leisten Fachrichtungen aus verschiedenen Disziplinen wie der Informatik, Biologie, Kybernetik, Kognitionspsychologie und Philosophie, die sich mit Regelungs- und Steuertechnik, Sensorik und Themen der Künstlichen Intelligenz-Forschung (KI) auseinandersetzen ([Zhang 1995]). Das Fachgebiet der *Mobilen Robotik* setzt sich insbesondere mit der Aufgabe der Bewegung von Robotern von einem Ort zu einem anderen auseinander ([Nehmzow 2002]).

Um der hohen Komplexität von natürlichen Umgebungen gerecht zu werden, bietet sich das Prinzip der *Modularisierung* an ([Zhang 1995]). Durch die Modularisierung entstehen mehrere Ebenen der Verarbeitung, die hierarchisch geordnet sind (Siehe Abb. 1.1).

Welche Module bei der Modellierung des Geometrischen Agenten eine Rolle spielen, soll im Folgenden gezeigt werden. Die verschiedenen Module sind auf die Verarbeitung besonderer Umweltaspekte spezialisiert und geben ihre Ergebnisse entweder direkt an anliegende Module oder indirekt über eine zentrale **Wissensbasis** weiter.

Das Modul **Wahrnehmung** verarbeitet die Daten der Robotersensoren. Die Aufbereitung der Daten kann dabei Ergebnisse auf verschiedenen Abstraktionsebenen liefern. So können Daten auf einer niedrigen Abstraktionsebene an die Module für die Bahnplanung oder Bewegungsausführung weitergericht werden, und Daten auf höherer Abstraktionsebene als Basis für die Modellierung der Umwelt dienen.

Die durch das Wahrnehmungsmodul aufbereiteten Daten dienen zur Erstellung bzw. Aktualisierung des internen Weltmodells (**Modellierung**). Das Weltmodell repräsentiert eine abstrakte Sicht auf die Umgebung, die alle für die Weiterverabeitung in höheren Modulen relevanten Informationen enthält.

Die Komponente **Aufgabenplanung**[1] legt einen übergeordneten Handlungsplan für eine dem System gestellte Aufgabe fest. Dieser Plan beschreibt, welche Ziele der Roboter als nächtes zu verfolgen hat (z.B. die Bewegung von einem Punkt A zu einem nächsten Punkt B). Diese Art der Planung basiert hauptsächlich auf symbolischer Repräsentation und Inferenz hierüber.

Da der Aufgabenplan auf abstrakter Ebene vorliegt und nicht direkt in der realen Umgebung ausführbar ist, findet durch ein Modul **Bahnplanung** die Umsetzung des übergeordneten Plans statt. So wird z.B. die Aufgabe, sich von A nach B zu bewegen, in eine geometrische Bahn umgewandelt, die bestimmte Anforderungen wie z.B. das Ausweichen von Hindernissen erfüllt. Auf dieser Ebene spielen hauptsächlich topologische und geometrische Repräsentationen eine Rolle.

Das Modul **Bewegungsausführung** steuert dann mit Hilfe der geometrischen Bahn und den aktuellen Sensorwerten unter Berücksichtigung der kinematischen und dynamischen Randbedingungen die Aktoren des Roboters an.

Zusätzlich gibt es noch das Modul der **Benutzerschnittstelle**, welches eine Veränderung von Steuerungsparametern und Algorithmen durch den Benutzer ermöglicht, sowie evtl. ein Modul **Lernen**, welches diese Modifikationen selbst vornimmt.

Die Probleme, die auf den verschiedenen Ebenen auftreten, sind dabei sehr unterschiedlich. Auf höheren bzw. abstrakteren Ebenen muss z.B. geklärt werden, welche Aspekte der Wahrnehmung für die weitere Verarbeitung relevant sind und im Weltmodell repräsentiert werden müssen, und welche Inferenz- und Planungsmechanismen mit Hilfe des Weltmodells im Aufgabenplaner angewandt werden können. Auf der unteren Ebene hingegen liegen die Sensorik- und Aktorikkomponenten des Roboters im Fokus der Untersuchung; dabei ist z.B. zu klären, welche Aktoren des Roboters zu welchem Zeitpunkt auf welche Weise angesteuert werden und wie dabei die Echtzeitfähigkeit gewährleistet werden kann. Zusätzlich gibt es noch Fragen, die auf den dazwischenliegenden Ebenen auftreten, wie z.B. quantitative Sensordaten in qualitative Konzepte übersetzt werden können ([Liu und Daneshmend 2004]), oder wie die Integration der verschiedenen Ebenen vorzunehmen ist ([Zhang 1995]).

Ziel der KI- und Robotikforschung ist es, diese Probleme durch geeignete Methoden zu lösen und das gewünschte Verhalten des Roboters zu erzeugen. Diese Methoden werden in Simulationen oder in der Verwendung in Robotern, die in der ‚wirklichen' Welt wahrnehmen und agieren, angewandt und verifiziert.

Untersuchungsgegenstand des Projekts *Geometrischer Agent* und auch dieser Arbeit sind Verfahren, die die Modellierung, Wahrnehmung, Aufgabenplanung sowie bestimmte Aspekte der Bahnplanung vornehmen. Ebenso wird eine Wissensbasis bereitgestellt, die zum einen sprach- und domänenspezifisches Wissen bereitstellt, und zum anderen das Umgebungsmodell und den Aktionsplan des Agenten beinhaltet. Die Modellierung wird durch den CRIL-Formalismus geleistet. Durch die Verarbeitung der natürlichsprachlichen Routeninstruktion wird ein formales Modell der Routeninstruktion generiert, welche die räumliche Informati-

[1] In [Nehmzow 2002] *Routenplanung* genannt.

on repräsentiert. Zusätzlich wird bei der Verarbeitung der Aktionsplan erzeugt, der die zu tätigenden Handlungen des Agenten auf abstrakter Ebene beschreibt und somit dem Ergebnis der Aufgabenplanung entspricht. Da die Handlungen im Aktionsplan abstrakt sind, muss eine Umsetzung in konkrete Aktionen vorgenommen werden. Diese Aufgabe der Bahnplanung wird durch das Aktionsmodul gelöst, indem dieses die abstrakten Anweisungen aus dem Aktionsplan im Kontext der aktuellen Situation interpretiert und auf primitive Aktionen des Agenten abbildet.

1.4 Vorgaben

Im Projekt sind die Module, die die Modellierung und Aufgabenplanung bereitstellen, sowie die Wissensbasis prototypisch ausgearbeitet, so dass bestimmte Vorgaben für die Weiterverarbeitung im Aktionsmodul gegeben sind. Da das Wissen, das für den Geometrischen Agenten von besonderer Bedeutung ist, geometrischer und topologischer Natur ist, wird durch die Modellierung eine symbolische Repräsentation geometrischer und topologischer Konzepte bereitgestellt. Inferenz und Aufgabenplanung finden dann mit Hilfe dieser Konzepte statt.[2] Ergebnis der Verarbeitung der Routeninstruktion ist das Instruktionsmodell, welches das räumliche Modell und den Aktionsplan beinhaltet. Dieses Modell und das Modell, das während der Navigation durch die Perzeption gewonnen wird, stellen die Information zur Verfügung, die vom Aktionsmodul bei der Navigation verwendet werden.

Neben dieser Repräsentation der für die Navigation relevanten Informationen ergeben sich durch die Modellierung der simulierten Umgebung weitere Vorgaben, die die Arbeitsweise des Aktionsmoduls maßgeblich beeinflussen:

- In der Simulation des Campus werden nur bestimmte Objekte repräsentiert. Diese Objekte sind Landmarken (Häuser) und Wege, auf denen sich der Agent bewegen kann. Nicht repräsentiert werden z.B. Grünflächen oder andere Agenten bzw. Personen.

- Der Agent ist zu perzeptiven Leistungen fähig, wie z.B. Landmarken und Wege als solche zu erkennen. Insbesondere kann er ein Objekt, welches er einmal gesehen hat, in der nächsten Perzeptionshandlung wieder als dasselbe Objekt erkennen. Dies ist wichtig, da der Agent über keine kontinuierliche Wahrnehmung verfügt, sondern nur zu bestimmten Zeitpunkten Perzeptionshandlungen ausführt.

- Das Verfahren, das ein Wiedererkennen der Objekte aus der Routeninstruktion in der Umgebung leistet ist in ([Helwich 2003]) prototypisch ausgearbeitet. Dieses Verfahren für die *Auflösung von Koreferenzen* vergleicht für eine Verknüpfung von Objekten aus der Instruktion und der Perzeption entsprechende als relevant bewertete CRIL-Teilnetze

[2] Dies soll nicht im Widerspruch zur obigen Unterteilung stehen, in der geometrische und topologische Wissensverarbeitung der Ebene der Bahnplanung zugeordnet werden. In dem Fall eines Roboters, der das Modul des Geometrischen Agenten integriert, würden auf beiden Ebenen geometrische sowie topologische Repräsentationen eine Rolle spielen, allerdings auf verschiedenen Granularitätsebenen. So kann auf höherer Ebene eine Anweisung lauten ‚Bewege Dich auf dem Pfad, der sich zwischen den beiden Häusern befindet.' (und somit ordnungsgeometrische Konzepte wie *zwischen* beinhalten), die aber auf der Ebene der Bahnplanung mit Hilfe von weiteren, auf niedriger Granularität basierender topologischer und geometrischer Konzepte verfeinert wird.

und ordnet diesen einen strukurellen Ähnlichkeitswert zu. Diese Herangehensweise muß bei den Prozessen, die sich dem Prozess der Koreferenzauflösung anschließen und dessen Ergebnis verwenden, berücksichtigt werden.

- Eine primitive Bewegungsaktion des Agenten wird als eine abgeschlossene Bewegung auf einem fest vorgegebenen Wegstück simuliert. Dadurch können bestimmte Probleme der Bahnplanung wie das Ausweichen von Hindernissen oder die Ermittlung der Trajektorie, die einem Wegstück entspricht, wie auch Probleme, die bei der Bewegungsausführung auftreten, ausgeschlossen werden.

Durch die Abstraktion von Aspekten, denen die Robotikforschung in letzter Zeit besondere Aufmerksamkeit gewidmet hat, ist die Problemstellung momentan eher in den Bereich der Künstlichen Intelligenz-Forschung bzw. Wissensrepräsentation anzusiedeln, die für ihre Domäne den Begriff des *Agenten* dem des *Roboters* vorzieht.[3]

1.5 Was *nicht* behandelt wird

Nicht behandelt werden insbesondere Probleme, die auf den unteren Ebenen der Darstellung 1.1 auftauchen. Durch die spezifische Modellierung der Umgebung sind die Kompetenzen, die die Module der unteren Ebenen (Wahrnehmung, Bewegungsausführung und bestimmte Aspekte der Bahnplanung) bereitstellen, nicht von Interesse und werden in dieser Arbeit nicht weiter berücksichtigt. Es wird vielmehr angenommen, dass eine Schnittstelle zwischen dem Modul des Geometrischen Agenten und den unteren Schichten eine Kommunikation und Integration dieser verschiedenen Schichten ermöglichen könnte. Aber auch die Spezifikation einer derartigen Schnittstelle kann in dieser Arbeit nicht geleistet werden.

Desweiteren werden hardwarespezifische Themen nicht behandelt (wie z.B. Aufbau und Funktionsweise von Sensoren und Aktoren), da sie für die Modellierung des Geometrischen Agenten und das Projekt keine Relevanz besitzen.

Ebenso wird in der Arbeit von den Prozessen der Spracherkennung auf niedriger Ebene abstrahiert. Die Wegbeschreibungen, die im Projekt Verwendung finden, liegen in textueller Form vor, bzw. werden dem Programm des Geometrischen Agenten über die Tastatur mitgeteilt.

1.6 Aufbau der Arbeit

Das zweite Kapitel *Der Geometrische Agent* klärt den Begriff des Agenten in der Informatik. Dabei wird gezeigt, wie bestimmte Eigenschaften von Agenten und deren Umgebungen zu bewerten sind und wie der Geometrische Agent in die resultierenden Kategorien eingeordnet werden kann.

Im dritten Kapitel *Verarbeitung der Routeninstruktion* wird das Verfahren vorgestellt, das die natürlichsprachliche Routeninstruktion in ein formales Modell, das in CRIL formuliert ist,

[3]Der Geometrische Agent ist insbesondere kein Geometrischer *Roboter*, da er in einer rein simulierten Umgebung situiert ist. Der Agentenbegriff ist weiter gefasst als der des Roboters und beinhaltet auch autonome Programme, die in einer rein virtuellen Umwelt agieren (siehe Abschnitt 2.1 *Agentensysteme*).

umwandelt. Dazu gehören insbesondere der Formalismus zur Darstellung des Weltmodells, sowie die Prozesse, die bei der Erzeugung der internen Repräsentationen beteiligt sind.

Ziel des Aktionsmoduls ist es, primitive Aktionen des Agenten auszuwählen und deren Ausführung anzustoßen. Da die Auswahl von Aktionen eines Agenten in das umfassende Themengebiet der Planung fällt, werden im vierten Kapitel *Planungsverfahren unter Unsicherheit* zum einen etablierte Planungskonzepte der KI und zum anderen konkrete Systeme vorgestellt, die speziell unter Berücksichtigung des Problems der Unsicherheit entwickelt wurden. Dabei wird untersucht, welche dieser Konzepte für eine Verwendung im Geometrischen Agenten geeignet sind.

Das fünfte Kapitel *Das Aktionsmodul* widmet sich dem Modul des Agenten, welches für die lokale Planung, d.h. für die Abarbeitung des Aktionsplans zuständig ist. Das Aktionsmodul stellt hierfür eine Ablaufsteuerung bereit, die die verschiedenen Prozesse und primitiven Aktionen anstößt, die während der Navigation des Agenten auszuführen sind.

Zur Notation

Kursive Schrift wird bei erster Nennung wichtiger Begriffe, sowie bei Eigennamen und für die Hervorhebung verwendet. Konzepte, die in einem Absatz beschrieben werden, der nicht durch eine eigene Überschrift gekennzeichnet ist, werden zur Hervorhebung **fett** dargestellt. Formulierungen einer formalen Repräsentationssprache (z.B. CRIL) werden in serifenloser Schrift dargestellt. Für Bezeichnungen, die Komponenten des implementierten Moduls entsprechen oder Algorithmen beschreiben, findet `Maschinenschrift` Verwendung. Der ‚Geometrische Agent' wird *kursiv* geschrieben, wenn es sich um den Projekt- oder Programmnamen handelt, jedoch in normaler Schrift, wenn der Agent des Programms gemeint ist.

Übersetzungen aus dem Englischen

In Fällen, wo mir eine Übersetzung aus dem Englischen ins Deutsche sinnvoll erschien, ist hinter dem deutschen Begriff der englischen Originalbegriff, wie ihn der entsprechende Autor verwendet hat, kursiv in Klammern gesetzt.

Bei englischen Begriffen, die sich auch im Deutschen etabliert haben, oder in Fällen, in denen ich keine sinnvolle Übersetzung ins Deutsche gefunden habe, wird der englische Originalbegriff verwendet.

Kapitel 2

Der Geometrische Agent

2.1 Agentensysteme

Die Idee des intelligenten Agenten hat eine zentrale Rolle in der Künstlichen Intelligenz-Forschung eingenommen ([Russell und Norvig 2003]). Mit spezifischen Eigenschaften, durch die sich intelligente Agenten von ‚gewöhnlichen' Computerprogrammen unterscheiden, soll dabei ein Verhalten künstlich erzeugt werden, welches hinreichend komplexen Umgebungen gerecht wird ([Wooldridge 1999], [Russell und Norvig 2003]). Das Verständnis darüber, welche dieser Eigenschaften notwendig und hinreichend für den entscheidenen Schritt für intelligentes Verhalten sind, divergiert: Von einfacher Autonomie bzw. *Proaktivität*[1], bis hin zur Modellierung kognitiver und gar mentaler Prozesse (um *menschliches* Verhalten zu simulieren) reicht die Spanne spezifizierter und implementierter Konzepte.

Poole, Mackworth und Goebel unterscheiden in ihrem Lehrbuch „Computational Intelligence" zuerst nicht zwischen natürlichen und künstlichen Agenten:

> An agent is something that acts in an environment - it does something. Agents include worms, dogs, thermostats, airplanes, humans, organizations, and society. An intelligent agent is a system that acts intelligently: What it does is appropriate for its circumstances and its goal, it is flexible to changing environments and changing goals, it learns from experience, and it makes appropriate choices given perceptual limitations and finite computation.[2]

In dem Lehrbuch „Artificial Intelligence" geben Russell und Norvig folgende Definition:

> An Agent is just something that acts (*agent* comes from the Latin *agere*, to do). But computer agents are expected to have other attributes that distinguish them from mere "programs", such as operating under autonomous control, perceiving their environment, persisting over a prolonged time period, adapting to change, and being capable of taking on another's goals. A rational agent is one that acts so as to achieve the best outcome or, when there is uncertainty, the best expected outcome.[3]

[1] Das bedeutet, das der Agent von sich heraus handelt, ohne von außen zum Handeln aufgefordert zu werden.
[2] [Poole et al. 1998], S. 1
[3] [Russell und Norvig 2003], S. 4, Hervorhebungen im Original.

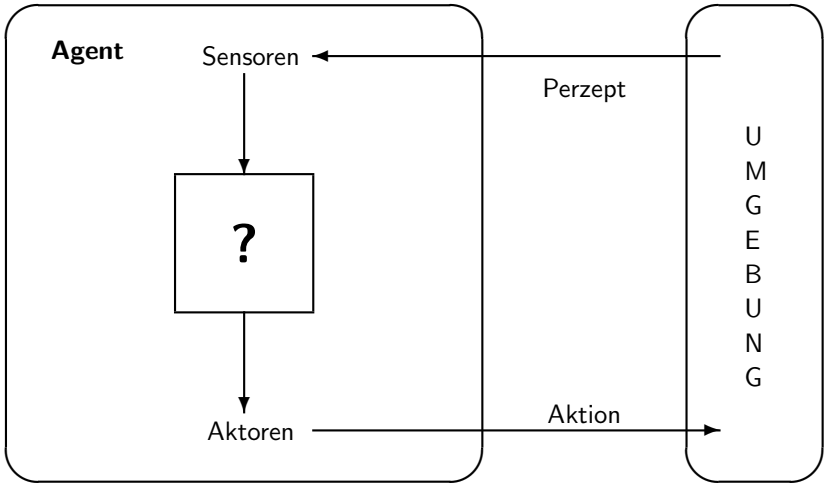


Abbildung 2.1: Interaktion des Agenten mit seiner Umwelt. Das Fragezeichen steht hier als Platzhalter für die *Agentenfunktion*, d.h. die Funktion, die die Perzeption des Agenten auf seine Aktionen abbildet. Abbildung nach [Russell und Norvig 2003] (Abb. 2.1).

Eine ähnliche Definition des Begriffs Agent findet man bei Wooldridge:

> An *agent* is a computer system that is situated in some environment, and that is capable of outonomous action in this environment in order to meet its design objectives.[4]

Obwohl sich die Definitionen im Detail unterscheiden, sind jedoch zwei Konzepte in diesen Definitionen elementar für das Wesen eines Agenten: die Einbettung in eine Umgebung und die Fähigkeit, in dieser Umgebung autonom zu handeln. Um autonom handeln zu können, muss der Agent die Fähigkeit besitzen, für ihn relevante Aspekte seiner Umwelt wahrzunehmen, die seine Entscheidungen bezüglich seiner auszuführenden Handlungen beeinflussen. Dies wird durch die Abbildung 2.1 schematisch dargestellt.

Poole, Mackworth und Goebel sprechen in ihrer Definition von *intelligenten Agenten*. Da der Begriff der Intelligenz jedoch weiterhin mit Problemen behaftet ist[5], soll der Geometrische Agent einem anderen Kriterium genügen: dem der *Rationalität*. Russell und Norvig kommen in ihrer Definition auf das Konzept des rationalen Agenten zu sprechen, und Wooldridge entwickelt in [Wooldridge 2000] eine modallogische Sprache, mit der rationales Verhalten von Agenten spezifiziert werden soll.

Rationales Verhalten von künstlichen Agenten zu verlangen, hängt mit der *intentional stance* gegenüber anderen Lebewesen oder auch Computersystemen zusammen ([Dennett 1987]). Um die Funktionsweise eines Systems zu verstehen und dieses zu akzeptieren, nimmt man eine Haltung ein, die dessen Intentionalität (einfach ausgedrückt: die

[4] [Wooldridge 1999], S. 5, Hervorhebungen im Original.

[5] Der Begriff der Intelligenz ist durch die enge Verknüpfung an menschliche Fähigkeiten vielschichtig und schwer zu objektivieren. Die Schwierigkeit des Problems wird beispielsweise durch die in der KI legendär gewordene Debatte zwischen A. Turing, J. Searle und A. Church deutlich. (Vgl. z.B. [Turing 1950] und [Searle 1980]).

Zielgerichtetheit auf bestimmte Dinge oder Wirkungsweisen[6]) zu ergründen versucht. Ein derartiges Verstehen und die damit verbundene Akzeptanz wird durch rationales Verhalten erleichtert. Im folgenden wird ein pragmatisches Verständnis des Konzepts der Rationalität vorgestellt; philosophische bzw. erkenntnistheoretische Probleme werden dabei außer Acht gelassen.

2.2 Das Rationalitätskriterium

Rationales Verhalten wird als dasjenige Verhalten charakterisiert, welches ein geeignetes Mittel darstellt, um ein bestimmtes Ziel zu erreichen. Wenn man z.B. das Ziel hat, ein Buch zu beschaffen, ist es rational, eine Bibliothek oder eine Buchhandlung aufzusuchen, irrational hingegen, wenn man zur Erfüllung dieses Ziels in den Zoo ginge. Auf das Problem, dass auch rationales Verhalten nicht immer zum Erfolg führt (wenn das Buch ausgeliehen bzw. vergriffen ist), wird weiter unten eingegangen.

Rationales Verhalten vom künstlichen Agenten wird durch vier Aspekte bestimmt ([Russell und Norvig 2003]):

- Das Performanzmaß
- Das *a-priori* Wissen des Agenten über die Umwelt
- Die Aktionen, die der Agent ausführen kann
- Die aktuelle Perzeptionssequenz des Agenten

Das Performanzmaß gibt ein Kriterium vor, durch welches sich der Erfolg oder Misserfolg des Agenten messen lässt. Typischerweise wird ein Performanzmaß beim Entwurf eines Agenten vom Designer von außen vorgegeben. Verschiedene Agenten haben verschiedene Aufgaben und da auch die Beschaffenheit der Umgebung das Performanzkriterium beeinflusst, ist es sinnvoll, für verschiedene Agenten verschiedene Performanzmaße festzulegen.

Um autonom handeln zu können, benötigt der Agent Informationen. Zum einen besitzt der Agent Wissen, über das er vor der Ausführung jeglicher Perzeptionshandlungen in der Umwelt verfügt (a-priori Wissen). Dieses Wissen beinhaltet z.B. Informationen für die Kategorisierung von perzipierbaren Objekten (z.B. in Form einer Taxonomie). Zum anderen gewinnt der Agent Informationen durch die Perzeption seiner Umgebung. Die aktuelle Perzeptionssequenz repräsentiert die Folge von wahrgenommenen Zuständen.

Schließlich bestimmen noch die ausführbaren Aktionen das Kriterium der Rationalität. Wenn ein Agent keine geeigntete Methode besitzt, um eine Aufgabe zu lösen, dann ist rationales Verhalten für ihn in diesem Fall unmöglich.

Dies führt Russell und Norvig zu folgender Definition eines *rationalen Agenten*:

> For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has.[7]

[6] Dies wird nur einem Aspekt des vielschichtigen Begriffs gerecht. Vgl. hierzu z.B. [Searle 1983].
[7] [Russell und Norvig 2003], S. 36.

Auffallend ist, dass in der Definition die Formulierung „...an action that is *expected* to maximize its performance measure“ verwendet wird. Ein Agent, der über vollständiges und sicheres Wissen über seine Umwelt verfügt, besitzt den Status der *Allwissenheit* (*omniscience*). Ein allwissender Agent kann zu jedem Zeitpunkt eine Aktion auswählen, die sein Performanzmaß maximiert, da ihm alle Umstände seiner Umgebung und alle Konsequenzen seines Tuns bekannt sind. Ein Agent, der mit dem Problem der Unsicherheit konfrontiert ist, da er nur über unvollständiges Wissen über seine Umwelt verfügt und dessen Handlungen auch einen anderen als den erwarteten Effekt haben können, kann eben nur erwarten, dass die Ausführung einer Aktion den gewünschten Erfolg hat. Allwissenheit ist nur bei einfachen Umgebungen und Problemstellungen möglich; der Geometrische Agent besitzt diesen Status nicht.

Die oben genannten Punkte sind auch für den Geometrischen Agenten entscheidend. Da Aktion und Perzeption des Agenten eng mit der Umgebung, in die er eingebettet ist, zusammenhängen, werden im nächsten Abschnitt Eigenschaften der Umgebung des Agenten untersucht. Auf welche Weise der Agent in der Simulation perzipiert und agiert, wird in Abschnitt 5.2 *Die primitiven Aktionen des Agenten* vorgestellt. Das a-priori-Wissen über die Umwelt ist auf zweierlei Weise im Agenten repräsentiert. Zum einen besitzt der Agent unabhängig von der konkreten Zielvorgabe linguistisches, räumliches und ontologisches Wissen, welches er verwendet, um ein internes Modell der Routeninstruktion aufzubauen und Inferenzen durchzuführen. Zum anderen besitzt der Agent durch das Instruktionsmodell spezifisches Wissen über eine Route, die er konkret für seine Aufgabenstellung verwenden kann. Diese Aspekte werden in Abschnitt 3.3 *Die Instruktionsphase* vorgestellt. Welches ein geeignetes Performanzmaß für den Geometrischen Agenten darstellt, wird in Abschnitt 2.6 *Ein Performanzmaß für den Geometrischen Agenten* gezeigt.

2.3 Die Umgebung des Geometrischen Agenten

Der Geometrische Agent ist ein Softwareagent. Perzeption und Aktion finden in einer simulierten Umgebung statt, die im Rahmen der planaren Euklidischen Geometrie spezifiziert ist. Objekte in der virtuellen Umgebung haben geometrische Eigenschaften wie Form und Position und werden mit Hilfe von Punkten, Linien oder Polygonzügen repräsentiert. Die Objekte sind in einem absoluten Koordinatensystem positioniert, so dass metrische Aspekte wie z.B. Entfernungen, die Länge von Wegstücken oder Winkel zwischen Wegen repräsentiert werden. Nichtgeometrische Aspekte wie Klassenzugehörigkeit oder Namen von Objekten können ebenfalls dargestellt werden. Ein Sonderfall ist das Attribut ‚Höhe‘, welches eigentlich geometrisch ist, aber durch die Modellierung in einer planaren Umgebung wie ein nicht-geometrisches Attribut behandelt wird.

Als Grundlage für die Simulationsumgebung wurde der Informatikcampus der Universität Hamburg in Stellingen gewählt. Abb. 2.2a zeigt eine Karte. Das Umgebungsmodell, welches die für den Agenten wesentlichen Komponenten beinhaltet, wird durch Abb. 2.2b dargestellt. Im Umgebungsmodell können Landmarken, Entscheidungspunkte, Wegstücke, Routensegmente, Pfade und der Agent unterschieden werden (siehe Abb. 2.3):

- **Landmarken**: Der Agent orientiert sich während der Naviagtion mit Hilfe von Landmarken, die in der Routeninstruktion genannt werden. Als Landmarken fungieren

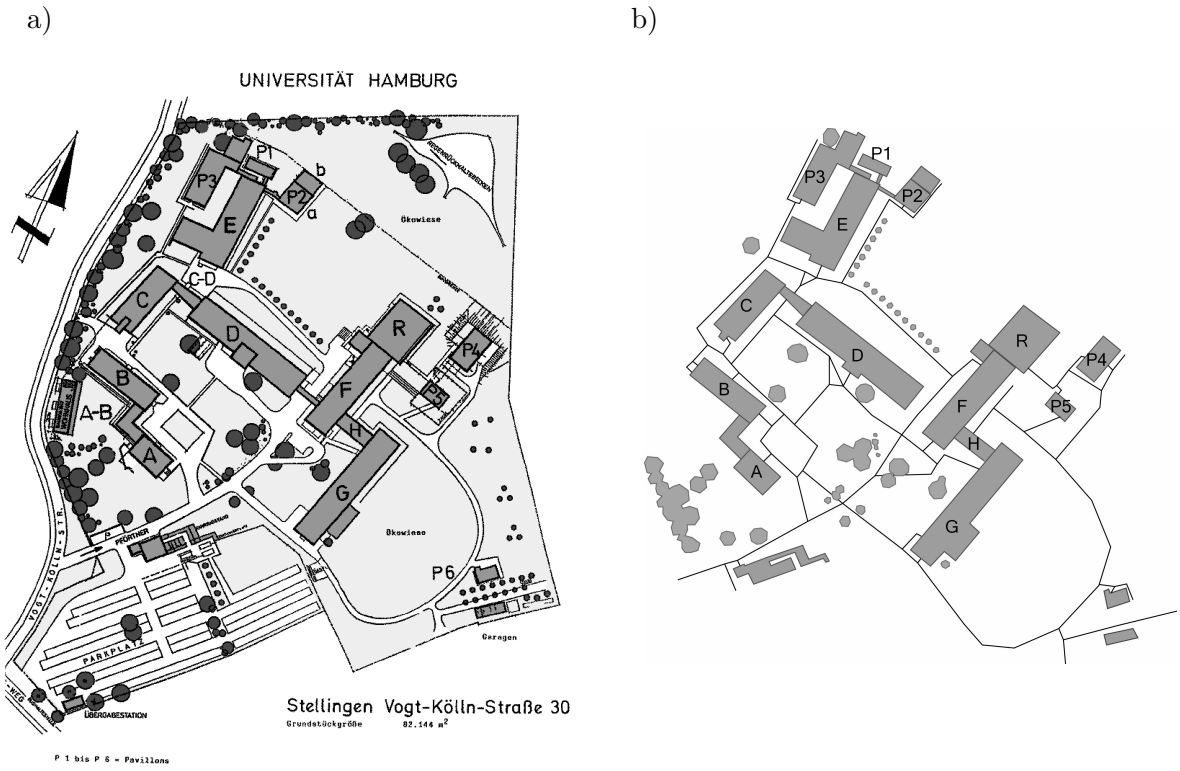


Abbildung 2.2: Der Informatikcampus Stellingen. a) zeigt eine Karte, b) die schematische Darstellung des Modells der Simulation.

Gebäude und Wegstücke. Der Agent kann bei der Perzeption eines Gebäudes dessen Namen erkennen. Diese Vereinfachung im Modell ist legitim bzw. *epistemisch adäquat*[8], da die Häuser auf dem realen Informatikcampus gut sichtbar mit Namen (einzelne Großbuchstaben, z.B. ‚A') bezeichnet sind.

- **Entscheidungspunkte**: Punkte, an denen der Agent bei der Bewegung eine Entscheidung bezüglich der Route treffen kann, heißen Entscheidungspunkte. Im Modell des Campus sind dies Orte, an denen mehrere Wegstücke aufeinandertreffen.

- **Wegstücke**: Ein Wegstück beinhaltet die direkte Verbindung eines Entscheidungspunktes mit einem weiteren Entscheidungspunkt, ohne dass innerhalb der Verbindung weitere Entscheidungspunkte passiert werden, sowie die beiden Entscheidungspunkte. Als Wegstücke kommen nur modellierte Wege infrage, die den vorgegebenen gepflasterten Wegen des Campus entsprechen.

- **Routensegmente**: Ein Routensegment entspricht entweder einem Wegstück oder setzt sich aus einer linearen Abfolge mehrerer verbundener Wegstücke zusammen. Im ersten

[8]Der Begriff der *epistemischen Adäquatheit* wird verwendet, um zu beschreiben, wie gut ein Modell ein reales Problem approximiert. Vgl. z.B. [Lazanas 1995], S. 5.

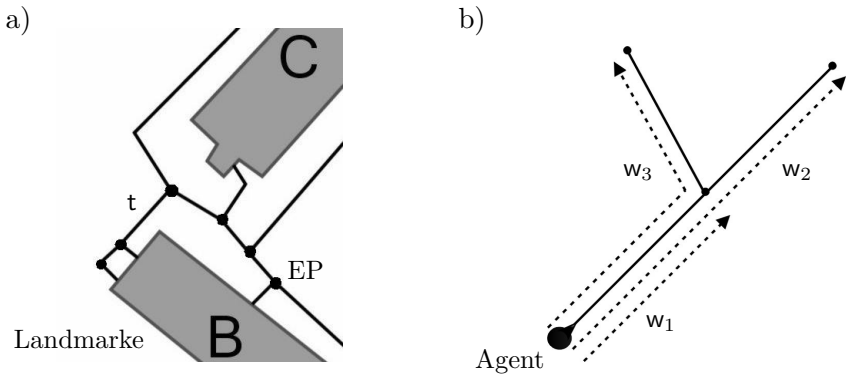


Abbildung 2.3: Die Komponenten der simulierten Umgebung. a) zeigt Landmarken (grau unterlegt), Entscheidungspunkte (EP) und Wegstücke (t). In b) werden drei mögliche Pfade dargestellt (w_1, w_2 und w_3), die ihren Startpunkt in der momentanen Position des Agenten haben und entlang dreier Routensegmente verlaufen.

Fall besteht es aus zwei Entscheidungspunkten (die das Routensegment an beiden Enden abschließen) sowie dem die Entscheidungspunkte verbindenen Wegstück, im zweiten Fall aus mehreren Entscheidungspunkten und den Wegstücken, die die Entscheidungspunkte miteinander verbinden.

- **Pfade**: Ein Pfad entspricht einer Trajektorie eines bewegten Objekts. Pfade besitzen einen Start- sowie einen Endpunkt. Pfade und Wegstücke können aufgrund ihrer verschiedenen Eigenschaften nicht gleichgesetzt werden; vielmehr kann ein Pfad entlang einer Straße verlaufen. Da der Agent durch die Perzeption der Umgebung nur statische Objekte wie Routensegmente wahrnehmen kann, werden für den Prozess, der die Instruktion mit der Perzeption verknüpft, automatisch die entsprechenden potentiellen Pfade erzeugt, die entlang dieser Routensegmente verlaufen und ihren Startpunkt in der aktuellen Position des Agenten haben (Siehe Abb. 2.3b).

- **Route**: Eine Route ist eine lineare Sequenz von aufeinanderfolgenden Routensegmenten.

- **Bäume**: Bäume schränken in der Simulation die Sicht des Agenten ein, indem sie hinter ihnen liegende Objekte verdecken.

- **Der Agent**: Der Agent besitzt im Modell eine Position und eine Orientierung (jedoch keine Ausdehnung). Dargestellt wird er als ein kleiner schwarzer Kreis, der mit einer Spitze versehen ist, die die Orientierung des Agenten anzeigt.

Die simulierte Umgebung wird durch das Programm zusätzlich auf dem Computermonitor dargestellt, so dass das Verhalten des Agenten während eines Navigationsdurchlaufs überwacht werden kann.

2.4 Umgebungstypen

Die Eigenschaften der Umgebung spielen eine entscheidende Rolle beim Entwurf eines Agenten, da sie bestimmen, welche Aspekte der Umgebung durch den Agenten perzipierbar sind und welche primitiven Aktionen der Agent beherrschen muss, um seinen Aufgaben gerecht zu werden ([Russell und Norvig 2003]). Die Umgebung bestimmt zudem mit, welche Planungsverfahren angemessen sind (Siehe Kapitel 4). Ebenso beeinflusst sie die Agentenfunktion (die direkt von der Perzeption und Aktion abhängig ist) und das Performanzmaß. Deshalb ist eine Einordnung der Umgebung des Geometrischen Agenten in die Vielzahl von möglichen Umgebungstypen sinnvoll. Verschiedene Arten von Umgebungen lassen sich laut Russell und Norvig anhand von sechs orthogonalen Dimensionen unterteilen:

Vollständig beobachtbar vs. **partiell beobachtbar**: Eine Umgebung ist vollständig beobachtbar, wenn der Agent zu jedem Zeitpunkt dank seiner Perzeption vollständiges Wissen über den Zustand der Umgebung erlangen kann. Die Umgebung des Geometrischen Agenten ist nur partiell beobachtbar, da der Agent zu jedem Zeitpunkt nur einen Ausschnitt der Welt sieht. Bestimmte Teile der Umgebung (die er noch nicht exploriert hat), sind ihm unbekannt.

Deterministisch vs. **stochastisch**: Eine Umgebung heißt deterministisch, wenn der nächste Zustand der Welt nur vom jetzigen Zustand und der vom Agenten ausgeführten Aktion abhängen. Andernfalls ist sie stochastisch. Die Umgebung des Geometrischen Agenten ist deterministisch, da das einzige bewegliche Objekt der Agent selbst ist.

Statisch vs. **dynamisch**: Statische Umgebungen verändern ihren Zustand nicht, während der Agent überlegt. In dynamischen Umgebungen kann währen der Informationsverarbeitung (z.B. durch andere Agenten) der Zustand der Welt verändert werden, so dass im ungünstigen Fall alle Mühe umsonst war, da das Ergebnis nicht mehr anwendbar ist. Dies ist für den Geometrischen Agenten kein Problem. Da seine Welt deterministisch ist, ist sie auch statisch.

Episodisch vs. **sequentiell**: In einer episodischen Umgebung ist es möglich, die Erfahrung des Agenten in atomare Einheiten zu unterteilen. Das bedeutet, dass nur das aktuelle Perzept des Agenten seine nächste Aktion beeinflusst und das letzte Perzept schon vergessen sein kann. In einer sequentiellen Umgebung hingegen ist eine Aktion von einer Perzeptionssequenz abhängig. Zusätzlich kann eine jetzt getätigte Aktion alle nachfolgenden Entscheidungen beeinflussen. Dies ist beim Geometrischen Agenten der Fall. Die Entscheidung, an einer Kreuzung links oder rechts zu gehen, hat entscheidenen Einfluss auf die nachfolgenden Aktionen (und darauf, ob das Ziel erreicht wird) und sollte im Gedächtnis des Agenten behalten werden. Nur so kann er, wenn er sich auf dem falschen Weg wähnt, zu einer Kreuzung zurückkehren und dort eine neue Entscheidung treffen.

Diskret vs. **kontinuierlich**: Die Frage, ob eine Umgebung diskret oder kontinuierlich ist, kann auf verschieden Aspekte der Umgebung bezogen werden. Die Zustände und Zustandsübergänge des Geometrischen Agenten sind diskret und sind durch seine atomaren Aktionen bestimmt. Ebenso kann die Unterscheidung auf die Repräsentaton der Zeit angewendet werden. In der momentanen Version des *Geometrischen Agenten* ist keine kontinuierliche Repräsentation der Zeit vorgesehen. Die diskreten Zeitschritte entsprechen den Zustandsübergängen. Ein besonderes Rolle spielt die Repräsentation der geometrischen Aspekte der Umgebung. Hier kann eine etwas abweichende Unterscheidung getroffen werden: In der Umgebung selbst (in der Simulation) werden **quantitative** geometrische Aspekte repräsentiert. Der Geometrische Agent erzeugt sich ein Modell dieser Umgebung, das jedoch

rein auf **qualitativen** Konzepten basiert, da die Repräsentationssprache CRIL nur qualitative Konzepte bereitstellt. Dass bedeutet insbesondere, dass im Umgebungsmodell des Agenten quantitative Konzepte wie z.B. die Entfernung zwischen zwei Gebäuden oder der Winkel zwischen zwei Wegen nicht repräsentiert werden.

Einzelner Agent vs. **Multiagentensystem**: In Multiagentensystemen gibt es mehr als einen Agenten, und es können soziale Phänomen wie Konflikte (z.B. um Ressourcen) oder Kooperation auftreten. Dies ist im Projekt nicht von Interesse; der Geometrische Agent ist der einzige Agent im simulierten Campus.[9]

Fasst man die oben genannten Aspekte zusammen, lässt sich die Umgebung des Geometrischen Agenten wie folgt charakterisieren: Sie ist partiell beobachtbar, deterministisch, sequentiell, statisch, diskret und es gibt einen einzelnen Agenten. Dies ist nicht die ungünstigste Konstellation; Probleme wie Veränderung durch die Umwelt oder andere Agenten können ausgeschlossen werden. Die Aspekte partielle Beobachtbarkeit, Sequentalität und qualitative Repräsentation stellen besondere Vorgaben an den Entwurf des Aktionsmoduls. Statt vollständig über den Zustand der Welt informiert zu sein, verwendet der Agent Modelle, die unsicheres Wissen über die Welt repräsentieren. Da die Informationen in diesen Modellen hinsichtlich seiner Aufgabenstellung unvollständig sind, muss der Agent gewisse pragmatische Annahmen treffen, um dieses Problem zu umgehen, wie z.B. ein bestimmtes Wegstück aus einer Menge von Wegstücken auszuwählen. Die Sequentialität hat zur Folge, dass eine falsch getroffene Entscheidung des Agenten alle weiteren Entscheidungen betreffen kann (wenn er z.B. ‚falsch abgebogen' ist und sich auf einem falschen Weg befindet). Hier muss ein geeigneter Mechanismus den Fehler erkennen können und eine sinnvolle Korrektur bereitstellen (Dies könnte z.B. bedeuten, umzukehren, anstatt stur weiterzulaufen).

2.5 Agententypen

In [Russell und Norvig 2003] werden verschiedene Typen von Agentenmodellen vorgestellt, die für verschiedene Umgebungstypen prädestiniert sind und sich in der Komplexität der Agentenfunktion unterscheiden:

Der einfachste Agententyp ist ein **einfacher Reflex-Agent** (*simple reflex agent*)[10], bei dem einfache Reiz-Reaktions-Schemata (bzw. *condition-action-rules*) nur anhand des aktuellen Perzepts das Verhalten bestimmen. Einsetzbar ist dieser einfache Agententyp bei einer vollständig beobachtbaren Umgebung, da dem Agenten der vollständige Zustand der Welt bekannt ist und dieser Zustand direkt über das Reiz-Reaktions-Schema in eine Aktion überführt werden kann. Dieses Verfahren ist nicht für unsichere Umgebungen und somit auch nicht für den Geometrischen Agenten geeignet.

Der nächst komplexere Agententyp ist der des **modellbasierten Reflex-Agenten** (*model-based reflex agent*). Eine Möglichkeit, partielle Beobachtbarkeit zu handhaben, besteht in der Verwendung eines internen *Weltmodells*, welches Informationen beinhaltet, die über das aktuelle Perzept hinausgehen. Diese zusätzlichen Informationen könnten z.B. durch ein ‚Gedächtnis' des Agenten bereitgestellt werden, welches früher empfangene Perzepte bein-

[9] Da die Sprache CRIL auch als Agenten-Kommunikationssprache dienen kann, ist zu einem späteren Zeitpunkt eine Erweiterung hinsichtlich mehrerer Agenten, die untereinander Routenwissen austauschen, denkbar.

[10] Bei anderen Autoren (z.B. [Ferber 2001]) auch *reaktiver Agent* genannt.

haltet. Dieses Modell kann dann mit Hilfe von speziellem Wissen über die Domäne weiterverabeitet werden, so dass der Agent auch über Wissen verfügen kann, welches nicht direkt über die Perzeption gewonnen wird.

Einen internen Zustand und somit ausreichendes Wissen über den aktuellen Zustand der Umgebung zu besitzen, reicht oftmals nicht aus, um zu entscheiden, welche Aktion als nächstes zu tätigen ist. Ein **zielbasierter Agent** (*goal-based agent*) besitzt zusätzlich noch eine Repräsentation seiner Ziele, d.h. Zustände, deren Erreichen für den Agenten (oder den Designer des Programms) wünschenswert sind. Wenn ein Ziel direkt durch eine Aktion erreichbar ist, fällt die Auswahl der Aktion nicht schwer. Kann ein Ziel erst durch die Ausführung mehrerer Aktionen erreicht werden, ist es für den Agenten sinnvoll, diese Sequenz zu ermittlen und somit einen *Plan* von Aktionen zu bilden. Diese Sequenz von Aktionen zu erstellen, ist eine klassische Problemstellung in der KI-Forschung und wird *Planung* genannt. Das 4. Kapitel *Aktionsplanung und Ausführung unter Unsicherheit* widmet sich verschiedenen Planungsansätzen und -methoden und prüft, welche dieser Konzepte für den Geometrischen Agenten sinnvoll sind.

Ein weiterer Typ ist der **nutzenbasierte Agent** (*utility-based agent*), der nicht nur Ziele verfolgt, sondern auch eine Bewertung seiner Handlungen vornimmt. Sinnvoll ist eine Bewertung mithilfe einer Nutzenfunktion, wenn es mehrere konfligierende Ziele oder für die Erfüllung eines Ziels mehrere Pläne geben kann. Diese Nutzenfunktion bildet einen Zustand auf einen Zahlenwert, das Maß seines Nutzens, ab. Dieses Maß kann z.B. von den Kosten, die ein Plan bis zu diesem Zustand verursacht, abhängen. Dann könnte z.B. ein Plan mit geringeren Kosten einem Plan mit höheren Kosten vorzuziehen sein. Die Einbeziehung einer expliziten Nutzenfunktion ist für das in dieser Arbeit vorgestellte Verfahren des Aktionsmoduls z. Zt. nicht vorgesehen. Bei der Bestimmung des Performanzmaßes (s.u.) spielt jedoch die Bewertung der Aktionen hinsichtlich ihrer Kosten eine Rolle, so dass eine derartige Bewertung indirekt in die Entwicklung des Aktionsmoduls miteinfließt.

Zuletzt wird noch der **lernende Agent** (*learning agent*) vorgestellt. Dieser kann mit Hilfe eines Lernelements Bewertungen der aktuellen Situation und der ihm zur Verfügung stehenden Aktionen vornehmen und wenn es die Umstände erfordern, neue Aktionen kreieren oder erlernen. Eine derartige Funktionalität ist für den Geometrischen Agenten zur Zeit nicht geplant, ist aber durchaus denkbar (siehe Abschnitt 6.2 *Ausblick*).

Insgesamt kommt das Modell des zielbasierten Agenten dem Geometrischen Agenten am nächsten. Seine Aufgabe ist es, einen Weg von einer Startposition zu einem Zielpunkt zu finden. Somit ist sein Ziel klar formuliert und dem Agenten direkt vorgegeben. Desweiteren greift der Geometrische Agent auf ein Modell seiner Umgebung zurück, welches aus der Instruktion und der Perzeption generiert wird. Wie dieses Modell beschaffen ist und durch welche Prozesse es konstruiert wird, wird im Kapitel 3 *Verarbeitung der Routeninstruktion* erläutert.

2.6 Ein Performanzmaß für den Geometrischen Agenten

Um ein Rationalitätskriterium für den Geometrischen Agenten bestimmen zu können, muss neben der Berücksichtigung des a-priori Wissens, der ausführbaren Aktionen und der Perzeption des Agenten noch ein geeignetes Performanzmaß bestimmt werden.

Das Performanzmaß soll ein Urteil über den Erfolg oder Misserfolg des Agenten nach

einem Durchlauf der Simulation ermöglichen. Erfolgreich kann ein Durchlauf genannt werden, wenn der Agent sein vorgegebenes Ziel erreicht hat. Darüberhinaus sind weitere Kriterien wünschenswert, durch die erfolgreiche Abläufe miteinander verglichen werden können.

Ein Vergleich zwischen zwei erfolgreichen Durchläufen ist nur sinnvoll, wenn in beiden dieselbe Navigationsaufgabe gestellt wurde. Die Durchläufe können sich jedoch in der Routeninstruktion und in der Methode der Verarbeitung (in der Instruktionsphase) und anschließenden Nutzung der Instruktion (in der Navigationsphase) unterscheiden. Für einen Vergleich sind die beiden folgenden Szenarien interessant:

- Verschiedene Routeninstruktionen, gleiche Methode für die Verarbeitung / Nutzung (Fall 1). Hier kann geprüft werden, welche Routeninstruktion vom Geometrischen Agenten durch das vorgegebene Verfahren besser verwendet werden kann. Dabei kann insbesondere festgestellt werden, ob der Agent mit spezifischen Problemen (wie Lücken in der Routeninstruktion), die nur in bestimmten Instruktionen auftreten, umgehen kann.

- Dieselbe Routeninstruktion, verschiedene Methoden für die Verarbeitung / Nutzung (Fall 2). Hier können die Methoden zur Verarbeitung und Nutzung der Routeninstruktion direkt verglichen werden, da die äußeren Umstände (Navigationsaufgabe und Routeninstruktion) gleich sind. Allerdings ist im Kontext dieser Arbeit das Verfahren zur Verarbeitung der Routeninstruktion vorgegeben, so dass nur zwischen verschiedenen Verfahren zur Nutzung des Instruktionsmodells in der Navigationsphase unterschieden wird.

Folgende Kriterien für die Beurteilung der Performanz bieten sich an:

- Eine Beurteilung nach der **Anzahl der ausgeführten primitiven Aktionen**: Dies ist problematisch, da die Aktionen zum einen unterschiedlicher Art sind, und somit eine gleichwertige Bewertung (von z.B. Perzeptionsaktionen und Bewegungsaktionen) nicht sinnvoll ist. Zum anderen sind selbst Aktionen des gleichen Typs nicht direkt vergleichbar: So kann in einem Durchlauf eine Bewegung auf einem Routensegment (das aus mehreren anschließenden Wegstücken besteht) durch eine, in einem anderen Durchlauf durch zwei Bewegungsaktionen realisiert werden.

- Eine Beurteilung durch die **zurückgelegte Distanz**: Hat der Agent in einem Durchlauf eine kürzere Strecke in der simulierten Umgebung als in einem anderen Durchlauf zurückgelegt, ist dieser Durchlauf als performanter zu bewerten. Diese Vergleich ist jedoch nur bei gleicher Navigationsaufgabe sowie gleicher Routeninstruktion zulässig, da durch verschiedene Routeninstruktionen evtl. unterschiedliche Routen vorgegeben werden, die Unterschiede in der Distanz aufweisen können.

- Eine Beurteilung nach dem Aspekt, ob der Agent einem **Routensegment mehrmals gefolgt** ist, wobei die Länge aller mehrmals gefolgten Routensegmente aufsummiert wird. Das mehrmalige Folgen eines Routensegments kann unabsichtlich geschehen (z.B. wenn der Agent im Kreis geht), oder aber gezielt, wenn der Agent sich auf dem falschen Weg wähnt und ein Stück der Route zurückgeht, um von einem früheren Entscheidnungspunkt einen anderen Weg auszuwählen. In beiden Fällen ist aber ein Ablauf, in dem diese Wiederholungen auftreten als weniger performant zu bewerten.

Unter diesen Umständen ist für die beiden oben genannten Fälle die folgende informelle Definition für ein **Performanzkriterium** geeignet:

Findet der Agent sein Ziel nicht, war die Navigation nicht erfolgreich. Wurde das Ziel gefunden, bestimmt das Szenario das Performanzkriterium. Im Fall 1 wird eine Beurteilung nach der Länge aller mehrmals gefolgten Routensegmenten vorgenommen. Im Fall 2 bestimmt die insgesamt zurückgelegte Distanz das Performanzkriterium.

Kapitel 3

Verarbeitung der Routeninstruktion

Im Projekt *Geometrischer Agent* wird das Modell eines Agenten entwickelt, der mit Hilfe eines formalen Modells einer natürlichsprachlichen Routeninstruktion und der Fähigkeit zur Perzeption in einer virtuellen planaren Umgebung navigieren kann ([Tschander et al. 2003]). Das folgende Kapitel gibt einen Überblick über die Funktionsweise der Prozesse, die die Umwandlung der natürlichsprachlichen Routeninstruktion in das formale Modell vornehmen.

3.1 Routeninstruktionen

Wenn man in einer unbekannten Umgebung ein bestimmtes Ziel finden will, ist es sinnvoll, eine mit der Umgebung vertraute Person nach einer Wegbeschreibung bzw. *Routeninstruktion* zu fragen. Routeninstruktionen spezifizieren sowohl räumliche Informationen über die Umgebung der Route als auch temporale Informationen über die auszuführenden Aktionen ([Denis 1997]). Routeninstruktionen können auf verschiedene Arten mitgeteilt werden: als natürlichsprachliche Äußerung, als Zeichnung auf einem Blatt Papier, als eine Liste von auszuführenden Aktionen oder als Markierung auf einer Landkarte. Verschiedene Modalitäten können kombiniert werden, so kann z.B. eine sprachliche Wegbeschreibung durch Gestik oder durch Zeichnungen unterstützt werden. Das Projekt beschränkt sich auf spezifische, monomodale natürlichsprachliche Wegbeschreibungen, die im Voraus einer Person mitgeteilt werden (d.h. bevor sie sich in der entsprechenden Umgebung befindet).[1]

Für die Konstruktion der Routeninstruktion aktiviert der Instrukteur eine mentale Repräsentation der Umgebung, die sowohl den Start- als auch den Zielpunkt beinhaltet. Dann wird in dieser Repräsentation eine geeignete Route, die die beiden Punkte verbindet, ausgewählt. Diese Route wird in Routensegmente unterteilt, wobei diese Routensegmente mit Anweisungen verknüpft werden. Diese Anweisungen beschreiben Aktionen wie die Reorientierung des Agenten oder die Bewegung auf einem Pfad, der entlang des Routensegments verläuft. Zusätzlich muss eine Menge von Landmarken ausgewählt werden, die als Orientierungspunkte für den Instruierten dienen sollen. Schließlich wird eine verbale Beschreibung erzeugt und geäußert ([Wunderlich und Reinelt 1982]).

[1] Im Projekt werden aufgeschriebene sprachlichen Äußerungen behandelt. Hiermit wird von Problemen der Spracherkennung auf niedriger Ebene (Maschinelle Spracherkennung in Echtzeit etc.), die nicht im Fokus des Projekts liegen, abstrahiert.

(1)	(a)	Um von der Mensa zu Haus E zu gelangen,	(2)	(a)	Wenn du aus der Mensa kommst,
	(b)	hält man sich nach dem Verlassen der Mensa links		(b)	geh nach links,
	(c)	und geht auf die geschlossene Pforte zu.		(c)	zwischen Haus B und Haus C durch.
	(d)	Auf diesem Weg trifft man auf eine Abzweigung eines kleinen Weges nach rechts,		(d)	Geh hinter Haus C lang,
	(e)	der zwische Zaun und Haus entlangführt.		(e)	und dann, wenn du an Haus C vorbei bist,
	(f)	Dieser Weg mündet hinter dem Haus auf einem gepflasterten Platz,		(f)	wieder nach rechts.
	(g)	von dem man mit einer Treppe in Haus E gelangt.		(g)	Dann stehst du vor Haus E.

Tabelle 3.1: Zwei Beispiele für Routeninstruktionen, die dieselbe Route repräsentieren. (1) ist in deklarativer Art, (2) in imperativer Art gehalten.

Um die Routeninstruktion zu verstehen, generiert der Instruierte mit Hilfe von linguistischem Wissen eine Repräsentation der Bedeutung der Routeninstruktion. Diese Repräsentation hat typischerweise eine netzartige anstatt einer kartenartigen Struktur ([Werner et al. 2000]). Aus dieser Repräsentation werden die räumlichen Informationen extrahiert und etwaige Lücken in der Instruktion mithilfe pragmatischer Annahmen geschlossen. Da der Instruierte weder die Umgebung kennt noch sehen kann, ist die resultierende räumliche Repräsentation bezüglich quantitativer Aspekte wie dem Abstand zwischen zwei Landmarken oder dem Winkel zwischen zwei Wegstücken unterspezifiziert und somit *unsicher*.

Im Fokus des Projekts liegt die Verarbeitung und Verwendung der Routeninstruktion durch den Instruierten. Da die natürlichsprachliche Routeninstruktion und das resultierende Instruktionsmodell unterspezifiziert sind, muss der Agent die Routeninstruktion im Kontext der jeweils aktuellen Situation interpretieren.

Um für das Projekt beispielhafte Wegbeschreibungen zu erhalten, wurden von Personen, die mit den räumlichen Gegebenheiten des Informatikcampus Stellingen vertraut sind, Routeninstruktionen für zwei Aufgabenstellungen aufgeschrieben. Die erste Aufgabenstellung bestand in der Nennung einer Route von der Mensa zu Haus E, die zweite in einer Route vom Pförtnerhaus zu Haus E. Der Korpus besteht z.Zt. aus 13 geschriebenen Routeninstruktionen, die aus jeweils drei bis fünf Sätzen bestehen. Dabei beschreiben acht Instruktionen eine Route von der Mensa zu Haus E, fünf Instruktionen einen Weg vom Pförtnerhaus zu Haus E. Zwei Beispiele aus dem Korpus für die erste Aufgabenstellung werden in Tabelle 3.1 gegeben. Die hierdurch beschriebene Route wird in Abb. 3.1 dargestellt.

Zusätzlich zu den tatsächlich geäußerten Wegbeschreibungen werde ich für die Entwicklung des Aktionsmoduls auch von mir geschaffene Routeninstruktionen verwenden, die besondere Eigenschaften haben, d.h. besonders einfach zu handhaben sind oder spezifische Probleme

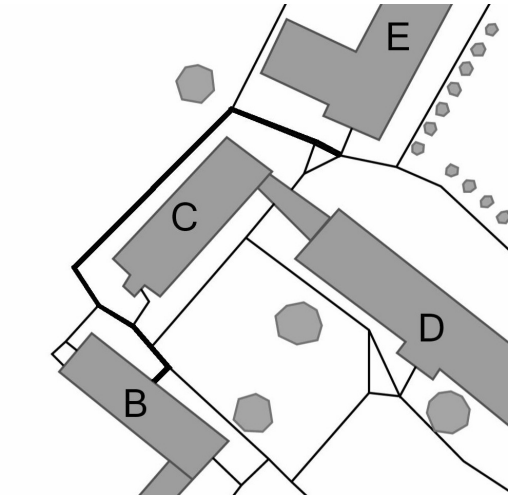


Abbildung 3.1: Die Route, die durch die Instruktionen (1) und (2) beschrieben wird.

aufweisen.

Die verbal geäußerte Wegbeschreibung wird während der Instruktionsphase des Geometrischen Agenten in ein internes Modell, das *Instruktionsmodell*, umgewandelt. Für eine adäquate Repräsentation der Instruktion (und auch der Perzeption und der Verknüpfung zwischen Instruktion und Perzeption) ist die Sprache CRIL (Conceptual Route Description Language) entwickelt worden. Diese Sprache leistet zum einen eine Spezifizierung der Semantik natürlichsprachlicher Ausdrücke in der Tradition der formalen Sematik. Zum anderen dient sie auch als interne Sprache des Geometrischen Agenten. Mit ihr lässt sich das Wissen des Geometrischen Agenten (sein interner Zustand) beschreiben, und sie ermöglicht formale Inferenz über dieses Wissen. Zudem werden in CRIL die Aktionen des Agenten auf abstrakter Ebene spezifiziert.

3.2 Die Conceptual Route Description Language CRIL

Allgemeine Aufgabe von Repräsentationssprachen ist die explizite Darstellung von Weltmodellen ([Habel 1986]). Durch das Projekt *Geometrischer Agent* soll die Entwicklung eines Formalismus unterstützt werden, der die Spezifikation einer Repräsentationssprache erlaubt, durch die Modelle natürlichsprachlicher Routeninstruktionen gebildet werden können. Diese Beschreibungssprache heisst *Conceptual Route Description Language* (CRIL). Da in der Navigationsphase eine hohe Interaktion zwischen der Repräsentation der Instruktion und der Repräsentation der Perzeption stattfindet, ist es sinnvoll, auch die Repräsentation der Perzeption in CRIL zu formulieren. Somit ist es möglich, eine gemeinsame Repräsentation zu verwenden, in der beide Repräsentationen vereint sind, und die während der Navigationsphase durch weitere Informationen angereichert wird (Siehe hierzu Abschnitt 3.4 *Die Navigationsphase*).

Da mentale räumliche Repräsentationen von sprachlichen Wegbeschreibungen eher netzartigen als kartenartigen Charakter haben ([Werner et al. 2000]), ist es sinnvoll, für das Instruktions- und das Perzeptionsmodell eine netzartige Struktur zu wählen. Diese Struktur

wird durch sogenannte CRIL-Netze bereitgestellt, welche eine Weiterentwicklung sogenannter *Referentieller Netze* darstellen.

3.2.1 Referentielle Netze

Referentielle Netze bieten die Möglichkeit, *Objektwissen* zu repräsentieren, indem sie Wissen über spezifische Objekte des Diskurses (hier der sprachlichen Wegbeschreibung) strukturieren ([Habel 1986], [Eschenbach 1988]). Diese Objekte werden durch *Referenzobjekte* (RefOs) repräsentiert und stellen im formalen Modell des Weltausschnitts die relevanten Objekte des Diskurses dar. Dies sind insbesondere die explizit genannten Objekte der sprachlichen Äußerung (bzw. im Perzeptionsmodell die perzipierten Objekte). Die Struktur des Netzes wird nicht wie z.B. bei semantischen Netzen durch vordefinierte Relationen erreicht, vielmehr wird eine Strukturierung durch Angabe von *Designatoren* und *Attributen* geschaffen. Dadurch werden den Referenzobjekten bestimmte Eigenschaften zugeordnet oder sie werden zu anderen Objekten in Beziehung gesetzt.

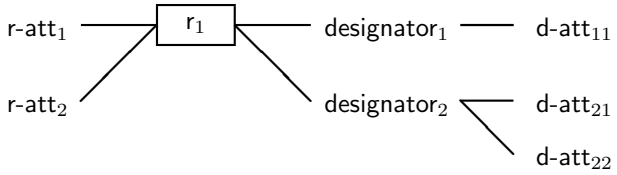


Abbildung 3.2: Ein referentielles Netz (Beispiel)

Referentielle Netze beinhalten Referenzobjekte zusammen mit ihren Designatoren und Attributen (Siehe Abb. 3.2). Formal ist ein referentielles Netz eine Teilmenge aus

(R-ATT x REFO) x DESIGN x D-ATT.

Dabei bezeichnet REFO die Menge der Referenzobjekte und DESIGN die Menge der Designatoren. R-ATT und D-ATT sind die Mengen der RefO- resp. Designatorattribute.

Designatoren dienen zur Bezeichnung eines Referenzobjekts. Eine derartige Bezeichnung kann auf zwei Arten geschehen. Zum einen kann ein Referenzobjekt durch einen (oder mehrere) Namen bezeichnet werden (z.B. ‚Hans' oder ‚Haus A'). Dies ist die Klasse der *Eigennamen*. Zum anderen kann ein Referenzobjekt durch sogenannte *Deskriptionen* bezeichnet werden. Deskriptionen sind komplexe geschlossen Terme und stellen durch die Bindung der in ihnen auftauchenden Variablen die Beziehung zu anderen Referenzobjekten her (z.B. ηx: Ehemann(r_3, x) - ein x, welches mit der Person, die durch das Referenzobjekt r_3 repräsentiert wird, verheiratet ist). RefO-Attribute beschreiben die Beschaffenheit von Referenzobjekten, so wie den Typ (z.B. Haus) oder andere Eigenschaften (z.B. Farbe). Designatorattribute dienen zur Spezifizierung der Designatoren und ermöglichen z.B. die Einschätzung der Qualität des entsprechenden Wissens (durch einen Konfidenzwert).

Die Typen (auch *Sorten* genannt) von Referenzobjekten wie auch andere Operatoren (Prädikate und Relationen) werden durch eine *Taxonomie* strukturiert. Taxonomien stellen eine Vererbungshierarchie zur Verfügung, d.h. mit ihnen kann man bestimmen, ob eine Sorte von einer anderen Sorte subsumiert wird, oder ob zwei Typen sich gegenseitig ausschließen. Die Sorten ‚Mann' und ‚Frau' werden z.B. von der Sorte ‚Mensch' subsumiert (und erben somit alle Eigenschaften der Sorte ‚Mensch') schließen sich jedoch gegenseitig aus. Diese Strukturierung spielt eine wichtige Rolle bei der Auflösung von Koreferenzen (sowohl bei der Sprachverarbeitung wie während der Navigation), da durch sie ermittelt werden kann, wie ähnlich sich zwei Objekte bezüglich ihres Typs sind.

Zusätzlich zum Objektwissen besitzt ein sprachverarbeitendes System noch *regelhaftes* und *faktuelles* Wissen. Regelhaftes Wissen betrifft regelmäßige, gesetzmäßige, systematische Beziehungen zwischen Zuständen, Ereignissen, Handlungen und Situationen der Welt ([Habel 1986]). Das regelhafte Wissen ermöglicht mittels Inferenzprozesse die Transformation bestehender referentieller Netze. Im Geometrischen Agenten ist diese regelhafte Wissen geometrischer Natur und im GCS gespeichert. Es dient zum Aufbau und zur Weiterverarbeitung der CRIL-Netze. Faktuelles Wissen betrifft Zustände, Ereignisse, Situationen und Handlungen. Das Wissen, das (vom Agenten auszuführende) Handlungen betrifft, wird im Geometrischen Agenten im Aktionsplan zusammengefasst. Diese beiden Aspekte des Wissens werden in Abschnitt 3.3 behandelt.

3.2.2 CRIL-Netze

Die Sprache CRIL stellt Operatoren zur Bildung von Prädikaten und Relationen zur Verfügung, mit deren Hilfe Referenzobjekte spezifiziert werden. Das Modell wird durch eine Menge von Prädikaten und Relationen aufgebaut, die implizit konjunktiv verknüpft sind, und die Attribute von Objekten (Prädikate), bzw. Beziehungen zwischen Objekten (Relationen) beschreiben. Einen Überblick über verschiedene Operatoren gibt die Tabelle 3.3, ein Beispiel für ein Modell die Tabelle 3.4.

In CRIL-Netzen wird die Beziehung von Referenzobjekten durch Kanten visualisiert. Sortenattribute werden durch die Bezeichnung des Referenzobjekts deutlich gemacht (z.B. werden Regionen durch ein ‚r' bezeichnet). Weitere Attribute und Eigennamen werden in der Visualisierung eines Referenzobjekts im CRIL-Netz mit angegeben. Diese beiden Modifikationen tragen zu einer strukturellen Vereinfachung der Darstellung der Netze bei, was besonders bei komplexen Netzen, die schon aus einigen Arbeitsschritten der Verarbeitung in der Simulation resultieren, eine bessere Lesbarkeit der Netze ermöglicht.

Tabelle 3.2 zeigt, wie die räumliche Information eines Satzes als referentielles Netz oder als CRIL-Netz repräsentiert werden kann. Der repräsentierte Satz ist der erste Teilsatz aus der zweiten beispielhaften Wegbeschreibung (2)(a): „Wenn Du aus der Mensa kommst,...". Die Sorte ‚landmark' wird in der CRIL-Darstellung nicht mehr explizit genannt, da die Sorte ‚building' von der Sorte ‚landmark' subsumiert wird.

Der Generierung von CRIL-Netzen aus der sprachlichen Repräsentation und der Weiterverarbeitung der Netze widmet sich der nächste Abschnitt.

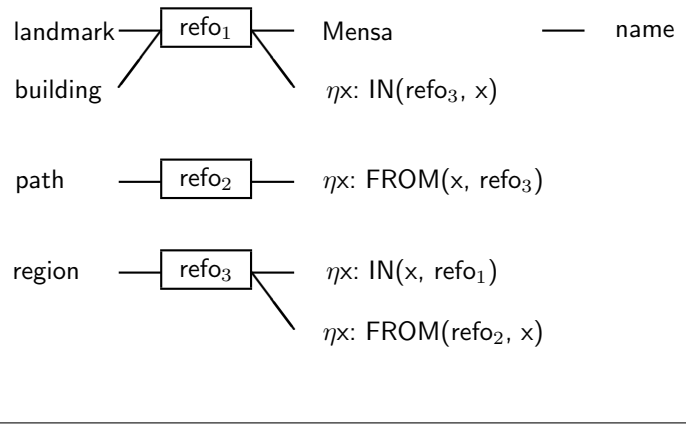
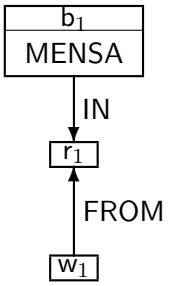
Referentielles Netz	**CRIL-Netz**
	

Tabelle 3.2: Visualisierung der Repräsentation der räumlichen Information des Beispielsatzes (2)(a) „Wenn Du aus der Mensa kommst,...“ als referentielles Netz und als CRIL-Netz.

3.3 Die Instruktionsphase

Da die Wegbeschreibung dem Agenten im voraus mitgeteilt wird, kann zwischen zwei zeitlich getrennten Prozessen, der *Instruktionsphase* und der *Navigationsphase*, unterschieden werden. In der Instruktionsphase wird dem Geometrischen Agenten die verbale Routeninstruktion präsentiert, aus der er dann mittels zweier Prozesse (*Syntaktische und semantische Verarbeitung* und *Verarbeitung der Instruktion*) ein internes Instruktionsmodell erzeugt, welches die räumliche Information der Instruktion in einem räumlichen Modell und die imperativen Anteile in einem Aktionsplan repräsentiert (Siehe Abb. 3.3). Die Navigationsphase simuliert die Navigation des Agenten vom Start- zum Zielpunkt. Während der Navigation führt der Agent Aktionen aus, die das Aktionsmodul unter Zuhilfenahme des internen Instruktionsmodells und der Perzeption während der Navigation auswählt. Instruktionsmodell und Perzeptionsmodell werden durch CRIL-Netze spezifiziert.

Das Instruktionsmodell beinhaltet den Aktionsplan, welcher die vom Agenten auszuführenden Handlungen beschreibt. An der Generierung des Aktionsplans sind zwei Prozesse beteiligt: Den ersten Schritt leistet der Instruktor, der durch die natürlichsprachliche Formulierung der Routeninstruktion implizit einen Plan von auszuführenden Handlungen aufstellt. Diese imperativen Anteile der Routeninstruktion werden dann im Geometrischen Agenten während der Verabeitung der Routeninstruktion in ein formales Modell der zu tätigenden Handlungen übersetzt: in den Aktionsplan. Da der Aktionsplan Aktionen (oder besser: *Anweisunegn*) beinhaltet, die aufgrund des Problems der Unterspezifiziertheit nicht direkt ausführbar sind, müssen diese Anweisungen während der Navigationsphase in einem lokalen Planungsprozess auf primitive ausführbare Aktionen des Agenten abgebildet werden.

Da der Aktionsplan und das räumliche Instruktionsmodell die wesentlichen Informationsressourcen für den Prozess der lokalen Planung darstellen, werde ich die Prozesse der Instruktionsphase und deren Ergebnisse im folgenden kurz vorstellen. Eine ausführliche Beschreibung

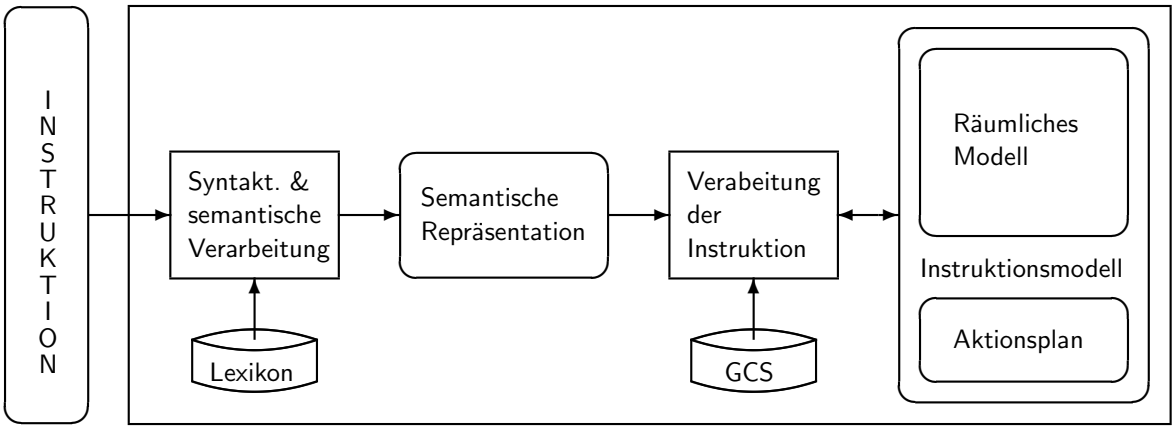


Abbildung 3.3: Die Instruktionsphase des Geometrischen Agenten.

wird in [Tschander et al. 2003] gegeben.

3.3.1 Syntaktische und semantische Verarbeitung

Im ersten Prozess der syntaktischen und semantischen Verarbeitung wird linguistisches Wissen aus einem räumlichen Lexikon verwendet, um die natürlichsprachliche Repräsentation der Wegbeschreibung in eine semantische Repräsentation zu übertragen. Dies geschieht durch die Verwendung von Lexikoneinträgen, die den semantischen Gehalt von Ausdrücken in Abhängigkeit der syntaktischen Struktur wiedergeben.

Wegbeschreibungen spezifizieren im allgemeinen Aktionen, Pfade, Wege, Positionen und Landmarken. Jede dieser Komponente wird dabei von einem bestimmten Typ von Wörtern charakterisiert. Aktionen werden z.B. meistens durch Bewegungsverben beschrieben.[2] Die semantische und syntaktische Verarbeitung erzeugt dabei für Wörter eines bestimmten Typs einen entsprechenden Ausdruck. So werden beispielsweise Bewegungsverben in CRIL durch Ausdrücke repräsentiert, die mit Hilfe des Operators $\mathsf{GO}(x, w)$ gebildet werden (dies entspricht der Anweisung an den Agenten x, sich auf dem Pfad w zu bewegen). Einen Überblick über die Operatoren gibt die Tabelle 3.3.

Die resultierende semantische Repräsentation beinhaltet sowohl räumliche Informationen, wie auch Informationen über die vom Agenten auszuführenden Aktionen. In der Tabelle 3.4 werden die drei Typen von Informationen, die aus einer Wegbeschreibung extrahiert werden, dargestellt: die Aktionen, die räumlichen Relationen und die Landmarken. Die räumlichen Relationen und Landmarken bilden die Basis für eine Verfeinerung des Netzes im Prozess der Verabeitung der Instruktion (siehe nächsten Abschnitt). Sie können auch in einem CRIL-Netz dargestellt werden (Siehe Tabelle 3.5).

3.3.2 Verarbeitung der Instruktion

Der zweite Prozess verarbeitet mit Hilfe des GCS und weiterer Inferenzmechanismen, die auf pragmatischen Annahmen beruhen, die semantische Repräsentation weiter zu einem Instruktionsmodell. Bei der Weiterverarbeitung werden die räumlichen Informationen von den

[2] Abweichungen hiervon können unter anderem zu sog. *Lücken* im Aktionsplan führen (s.u.).

Typ des natürlichsprachlichen Ausdrucks	**Charakteristische semantische Komponente**
Positionsverben	BE_AT(x,p)
Bewegungsverben	GO(x,w)
Verben des Orientierungswechsels	CH_ORIENT(x,d)
lokale Präpositionen oder Adverben	LOC(u,PREP(l))
direktionale Präpositionen oder Adverben	TO(w,PREP(l))
	FROM(w,PREP(l))
	VIA(w,PREP(l))
	LOC(w,PREP(l))
projektive Terme	PREP(l,rsys)

Tabelle 3.3: Deskriptive Operatoren, die in lexikalischen Einträgen von Verben und Präpostionen verwendet werden.

imperativen Informationen getrennt, so dass im Instruktionsmodell zwei in CRIL formulierte Teilbereiche resultieren: zum einen das interne räumliche Modell, und zum anderen der Aktionsplan.

3.3.3 Räumliche Anteile des Instruktionsmodells: der Instruktionsgraph

Der Instruktionsgraph repräsentiert die räumliche Information der Routeninstruktion. Er beinhaltet Referenzobjekte sowie Prädikate und Relationen, die diese Objekte spezifizieren. Die Referenzobjekte werden als Knoten im CRIL-Netz visualisert, Relationen als Kanten zwischen den Knoten. Objekttypen sind Landmarken (l), Pfade (w), Wege (t, da engl. *tracks*), Regionen (r) und Positionen (p). Die Landmarken entsprechen gut erkennbaren Objekten, die dem Navigator die Orientierung während der Navigationsphase ermöglichen sollen. Pfade sind Trajektorien bewegter Objekte und besitzen eine Richtung. Sie werden in sprachlichen Beschreibungen durch Bewegungsverben, direktionale Präpositionalphrasen oder direktionale Adverbien eingeführt, also Ausdrücken, die zwischen einem Start- und einem Zielpunkt unterscheiden. Pfade sind in einer statischen Situation nicht erkennbar, im Gegensatz zu Wegen (zu denen auch Straßen zählen), die als solche in der Umgebung deutlich sichtbar sind. Regionen sind geometrische Bereiche, die durch die Funktionen von Relationen zu Landmarken oder Referenzsystemen spezifiziert werden. So wird z.B. durch den Ausdruck $\mathsf{FROM}(w_1,\mathsf{IN}(l_1))$ der Pfad w_1 mit der Region r_1 in Beziehung gesetzt. Dabei bildet die Funktion IN die Landmarke l_1 auf die Region r_1, die in ihr liegt, ab. Zusätzlich gibt es CRIL-Knoten, die ein Referenzsystem (rsys) repräsentieren, welches eine projektive Relation mit dem Kontext in Beziehung setzt.

Das initiale CRIL-Netz (Tabelle 3.5) ist eine direkte Umsetzung der propositionalen Spezifikation der Satzteile (2a) und (2b) aus Tabelle 3.4 in die netzartige Darstellung. Die Pfade w_1 und w_2 können mit den Regionen r_1 und r_2 über die Relationen FROM und TO verbunden werden, die Regionen sind über Landmarken (l_1) bzw. Referenzsysteme spezifiziert ($rsys_2$).

Das GCS (Geometric Concept Specification) beinhaltet axiomatische Charakterisierungen räumlicher Konzepte, wie TO, FROM und VIA, wie sie in [Eschenbach et al. 2000] vorgestellt werden. Sie spezifizieren, wie sich der Startpunkt stpt(w) oder Zielpunkt fpt(w) eines Pfades

(2)		**Aktionen**	**räumliche Relationen**	**Landmarken**
(a)	Wenn Du aus der Mensa kommst,		FROM(w_1, IN(l_1))	MENSA(l_1)
(b)	geh nach links,	!GO(w_2)	TO(w_2, LEFT($rsys_2$))	
(c)	zwischen Haus B und Haus C durch.	!GO(w_3)	VIA(w_3,BETWEEN(l_2, l_3))	HOUSE(l_2) NAME(l_2,'B') HOUSE(l_3) NAME(l_3,'C')
(d)	Geh hinter Haus C lang,	!GO(w_4)	LOC(w_4,BEHIND(l_3, $rsys_4$)) ALONG(w_4, l_3)	HOUSE(l_3) NAME(l_3,'C')
(e)	und dann, wenn du an Haus C vorbei bist,	!BE_AT(p_1)	LOC(p_1, PAST(l_3, $rsys_5$))	HOUSE(l_3) NAME(l_3,'C')
(f)	wieder nach rechts.	!GO(w_6)	TO(w_6),RIGHT($rsys_6$)	
(g)	Dann stehst du vor Haus E.	!BE_AT(p_2)	LOC(p_2, FRONT(l_4,$rsys_7$))	HOUSE(l_4) NAME(l_4,'E')

Tabelle 3.4: CRIL-Repräsentation des Beispiels (2) aus Tabelle 3.1: Resultat der semantischen Analyse.

zu einer Region verhält, je nachdem ob der Pfad über TO, FROM oder VIA mit der Region in Beziehung steht. Derartige Definitionen können als Termersetzungsregeln dazu verwendet werden, um eine Transformation des CRIL-Netzes durchzuführen. Abb. 3.6 zeigt, wie durch die Verwendung der Regeln das CRIL-Netz transformiert wurde.

Das GCS wird nicht nur während der Instruktionsphase im Prozess der Verarbeitung der Instruktion zur Verfeinerung des CRIL-Netzes verwendet; auch in der Navigationsphase finden Prozesse statt, die auf die geometrischen Konzeptionen des GCS zurückgreifen. So wird beispielsweise die Spezifikation der Funktion BETWEEN dazu verwendet, um zu prüfen, ob sich ein perzipiertes Wegstück zwischen zwei wahrgenommenen Häusern befindet.

Das so verfeinerte CRIL-Netz kann noch mittels pragmatischer Annahmen weiterverabeitet werden. Eine pragmatische Annahme ist beispielsweise, dass der Zielpunkt eines Pfades

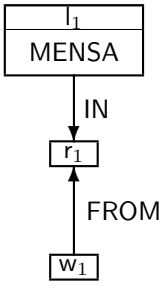
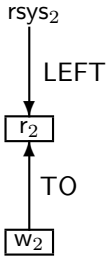
(2)	(a)	(b)
Räumliche Relation	FROM(w_1, IN(l_1))	TO(w_2, LEFT($rsys_2$))
Landmarken	MENSA(l_1)	
CRIL-Netz		

Tabelle 3.5: Räumliche Repräsentation der Beispielsätze (2a) und (2b) als CRIL-Netz.

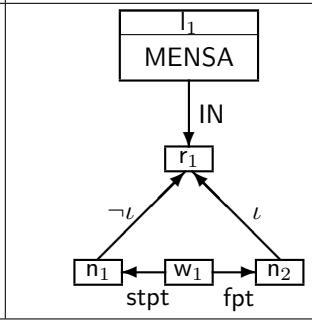
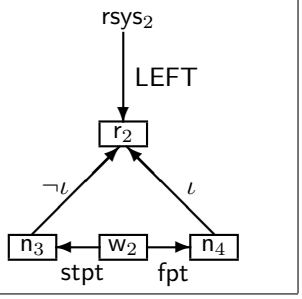
(2)	(a)	(b)
Räumliche Relation	FROM(w_1, IN(l_1))	TO(w_2, LEFT($rsys_2$))
Landmarken	MENSA(l_1)	
CRIL-Netz		

Tabelle 3.6: Repräsentation als CRIL-Netz des Beispiels (2): Resultat der Verarbeitung mit dem GCS.

dem Startpunkt des nächsten Pfades entspricht. Auf diese Weise können zwei Teilnetze miteinander verknüpft werden, indem Knoten in ihnen als identisch identifiziert werden (im Beispiel sind dies die Knoten n_2 und n_3). Zusätzlich kann im Beispiel die Annahme getroffen werden, dass das Referenzsystem $rsys_2$, welches durch die Spezifikation von w_2 eingeführt wird, mit der Richtung des vorangegangenen Pfades gleichzusetzen ist. Das aus der Inferenz resultierende CRIL-Netz wird in Abbildung 3.4 dargestellt.

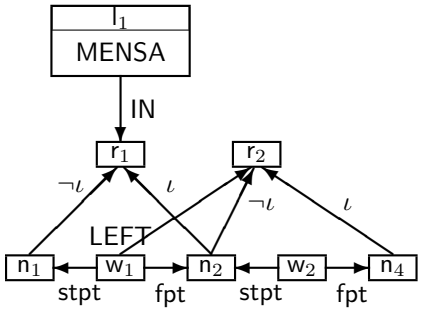


Abbildung 3.4: Repräsentation als CRIL-Netz des Beispiels (2): Resultat nach Inferenz (Pragmatische Verarbeitung).

3.3.4 Imperative Anteile des Instruktionsmodells: der Aktionsplan

Während der Verarbeitung der Instruktion wird das räumliche Modell der Routeninstruktion als ein CRIL-Netz aufgebaut. Zusätzlich werden die imperativen Anteile aus der Routeninstruktion extrahiert und in einem Aktionsplan zusammengefasst. Der Aktionsplan beinhaltet eine Liste von imperativen Anweisungen, die im folgenden *Instruktionsanweisungen* genannt werden. Die Instruktionsanweisungen werden während der Navigationsphase in Abhängigkeit der aktuellen Situation interpretiert. Durch einen dynamischen Prozess der lokalen Planung werden durch das Aktionsmodul primitive Aktionen ausgewählt, die der Agent in der simulierten Umgebung ausführen kann.

Die imperative Anweisungen ähneln in der Form ihren deskriptiven Gegenstücken GO(x,w), BE_AT(x,p) und CH_ORIENT(x, d): !GO(w) bedeutet ‚bewege dich entlang Pfad w', !BE_AT(p) ‚versichere Dich, an Position p zu sein, andernfalls gehe zu Position p' und !CH_ORIENT(d) ‚drehe dich in Richtung d'.[3]

Aktionen und Pläne liegen im Geometrischen Agenten in zwei Stufen vor: Als imperative Anweisungen im Aktionsplan des Instruktionsmodells sind sie wegen des Problems der Unterspezifiziertheit nicht direkt ausführbar, sondern dienen dem Agenten als Vorgabe und haben daher eher deklarativen als prozeduralen Charakter. Dies entspricht der *plan-as-communication-view* (siehe Abschnitt 4.4.4), in der Pläne auf höherer Stufe nicht als strikt auszuführende Aktionssequenzen, sondern als Hilfsmittel für einen Planungsprozess auf niedrigerer Ebene angesehen werden. Der Aktionsplan wird während der Instruktionsphase gefüllt, indem zu jedem deskriptiven Operator, der einem Bewegungsverb entspricht, eine imperative Anweisung erzeugt und im Aktionsplan gespeichert wird.

Auf der untersten Stufe der Simulation führt der Agent primitive Aktionen in der virtuellen Umgebung aus. Zu diesen Aktionen zählen sowohl Bewegungsaktionen als auch Perzeptionsaktionen. Da durch den lokalen Planungsprozess eine Anweisung des Aktionsplans auf mehrere Aktionen auf der niedrigen Stufe abgebildet wird, muss auf niedriger Stufe für jede Anweisung ein Plan von primitiven Aktionen erzeugt werden. Diese *Lokale Aktionssequenz* (*local action sequence*) zu bestimmen, ist eine der zentralen Aufgaben des Aktionsmoduls.

Lücken im Aktionsplan

Neben dem Problem der Unterspezifiziertheit der Aktionen im Aktionsplan kann ein weiterer Aspekt der Unsicherheit auftreten, der zu sogenannte *Lücken* im Aktionsplan führt. Eine Lücke tritt auf, wenn eine vom Instruktor zu tätigende Handlung in der natürlichsprachlichen Routeninstruktion durch einen Ausdruck beschrieben wird, der durch den Prozess der Verabeitung der Instruktion nicht als imperative Anweisung interpretiert wird. Manche Aktionen werden deklarativ formuliert. Im Beispiel (1) (siehe Abschnitt 3.1 *Routeninstruktionen*) wird in den Satzteilen (d) bis (f) eine Handlung beschrieben, einem kleinen Weg zu folgen:

(1) (d) Auf diesem Weg trifft man auf eine Abzweigung eines kleinen Weges nach rechts,
(e) der zwische Zaun und Haus entlangführt.
(f) Dieser Weg mündet hinter dem Haus auf einem gepflasterten Platz, ...

Eine derartige Formulierung führt zu einer Lücke im Aktionsplan, da bei der Verarbeitung der Routeninstruktion entsprechende Instruktionsanweisungen nur für Handlungen generiert werden, die durch imperative Ausdrücke (d.h. Bewegungsverben) beschrieben werden.

3.4 Verwendung des Instruktionsmodells in der Navigationsphase

In der Navigationsphase wird die Navigation des Geometrischen Agenten in dem Modell des Campus Stellingen simuliert. Dabei nutzt der Geometrische Agent das Wissen aus der Routen-

[3] Das erste Argument in der deskriptiven Darstellung, welches den Agenten beschreibt, wird hier weggelassen.

instruktion, um geeignete Aktionen auszuwählen und auszuführen. Um angemessen handeln zu können, ist der Geometrische Agent aufgrund des Problems der Unsicherheit auf Hilfsprozesse angewiesen, die Informationen über den aktuellen Zustand des Agenten und seiner Umgebung bereitstellen. Dies leisten die Prozesse *Verarbeitung der Perzeption*, *Koreferenzauflösung* und *Selbstlokalisation*, die aktuelle Informationen während der Navigation in das interne *Umgebungsmodell* des Agenten integrieren. Hierdurch kann der Agent die Informationen aus der Routeninstruktion angemessen im Kontext der aktuellen Situation interpretieren.

Der Agent interagiert mit seiner Umwelt über primitive Aktionen. Aktionen sind *primitiv*, wenn sie direkt ausgeführt werden können. Diese Aktionen sind entweder Perzeptions- oder Bewegungsaktionen (siehe Abschnitt 5.2 *Die primitiven Aktionen des Agenten*). Für die Auswahl dieser Aktionen stellt das Aktionsmodul den Prozess der *Lokalen Planung* bereit. Dieser Prozess orientiert sich zu einen am Aktionsplan, der die in der Routeninstruktion beschriebenen Hadlungen als abstrakte Anweisungen repräsentiert, und zum anderen am Umgebungsmodell, welches durch zusätzliche Informationen die Anweisungen des Aktionsplans näher spezifiziert, so dass sie auf eine Sequenz von primitiven Aktionen, die lokale Aktionssequenz (LAS), abgebildet werden können. Zusätzlich zur Auswahl der primitiven Aktionen koordiniert das Aktionsmodul die oben genannten Hilfsprozesse ‚Verabeitung der Perzeption', ‚Koreferenzauflösung' und ‚Selbstlokalisation', da während des lokalen Planungsprozesses die Ergebnisse dieser Prozesse verwendet werden.

Für die Prozesse der Perzeption, die Verarbeitung der Perzepion und die Auflösung von Koreferenzen liegt eine prototypische Implementation vor. Die Prinzipen und Funktionsweisen dieser Prozesse werden in [Helwich 2003] ausführlich vorgestellt, hier soll nur ein kurzer Überblick gegeben werden.

3.5 Prozesse in der Navigationsphase

3.5.1 Verarbeitung der Perzeption

Um über den aktuellen Zustand seiner Umgebung informiert zu sein, muss der Geometrische Agent seine Umgebung wahrnehmen. Hierzu verfügt er über eine Perzeptionsaktion sowie den anschließenden Prozess ‚Verarbeitung der Perzeption'.

Wie in der Tabelle 3.7 schematisch dargestellt, nimmt der Agent durch die Perzeptionsaktion den Teil der Umgebung wahr, der sich in seinem Sichtfeld befindet. Durch die Verarbeitung der Perzeption wird das Perzeptionsmodell erzeugt bzw. aktualisiert, d.h. es werden die Objekte, die der Agent perzipiert hat, mit deren geometrischen Relationen in den Perzeptionsgraphen eingefügt. Der Perzeptionsgraph wird als CRIL-Graph repräsentiert und wird im folgenden abkürzend P-Graph oder P-Netz genannt.

3.5.2 Auflösung von Koreferenzen

Um die Informationen aus dem Instruktionsmodell für die Navigation sinnvoll nutzen zu können, muss der Agent während der Navigation die sprachlich beschriebenen Objekte der Wegbeschreibung in der visuell perzipierten Ungebung wiedererkennen können, d.h. feststellen, welche Objekte koreferent sind. Dies wird durch den Prozess der Koreferenzauflösung geleistet. Dieser Prozess schließt die Lücke zwischen der Perzeption und der Instruktion, da

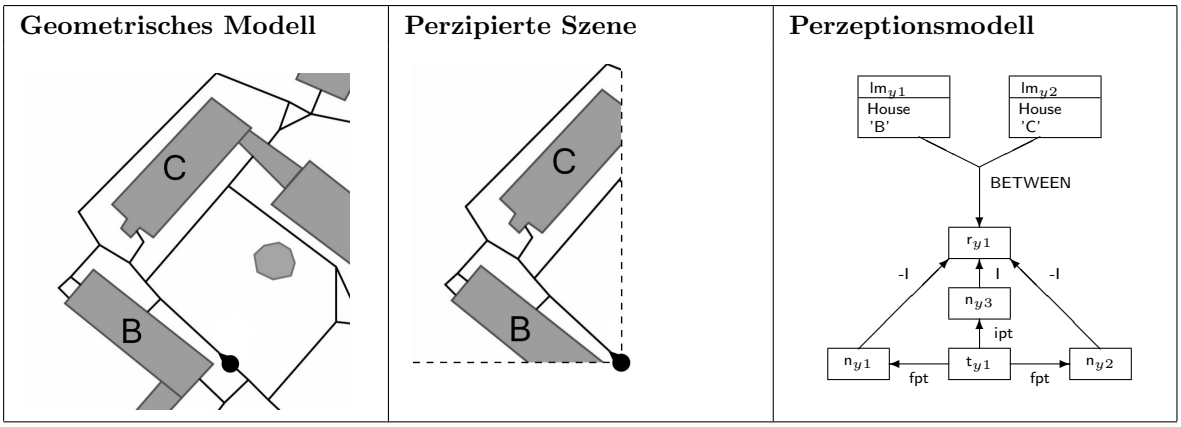


Tabelle 3.7: Der Agent nimmt mittels einer Perzeptionsaktion den sichtbaren Ausschnitt seiner Umgebung wahr. Der Prozess ‚Verarbeitung der Perzeption' aktualisiert das Perzeptionsmodell, indem er für die perzipierten Objekte und deren Relationen ein P-Netz erzeugt und dieses in das Perzeptionsmodell einfügt (hier wird nur ein beispielhaftes Teilnetz dargestellt).

er die zwei internen räumlichen Repräsentationen des Agenten, das Instruktionsmodell und das Perzeptionsmodell, miteinander verknüpft, indem er Objekte des I-Graphen als Objekte des P-Graphen identifiziert.

Die Auflösung von Koreferenzbeziehungen geschieht über die Bestimmung von Ähnlichkeitsmaßen, die zwischen Teilnetzen aus dem I-Graphen und dem P-Graphen bestehen. Dabei werden verschiedene Teilnetze des P-Graphen mit dem als relevant bewerteten Teilnetz des I-Graphen verglichen ([Helwich 2003]). Diese Teilnetze stellen jeweils einen Ausschnitt des gesamten Graphen dar und beinhalten das zu prüfende Objekt, sowie weitere Objekte, die in einer relevanten Beziehung zu diesem Objekt stehen. Die Ähnlichkeit zwischen zwei Teilnetzen wird über die Ähnlichkeit der Attribute der zu prüfenden Objekte und die Ähnlichkeit der in Relation stehenden Objekte ermittelt und mit einem Ähnlichkeitswert[4] versehen. Diese Herangehensweise hat zur Folge, dass alle Objekte eines Teilnetzes, das mit einem anderen Teilnetz koreferenziert wurde, denselben Koreferenzwert besitzen. Wird ein festgesetzter Schwellwert durch das Ähnlichkeitsmaß zweier Teilnetze überschritten, werden die Teilnetzte als koreferent gedeutet. Dieser Schwellwert stellt einen wesentlichen Parameter dar, durch den das Verhalten des Agenten gesteuert wird, da er bestimmt, welche Objekte (und insbesondere welche Pfade) vom Agenten als koreferent interpretiert werden.

Das Umgebungsmodell wird durch Relationen, die die Koreferenzbeziehungen repräsentieren, angereichert. Koreferenzen können in CRIL-Graphen durch *Koreferenzkanten* visualisiert werden (siehe Abb. 3.5).

Eine Koreferenzbeziehung kann aus mehreren Gründen nicht als sicher angesehen werden. So kann es z.B. mehrere Objekte aus der Perzeption geben, für die ein vergleichbar hoher Koreferenzwert ermittelt wurde. Ist dies der Fall, können mehrere Objekte für die Koreferenzauflösung in Frage kommen, und der Prozess der Koreferenzierung liefert als Ergebnis eine Liste von koreferenzierbaren Objekten, aus der mittels pragmatischer Annahmen noch ein Objekt ausgewählt werden muss. Ein relativ sicheres Erkennen liegt im Modell jedoch bei der

[4] auch *Koreferenzwert* genannt.

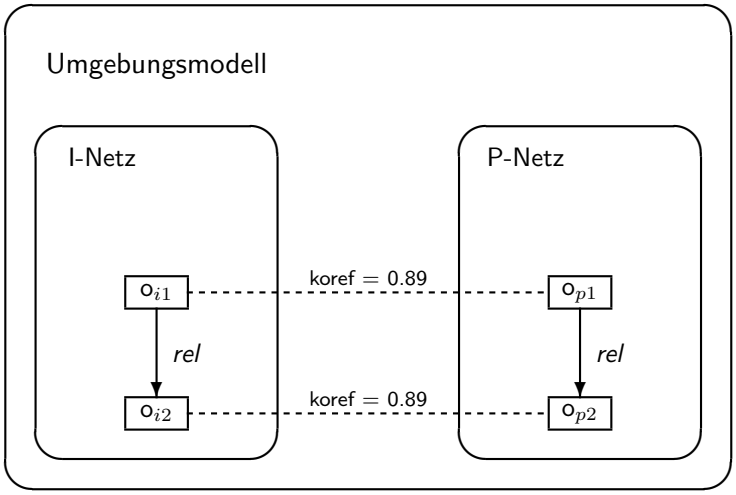


Abbildung 3.5: Koreferenzen (gestrichelte Kanten) zwischen Referenzobjekten zweier Teilnetze. Der Koreferenzwert koref gibt an, wie hoch die Ähnlichkeit der Teilnetze ist. Typischerweise beinhalten die Teilnetze etwa 4 - 8 Referenzobjekte.

Wiedererkennung von Häusern vor, da diese explizit mit einem Namen versehen sind.

3.5.3 Selbstlokalisation

Navigation setzt die Kompetenz der Selbstlokalisation in der mentalen Karte voraus. Das bedeutet konkret, dass der Geometrische Agent eine Selbstlokalisierung in seinem internen Umgebungsmodell vornehmen muss, um zu wissen, an welcher Stelle der Instruktion er sich befindet und welcher Teil der Instruktion für ihn momentan relevant ist. Die Koreferenzierung zwischen Objekten aus der Instruktion und der Perzeption dient zwar der Integration von Instruktionsmodell und Perzeptionsmodell, jedoch ist dadurch eine sichere Selbstlokalisation in dem erweiterten Umgebungsmodell noch nicht gegeben.

Im Geometrischen Agenten liegt bereits eine Methode vor, die die Selbstlokalisation vornimmt (siehe Abschnitt 5.4.1 *Pragmatische Annahmen für die Selbstlokalisation*).

Das initiale interne Umgebungsmodell, welches der Agent zu Beginn der Navigationsphase besitzt, besteht nur aus dem räumlichen Teil des Instruktionsmodells, dem I-Netz. Dieses Modell wird jedes Mal, wenn die Prozesse der Verarbeitung der Perzeption, der Koreferenzauflösung oder der Selbstlokalisation ausgeführt werden, um neue Informationen angereichert (durch Erweiterung des P-Netzes und Auflösung von Koreferenzen) oder aktualisiert (durch Bestimmung der aktuellen Position). Das Instruktionsmodell, das Perzeptionsmodell, die Verknüpfung dieser beiden Repräsentationen sowie die Position des Agenten innerhalb dieses Modells bilden das erweiterte interne Umgebungsmodell des Agenten. Dieses Umgebungsmodell beinhaltet zusammen mit dem Aktionsplan die relevanten Informationen, die für den lokalen Planungsprozess verwendet werden.

3.6 Aufgaben des Aktionsmoduls

Auswahl der primitiven Aktionen durch lokale Planung

Aufgabe der lokalen Planung ist es, primitive Aktionen auszuwählen, die der Agent in der simulierten Umgebung ausführen kann. Da natürlichsprachliche Routeninstruktionen typischerweise hinsichtlich bestimmter Aspekte unterspezifiziert sind, muss der initiale Aktionsplan durch lokale Planung weiterverarbeitet, d.h. auf primitive Aktionen abgebildet werden. Wenn z.B. eine Routeninstruktion vorliegt, die nur die Entscheidungspunkte beinhaltet, an denen der Agent eine Richtungsänderung vornehmen soll, dann muss durch die lokale Planung ein adäquates Verhalten zwischen diesen explizit erwähnten Entscheidungspunkten bereitgestellt werden.

Das Aktionsmodul ist für die Auswahl und Ausführung von primitiven Aktionen verantwortlich. Die Instruktionsanweisungen des Aktionsplans werden im Kontext der aktuellen Situation des Agenten interpretiert, indem sie auf primitive Aktionen, die in der simulierten Umgebung ausführbar sind, abgebildet werden. Da der Prozess zur Auswahl der primitiven Aktionen auf Informationen über den aktuellen internen Zustand des Agenten angewiesen ist, die durch die oben genannten Hilfsprozesse bereitgestellt werden, muss während der lokalen Planung eine Koordinierung dieser Prozesse erfolgen. Da zu erwarten ist, dass die Abarbeitung einer Anweisung mehrere atomare Aktionen als Resultat hat, ergibt sich für das Aktionsmodul die Aufgabe, diese Sequenz von Aktionen (und deren Hilfsprozesse), die lokale Aktionssequenz (LAS), in dem lokalen Planungsprozess zu bestimmen.

Koordination der Hilfsprozesse

Der lokale Planungsprozess ist auf Prozesse angewiesen, die das Wissen des Agenten über den aktuellen Zustand manipulieren. Dazu gehören die Prozesse zur Verarbeitung der Perzeption, zur Auflösung von Koreferenzen und zur Auswahl eines Wegstücks bei Bewegungsaktionen (s. u.). Während der Abarbeitung einer Anweisung aus dem Aktionsplan werden somit nicht nur primitive Aktionen des Agenten ausgewählt und ausgeführt, auch die Hilfsprozesse müssen zur rechten Zeit aktiviert werden. Dies führt dazu, dass die lokale Aktionssequenz, auf die die aktuelle Anweisung aus dem Aktionsplan abgebildet wird, nicht nur aus primitiven Aktionen besteht, sondern auch die Aktivierung der Hilfsprozesse beinhaltet.

Auswahl eines Pfades bei Bewegungsaktionen

Ein Aspekt der Unsicherheit besteht in dem Problem der sicheren Auflösung von Koreferenzen. Wenn mehrere potentielle Pfade (die entlang den in der aktuellen Situation wahrgenommenen Routensegmenten verlaufen) für eine Bewegungsanweisung !GO(w) in Frage kommen, muss eines dieser Pfade für die Bewegungsaktion ausgewählt werden.

Umgang mit Lücken

Wird eine Handlung durch einen Ausdruck in der natürlichsprachlichen Routeninstruktion beschrieben, der durch das Lexikon nicht als imperative Anweisung interpretiert wird, kann eine Lücke im Aktionsplan resultieren. Dies ist z.B. der Fall, wenn ein deskriptiver Ausdruck

(„Hinter Haus C befindet sich ein Weg.“) als die Anweisung, diesem Weg zu folgen, zu interpretieren ist. Das Aktionsmodul muss in der Lage sein, derartige Lücken im Aktionsplan zu erkennen und geeignete Maßnahmen treffen, um diese Lücken zu schließen.

Verhalten in Problemfällen

Aufgrund verschiedener Aspekte der Unsicherheit ist nicht auszuschließen, dass eine oder mehrere der oben genannten Prozesse fehlschlagen (d.h. kein Ergebnis liefern) oder ein fehlerhaftes Ergebnis liefern. Dies darf nicht dazu führen, dass der Agent handlungsunfähig wird. Im Idealfall erkennt der Agent das entsprechende Problem und handelt angemessen.

Ein derartiges Problem tritt beispielsweise auf, wenn der Agent einer falschen Route gefolgt ist und sich verlaufen hat. Hat der Agent das Problem erkannt, könnte er die momentane Route aufgegeben und zu einem früheren Entscheidungspunkt zurückkehren, um von dort eine andere Route auszuwählen. Eine andere Möglichkeit besteht darin, im Problemfall einen definierten Zustand einzunehmen. Dieser könnte erreicht werden, indem der Agent zum Ausgangspunkt der Route zurückkehrt.

Das Ziel dieser Arbeit ist es, ein Aktionsmodul zu entwickeln, welches unter Berücksichtigung der oben genannten Probleme dem in Abschnitt 2.2 definierten Rationalitätskriterium genügt. Welche pragmatischen Annahmen und das daraus resultierende Verhalten des Agenten während der Navigationsphase als adäquat einzustufen sind, ist ein wichtiger Untersuchungsgegenstand des Projekts und wird auch bei meiner Arbeit eine entscheidene Rolle spielen. Kapitel 5 *Das Aktionsmodul* beschäftigt sich ausführlich mit der Realisierung einer Ablaufsteuerung für das Aktionsmodul.

Kapitel 4

Aktionsplanung und Ausführung unter Unsicherheit

Die Navigationsaufgabe des Geometrischen Agenten besteht darin, in der simulierten Umgebung durch die Ausführung von primitiven Aktionen das ihm vorgegebene Ziel zu erreichen. Der Prozess, der als Lösung für eine konkrete Zielvorgabe eine Sequenz von auszuführenden Aktionen erzeugt, wird in der KI *Planen* genannt.

Die Abschnitte 4.2 und 4.3 widmen sich dem Paradigma klassischer Planer. Dabei wird sich zeigen, dass die Aufgabenstellung des Geometrischen Agenten aufgrund des Problems der Unsicherheit nicht in die Domäne klassischer Planer fällt. Im Abschnitt 4.4 werden die Konzepte verschiedener Planungsverfahren vorgestellt, die für unsichere Umgebungen entwickelt wurden, und auf eine Verwendungsmöglichkeit für die Entwicklung eines speziellen Planungsverfahrens für den Geometrischen Agenten untersucht. Der Abschnitt 4.5 stellt zwei konkrete Verfahren vor, die in einem Kontext entwickelt wurden, das der Aufgabenstellung des Geometrischen Agenten ähnlich ist. Abschnitt 4.6 zeigt schliesslich, welche Planungskonzepte für den Geometrischen Agenten geeignet sind.

4.1 Grundlagen der Planung

Angemessenes Verhalten in komplexen Umgebungen erfordert es, im voraus zu erwägen, welche Handlungen auszuführen sind, um ein Ziel zu erreichen ([Yang 1997]). Diese Sequenz von Handlungen zu ermitteln, wird in der KI *Planen* genannt ([Russell und Norvig 2003]). Die Aufgabe eines klassischen Planers besteht typischerweise darin, für die Beschreibung eines *Planungsproblems*, das die Spezifikation eines initialen Zustands, der möglichen Aktionen, sowie der Ziele repräsentiert, die Sequenz von Aktionen zu ermitteln, deren Ausführung den Initialzustand in einen Zustand transformiert, in der die Beschreibung des Ziels erfüllt wird ([Poole et al. 1998]). Das Arbeitsfeld der Planung in der KI hat zum Ziel, Kontrollalgorithmen zu entwickeln, die diese Sequenz von Aktionen erzeugen ([Weld 1998]).

Der Planungsalgorithmus, der das Lösungsverfahren bereitstellt, wird dabei unter drei Aspekten beurteilt: *Korrektheit*, *Vollständigkeit* und *Berechnungskomplexität (computation complexity)* ([Lazanas 1995]). Ein Planungsverfahren ist korrekt, wenn jedes Ergebnis des Planungsprozesses tatsächlich eine Lösung darstellt, d.h. jeder ermittelte Plan zu einem Ziel-

zustand führt. Vollständig ist ein Verfahren, wenn es für jedes Planungsproblem aus der spezifischen Domäne des Planers eine Lösung generiert. Die Berechnungskomplexität des Algorithmus gibt schließlich an, wie schnell bzw. in wievielen Schritten eine Lösung gefunden wird unter Berücksichtigung der Größe des Planungsproblems.

Ein idealer Planer ist korrekt, vollständig und besitzt eine niedrige Berechnungskomplexität. Derartige Planer existieren jedoch nur für einen eingeschränkten Aufgabenbereich. Dass bei hinreichend komplexen Umgebungen, die das Problem der Unsicherheit aufweisen, Eingeständnisse an diese Vorgaben gemacht werden müssen, wird in den nächsten Abschnitten gezeigt. In unsicheren Umgebungen bieten sich Planungskonzepte an, die von dem Paradigma des klassischen Planens, bei dem der Planungsprozess allein in der Generierung einer statischen Aktionssequenz besteht, abweichen. In den Abschnitten 4.4 und 4.5 werden verschiedene Ansätze vorgestellt, die sich alternativer Planungs- und Ausführungsverfahren bedienen.

4.2 Klassisches Planen

Bei klassischen Planungsverfahren wird für das Planungsproblem im voraus in der Planungsphase eine geordnete Aktionssequenz erzeugt und diese Sequenz in der Ausführungsphase ausgeführt ([Russell und Norvig 2003]). Das Planungsproblem spezifiziert den Startzustand, die ausführbaren Aktionen und die Zielvorgabe. Ein Zustand repräsentiert dabei die Umwelt des Agenten zu einem bestimmten Zeitpunkt. Aktionen haben Zustandsänderungen zur Folge, indem sie vom aktuellen Zustand zu einem Folgezustand führen.

4.2.1 Das klassische Planungsproblem

Das Planungsproblem wird typischerweise mittels einer formalen Beschreibungssprache formuliert, in der die Zustände und Aktionen spezifiziert werden. Das Planungsproblem wird durch drei Aspekte beschrieben ([Weld 1998]):

- **Repräsentation des Initialzustands:** Der Initialzustand ist der Zustand des Agenten, von dem aus der Plan ausgeführt werden soll. Zustände werden typischerweise durch die Propositionen, die in ihnen gelten, beschrieben.
- **Repräsentation der Ziele:** Die Ziele bestimmen, welcher Zustand durch die Ausführung des Plans erreicht werden soll. Ein Zustand erfüllt ein Ziel (und wird *Zielzustand* genannt), wenn in ihm alle Propositionen gelten, die durch das Ziel beschrieben werden. Im allgemeinen kann es mehrere Ziele für ein Planungsproblem geben.
- **Repräsentation der Aktionen:** Die Aktionen, die der Agent ausführen kann, werden durch ihre Vorbedingungen und Effekte beschrieben. Die Vorbedingungen einer Aktion müssen in ihrem Vorzustand erfüllt sein, damit die Aktion ausgeführt werden kann. Die Effekte beschreiben, wie die Aktion den Nachzustand beeinflusst.

Die Beschreibungssprache bestimmt, was für Planungsprobleme spezifiziert werden können, und welche Planungsverfahren anwendbar sind. Komplexere Probleme erfordern ausdrucksstärkere Sprachen, was zugleich den Nachteil hat, dass die Berechnungskomplexität

steigt. Wichtige Vertreter von klassischen Repräsentationssprachen bzw. Planungsverfahren sind das Situationskalkül (*situation calculus*) ([McCarthy und Hayes 1969]), das Ereigniskalkül (*event calculus*) ([Kowalski und Sergot 1986]), STRIPS ([Fikes und Nilsson 1971]) und ADL ([Pednault 1986]).

4.2.2 Die Domäne klassischer Planungsverfahren

Klassischen Planungsverfahren setzen eine Umgebung voraus, die vollständig beobachtbar, deterministisch, statisch und diskret ist ([Russell und Norvig 2003]). Die Umgebung muss für den Agenten vollständig beobachtbar sein, damit der Agent den Initialzustand, d.h. die in diesem geltenden Propositionen ermitteln kann. Die Anforderung an die Umgebung, sich deterministisch und statisch zu verhalten, ist in der Planungsmethode begründet, die einen Plan im voraus erzeugt, der dann in der Ausführungsphase ausgeführt wird. Nur in einem deterministischen und statischen System kann davon ausgegangen werden, dass durch die Ausführung der geplanten Aktionen tatsächlich die erwünschten Zustände erreicht werden. Die Unterteilung in aufeinanderfolgende Zustände stellt schließlich eine diskrete Modellierung dar, die nur für Umgebungen geeignet ist, die sich diskret verhalten.

Ein Beispiel für eine klassische Planungsdomäne: Die Blockwelt

Im Kontext klassischer Planungsverfahren wurde eine derart restringierte Umgebung geschaffen, die eine gewisse Popularität in der KI-Forschung erworben hat: die Blockwelt (*blocks world*). Obwohl die Domäne der Blockwelt wenig praktische Signifikanz besitzt, ist sie aufgrund ihrer Einfachheit das meist genannte Beispiel der Literatur der KI-Planung seit den sechziger Jahren ([Slaney und Thiébaux 2001]).

In der Blockwelt liegen kubisch geformte Blöcke auf einem Tisch, bzw. sind zu Stapeln aufgetürmt. Ein Roboter hat die Aufgabe, durch Aktionen, durch die die Blöcke bewegt werden, einen Zielzustand zu erreichen, in dem die Blöcke in einer vorgegebenen Art arrangiert sind. In dieser Welt können Zustände durch die Konjunktion von Propositionen wie $\mathsf{On(b,x)}$ beschrieben werden (das bedeutet, Block b befindet sich auf dem Objekt x, wobei x ein anderer Block oder der Tisch sein kann). Die Aktionen des Agenten werden durch ihre Vorbedingungen und Effekte spezifiziert. Eine Spezifikation einer Aktion, die den Block b von der Position x zu der Position y bewegt, sieht in STRIPS folgendermaßen aus:

$$\begin{aligned}&\mathsf{Action(Move(b,x,y))},\\&\quad\mathsf{PRECOND{:}On(b,x)} \wedge \mathsf{Clear(b)} \wedge \mathsf{Clear(y)},\\&\quad\mathsf{EFFECT{:}On(b,y)} \wedge \mathsf{Clear(x)} \wedge \neg\mathsf{On(b,x)} \wedge \neg(\mathsf{Clear(y)}).\end{aligned}$$

Ein Planungsproblem in der Blockwelt beinhaltet den Startzustand (der beschreibt, wie die Blöcke angeordnet sind), die Spezifikation aller ausführbaren Aktionen, und eine Beschreibung des Ziels. Für den Zielzustand könnte beispielsweise gefordert werden, dass Block c auf Block e liegt, wobei auf Block b kein weiterer Block liegen darf ($\mathsf{On(c,e)} \wedge \mathsf{Clear(b)}$). Der Planer hat die Aufgabe, für das Planungsproblem eine Sequenz von Aktionen zu erzeugen, deren Ausführung vom Startzustand in einen Zustand führt, in dem das Ziel erfüllt wird.

4.2.3 Klassische Planungsverfahren

Durch die spezifische Art der Modellierung des Planungsproblems kann die Aufgabe des klassischen Planens durch Problemlösungsverfahren (*problem solving*) gelöst werden, die sich bestimmter Suchstrategien bedienen ([Russell und Norvig 2003]).

Eine der einfachsten Planungsstrategien besteht darin, ausgehend vom Initialzustand des Systems den Zustandsraum nach einem Zielzustand zu durchsuchen (*forward planning*) . Im Zustandsraum repräsentiert ein Knoten einen Zustand und eine Verbindungskante zwischen zwei Zuständen eine Aktion. Die Lösung des Planungsproblems ist ein Pfad im Zustandsraum, der einer Sequenz von Aktionen entspricht, die vom Startzustand zu einem Zielzustand führt. Vollständige Suchstrategien wie A* oder iterative Breitensuche garantieren, dass der Planer eine Lösung findet, wenn es sie gibt. Die Komplexität des Algorithmus hängt dabei vom Verzweigungsfaktor des Zustandsgraphen ab ([Poole et al. 1998]).

Weitere Planungsverfahren nehmen eine rückwärtige Suche im Problemraum vor, bei der nur Aktionen berücksichtigt werden, die zur Lösung des Problems beitragen. Ein Beispiel hierfür ist der STRIPS-Planer, der ein *Divide-and-Conquer*-Verfahren darstellt, da er den Zielzustand in Teilziele zerlegt, die er nach und nach berabeitet ([Poole et al. 1998]). Weitere elaborierte klassische Planungsverfahren sind das *Partial-Order-Planning*-Verfahren, durch das die Sequenz von Aktionen nicht total, sondern partiell geordnet wird, was zu einer höheren Flexibilität bei der Ausführung führt ([Sacerdoti 1975], [Tate 1977], [Soderland und Weld 1991]), sowie das *Graphplan*-Verfahren, welches eine bessere Heuristik bei der Suche im Zustandsraum bietet ([Blum und First 1997]).

Klassische Planungsverfahren sind für Umgebungen prädestiniert, für die das Planungsproblem wie oben beschrieben modelliert, und ein Plan als Lösung im voraus generiert werden kann. Für die Domäne des Geometrischen Agenten kann eine derartige Spezifikation eines Planungsproblems nicht geleistet werden: Weder ist dem Agenten der Initialzustand vor dem Start der Simulation bekannt[1], noch können die Effekte seiner primitiven Aktionen vollständig beschrieben werden. Daher ist es nicht möglich, eine Sequenz von primitiven Aktionen im voraus zu ermitteln, die der Geometrische Agent während der Navigationsphase stur abarbeiten kann.

Aus diesem Grund sind klassische Planungsverfahren für die Aufgabenstellung des Geometrischen Agenten nicht geeignet und sollen deshalb hier nicht weiter untersucht werden. Eine Ausnahme bildet das *hierarchische Planen*, das zwar ein weiteres typisches klassisches Planungskonzept darstellt, aber auch in nichtklassischen Planern, die unter Unsicherheit funktionieren, Verwendung findet.

4.3 Hierarchisches Planen

Hierarchisches Planen bedient sich des Konzepts der *Hierachischen Abstraktion*. Hierbei wird zwischen elementaren Aspekten des Planungsproblems und den unwichtigen Aspekten, die eher Details darstellen, unterschieden. Die Einschränkung des Planungsproblems verringert

[1] Da der Agent die Umgebung nicht kennt, kann er nicht im voraus wissen, welche Objekte er aus seiner Startposition wahrnehmen kann.

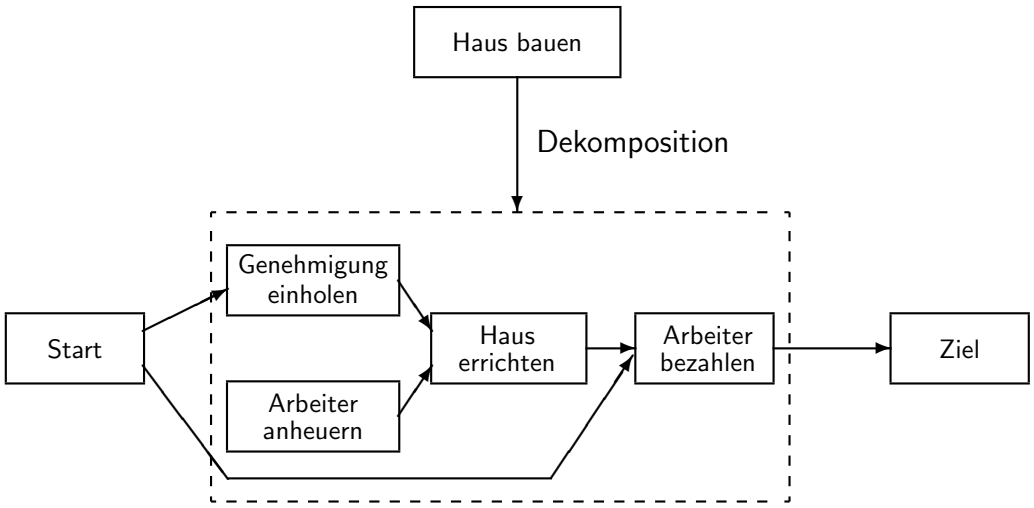


Abbildung 4.1: Eine mögliche Dekomposition der abstrakten Aktion *Haus bauen*. Beispiel nach [Russell und Norvig 2003].

die Größe des Lösungsraums, wodurch eine höhere Performanz des Planers ermöglicht werden soll ([Yang 1997]).

Hierarchische Planverfahren gehen von einem *initialen Plan* aus, der eine abstrakte Beschreibung der zu tätigenden Handlungen darstellt, indem er abstrakte Aktionen beinhaltet, die nicht direkt durch den Agenten ausführbar sind. Dieser Plan kann z.B. durch eines der oben dargestellten Planungsverfahren erzeugt worden sein. Durch einen *Verfeinerungsprozess* werden weitere Pläne generiert, die auf niedrigeren Abstraktionstufen liegen. Als Ergebnis des gesamten Planvorgangs liegt schließlich ein *konkreter Plan* vor, dessen Aktionen direkt ausgeführt werden können.

Pläne werden durch *Dekomposition* von Aktionen verfeinert ([Russell und Norvig 2003]). Eine Aktionsdekomposition (*action decomposition*) reduziert einen abstrakteren Plan (*higher-level plan*) zu einer partiell geordneten Menge von Aktionen, die auf einer niedrigeren Abstraktionsstufe liegen (*lower-level actions*).[2] Regeln zur Aktionsdekomposition repräsentieren das Wissen, wie eine Aktion auf niedrigerer Ebene auszuführen ist, und werden typischerweise in einer Planbibliothek (*plan library*) gespeichert.

Abb. 4.1 zeigt ein Beispiel für eine Planverfeinerung durch einen hierarchischen Planer. Die abstrakte Aktion ‚Haus bauen' wird durch den Verfeinerungsprozess durch einen Plan von Aktionen auf einer niedrigeren Abstraktionsstufe ersetzt. Eine *Dekompositionsregel* aus einer Planbibliothek gibt dabei an, welche Aktionen in dem verfeinerten Plan vorliegen und wie sie partiell geordnet sind. Es können mehrere Dekompositionsregeln für eine abstrakte Aktion in der Planbibliothek vorhanden sein, die sich zusätzlich in den Vorbedingungen und

[2] Die Terminologie in [Yang 1997] weicht von der in [Russell und Norvig 2003] verwendeten Terminologie ab: Yang unterscheidet explizit zwischen *Dekomposition* und *Planverfeinerung*: Der Prozess der Dekomposition liefert Teilpläne, die von einander unabhängig sind und somit parallel von verschiedenen Modulen bearbeitet werden können. Der Verfeinerungsprozess hingegen liefert partiell geordnete Pläne, in denen Aktionen durch die partielle Ordnung nicht vollständig unabhängig voneinander ausgeführt werden können.

Effekten unterscheiden können. Bestimmte Dekompositionen sind unter gewissen Umständen zu präferieren, was einen gewissen Grad an Adaptivität erlaubt.

Die Verfeinerungsprozesse, die die Generierung der Pläne auf den verschiedenen Abstraktionsstufen vornehmen, laufen bei rein hierarchischen Planern alle in einer Planungsphase vor der Ausführung des Plans ab. Da hierdurch ein statischer Plan erzeugt wird, der während der Ausführungsphase strikt befolgt werden muss, zählt auch das rein hierarchische Planen zu den klassischen Planverfahren und stellt keine geeignete Planungsmethode für dynamische oder unsichere Umgebungen dar.

4.4 Planen in unsicheren Umgebungen

Klassische Planverfahren sind geeignet für vollständig beobachtbare, statische und deterministische Umgebungen. Unter diesen Umständen kann durch den Planer im voraus ein Plan erzeugt werden und der Agent kann den Plan mit „geschlossenen Augen“ durchführen ([Russell und Norvig 2003]). In realen Umgebungen sind diese idealen Bedingungen jedoch nicht gegeben. Der Agent muss während der Planausführung die Umgebung und insbesondere die Ergebnisse seiner Aktionen überwachen und bei unerwarteten Begebenheiten sein Verhalten anpassen.

Ein einfaches Verfahren, das in unsicheren Umgebungen anwendbar ist, ist konditionales Planen (*conditional planning*). Ein bedingter Plan (*conditional plan*) wird wie bei klassischen Verfahren im voraus erzeugt, jedoch besitzt der Plan eine graphenartige Struktur. Während der Planausführung wird in jedem Knoten mittels Perzeption der Umgebung festgestellt, welche Bedingungen für die Fortsetzung des Plans erfüllt sind, und die entsprechende Folgeaktion ausgewählt. Dieses Planungsverfahren ist vollständig und korrekt, wenn es alle möglichen zukünftigen Zustände berücksicht. Damit ist es nur für Umgebungen mit beschränkter Unsicherheit einsetzbar.

Für den Geometrischen Agenten ist ein konditionales Planungsverfahren nicht geeignet. Der Agent ist während der Navigation auf Informationen angewiesen, die den aktuellen Zustand des Agenten und der Umwelt beschreiben. Diese Informationen spielen eine wesentliche Rolle bei der Auswahl der zu tätigenden Aktionen, wie auch bei der Intantiierung der Parameter der Aktionen. Als Argumente der Aktionen fungieren Objekte des Perzeptionsmodells. Da das Perzeptionsmodell dynamisch während der Navigation durch jede neue Perzeption aktualisiert wird, kann der Agent unmöglich im voraus durch ein konditionales Planverfahren alle möglichen Intantiierungen berücksichtigen.

4.4.1 Ausführungsüberwachung und Neuplanung

Die Konzepte *Ausführungsüberwachung* (*execution monitoring*) und *Neuplanung* (*replanning*) sind keine eigentlichen Planungsverfahren, da sie nicht der Plangenerierung dienen, sondern bei der Planausführung effektive Hilfsverfahren darstellen, durch die der Agent auf unerwartete Veränderungen der Umwelt reagieren kann ([Russell und Norvig 2003]). Der Plan wird vor der Ausführung mittels einer konventionellen Planungsmethode wie z.B. *partial-order planning* oder *conditional planning* erzeugt. Anders als bei den klassischen Planverfahren ist der Agent bei der Ausführung dieses Plans jeodch nicht gezwungen, den im voraus erzeugten

Plan strikt zu befolgen, sondern kann, wenn es die Umstände erfordern, eine Änderung des Plans vornehmen.

Während der Planabarbeitung wird vor Ausführung einer Aktion durch Perzeption festgestellt, ob der Plan in der aktuellen Situation noch Erfolg verspricht (*execution monitoring*) und ggf. eine Revision des Plans vorgenommen (*replanning*). Dies ermöglicht es dem Agenten, auch trotz unvorhersehbarer Ereignisse handlungsfähig zu bleiben.

In [Russell und Norvig 2003] werden die Überwachungsverfahren *Aktionsüberwachung*, *Planüberwachung* und *Neuplanung* vorgestellt:

- **Aktionsüberwachung** (*action monitoring*) ist die einfachere Variante der Ausführungsüberwachung. Hierbei wird vor Ausführung einer Aktion geprüft, ob die Vorbedingungen dieser Aktion erfüllt sind. Ist dies nicht der Fall, muss der aktuelle Plan aufgegeben und der Planer erneut angestoßen werden, um für die neuartige Situation einen neuen Plan zu generieren (siehe Neuplanung).

- **Planüberwachung** (*plan monitoring*) stellt eine elaboriertere Variante der Ausführungsüberwachung dar. Hierbei wird nicht nur die nächste Aktion, sondern der gesamte noch abzuarbeitende Plan auf Ausführbarkeit überprüft und ggf. revidiert. Dies wird dadurch erreicht, dass vor Ausführung einer Aktion die Vorbedingungen aller noch auszuführenden Aktionen geprüft werden. Unberücksichtigt bleiben jedoch die Vorbedingungen, die erst zu einem späteren Zeitpunkt der Planausführung erfüllt werden. Die Planüberwachung hat gegenüber der Aktionsüberwachung den Vorteil, dass die Aussichtslosigkeit eines Plans so früh wie möglich erkannt wird. Somit kann ein Plan zu einem frühen Zeitpunkt revidiert werden, und es kann nicht passieren, dass der Agent einige nutzlose Aktionen durchführt, um dann erst spät die Undurchführbarkeit des Plans zu erkennen. Ein weiterer Vorteil besteht darin, dass durch die Planüberwachung festgestellt werden kann, ob durch einen glücklichen Zufall (*serendipity*) aufgrund von äußeren Einflüssen ein Zustand eingetreten ist, der einem später im Plan befindlichen Zustand entspricht. Die Aktionen, die zu diesem Zustand führen, sind somit überflüssig geworden und können übersprungen werden.

- Durch **Neuplanung** (*replanning*) wird eine Revision des Plans vorgenommen. Dies kann dadurch geschehen, dass der Planer einen vollständig neuen Plan für das aktualisierte Planungsproblem (mit dem aktuellen Zustand als Initialzustand) erzeugt. Eine andere Methode besteht darin, den schon vorhandenen Plan zu modifizieren bzw. zu reparieren. Dies wird erreicht, indem von dem aktuellen Zustand aus (der kein Zustand des Plans ist) eine Aktionssequenz gefunden wird, die zu einem Zustand des ursprünglichen Plans führt.

Für den Geometrischen Agenten ist das Prinzip der Aktionsüberwachung durch die Perzeption der Umgebung vor Ausführung einer Aktion unumgänglich. Durch die Perzeptionsprozesse werden nicht nur die Vorbedingungen von Instruktionsanweisungen geprüft[3], sondern auch der wesentliche Parameter einer primitiven Aktion festgelegt.

[3] Siehe Abschnitt 5.1.4

4.4.2 Kontinuierliches Planen

Kontinuierliches Planen (*continuous planning*) erlaubt es einem Agenten, über einen unbeschränkten Zeitraum in einer Umgebung zu planen und zu handeln. Die Prozesse der Zielformulierung, der Plangenerierung und der Planausführung sind nicht mehr wie bei klassischen Planverfahren (bei denen erst das Ziel formuliert, dann geplant, und zuletzt der Plan ausgeführt wird) zeitlich getrennt, sondern werden in einem kontinuierlichen Prozess simultan ausgeführt ([Russell und Norvig 2003]).

Der kontinuierlich planende Agent bedient sich dabei verschiedener Planungskonzepte wie der Hierarchischen Abstraktion, der Planverfeinerung oder der Ausführungsüberwachung. Er verfügt nicht über einen statischen Plan, sondern kann dynamisch Pläne erzeugen und verarbeiten, indem er während seiner Lebensdauer neue Ziele von außen erhält oder selbst kreiert. Durch die dynamische Ziel- und Plangenerierung können verschachtelte Ziele entstehen (*task interleaving*), die in verschachtelten Teilplänen resultieren. Schon während der Generierung der Pläne kann er Planverfeinerungen von einzelnen Teilplänen vornehmen und primitive Aktionen ausführen, oder Modifizierungen an abstrakten Teilplänen vornehmen, wenn neue Informationen aus der Perzeption vorliegen ([Aylett et al. 2000]).

Eine Umsetzung eines kontinuierlichen Planes wird durch das RAP-System realisert, das im nächsten Abschnitt vorgestellt wird.

4.4.3 Das RAP-System

Firby beschreibt in [Firby 1994] und [Firby 1995] das *Reactive-Action-Package*-System (RAP-System), welches einen Roboter mit der Fähigkeit ausstattet, flexibel mit seinen Plänen umzugehen, um den Bedingungen einer Realwelt-Umgebung gerecht zu werden. Das RAP-System stellt einen kontinuierlichen Planer bereit, der zusätzlich von den Planungskonzepten der hierarchischen Abstraktion und der Ausführungsüberwachung Gebrauch macht.

Anders als in herkömmlichen Verfahren, in denen ein Plan als eine Sequenz von primitiven Roboteraktionen verstanden wird, die strikt nacheinander ausgeführt werden, soll durch eine geeignete Strukturierung des Plans eine situationsabhängige Interaktion mit der unsicheren Umgebung bei der Planausführung ermöglicht werden. Ein Plan besteht im RAP-System aus Aufgaben, die entweder primitiven Aktionen entsprechen oder mit Hilfe der RAP-Repräsentation spezifiziert werden (*RAP-defined tasks*). Ein RAP repräsentiert drei Aspekte, um in Abhängigkeit der aktuellen Situation interpretiert werden zu können: Einen Test auf Erfolg, ein Aktivitätsfenster und einer Menge von ausführbaren Methoden, die angemessen für unterschiedliche Situationen sind. Realisert wird ein RAP als ein kontextsensitives Programm, welches durch die Verwendung von Subroutinen bzw. Subaufgaben eine hierarchische Struktur besitzen kann.

Das RAP-System stellt eine Repräsentationssprache, einen Speicher für sensorische Eingaben und einen Interpreter bereit. Um angemessen auf unerwartete Ereignisse der Realwelt-Umgebung reagieren zu können (das bedeutet, angemessene Methoden für die Erfüllung einer Aufgabe auszuwählen), ist das System auf das Konzept der Ausführungsüberwachung angewiesen. Die Umwelt wird mittels Aktionen zur Perzeption (*active sensing processes*) wahrgenommen. Der RAP-Interpreter kann als ein kontinuierlicher Planer verstanden werden, der dynamisch während der Ausführung einer Aufgabe neue Subaufgaben erzeugt und Perzepti-

onsaktionen anstößt.

Ein Plan wird im RAP-System wie folgt abgearbeitet: Zuerst wird eine unerfüllte Aufgabe ausgewählt. Wenn die Aufgabe in der aktuellen Situation durch eine primitive Aktion bearbeitet werden kann, wird diese Aktion direkt ausgeführt. Gibt es keine primitive Aktion, wird durch den Interpreter mit Hilfe der Planbibliothek dasjenige RAP ermittelt, welches als komplexe Methode die Abarbeitung der Aufgabe vornimmt. Liefert ein Vergleich der aktuellen Situation mit dem Test auf Erfolg des RAPs ein positives Ergebnis, gilt das RAP als erfolgreich abgearbeitet, und die nächste übergeordnete Aufgabe kann bearbeitet werden. Anderenfalls wird ermittelt, welche Methode des RAPs am besten zu der aktuellen Situation passt. Nach Ausführung dieser Methode wird erneut auf Erfolg getestet. Bei einem positiven Testergebnis wird die nächste übergeordnete Aufgabe zur Abarbeitung ausgewählt, bei einem negativen Ergebnis wird eine andere Methode des RAPs ausgeführt.

Im RAP-Projekt soll wie beim Geometrischen Agenten eine abstrakte Aufgabe im Kontext der aktuellen Situation durch geeignete Methoden bzw. Aktionen abgearbeitet werden. Sowohl für den Test, ob eine Aufgabe aus dem Plan erfolgreich ausgeführt wurde, als auch für die Bestimmung der geeigneten Methode sind der RAP-Agent wie auch der Geometrische Agent auf die Perzeption der Umgebung angewiesen.

Eine lokale Planung, wie sie für den Geometrischen Agenten vorgesehen ist, kann durch das RAP-System jedoch nicht geleistet werden, da die möglichen Methoden für die Erfüllung einer Aufgabe im RAP-System statisch in einer Planbibliothek abgelegt sind und nicht dynamisch erzeugt werden können. Dies ist aber bei der Auswahl von Bewegungsaktionen notwendig, da der wesentliche Parameter der Bewegungsaktion, welchem Wegstück zu folgen ist, erst direkt vor Ausführung der Aktion ermittelt werden kann.

4.4.4 Die Plan-as-communication-Sichtweise

Agre und Chapman unterscheiden in [Agre und Chapman 1990] zwei Möglichkeiten, die Aufgabe, die Verwendungsweise und den Nutzen von Plänen zu interpretieren. Dem Paradigma des klassischen Planens, welches sie die *Pläne-als-Programm-Sichtweise* (plan-as-program view) nennen, setzen sie die Sichtweise der *Pläne-als-Kommunikation* (plan-as-communication view) entgegen.

Bei der *Pläne-als-Programm-Sichtweise*, die den klassischen Planverfahren zugrunde liegt, entsprechen die primitiven Aktionen des im voraus erzeugten Plans parametrisierten Programmanweisungen, die während der Ausführungsphase strikt nacheinander ausgeführt werden, ohne dass der Agent spezifisches Wissen über den Kontext oder die Aktivitäten besitzt, in die er involviert ist. Der Agent verfügt über kein Domänen- oder Umgebungswissen außer dem Wissen, welches implizit durch den Plan und den Interpreter, der für die Ausführung der primitiven Aktionen zuständig ist, gegeben ist. Agre und Chapman nennen vier Gründe, weshalb diese Sichtweise nicht für Realwelt-Umgebungen geeignet ist:

- Durch die Formalisierung des Planungsprozesses wird man bei Anwendung auf hinreichend komplexe Umgebungen mit Problemen konfrontiert, die aufgrund der hohen Komplexität praktisch nicht mehr handhabbar sind (Explosion des Zustandsraums).

- Es ist unmöglich, auf unvorhersehbare Ereignisse angemessen zu reagieren. Da der Agent nicht flexibel mit seinen Plänen umgehen kann, führt der Misserfolg einer Aktion des Agenten zum Fehlschlag des gesamten Plans.

- Um die Komplexität des Planungsprozesses zu verringern, ist es sinnvoll, vom Konzept der hierarchischen Abstraktion Gebrauch zu machen. Pläne werden dabei auf abstrakter Ebene erstellt und müssen entweder während des Planungsprozesses (wie bei rein hierarchischen Verfahren) oder während der Planausführung konkretisiert, d.h. mit Details angefüllt werden. Einfache Systeme für die Planausführung, wie sie für klassische Planungsverfahren typisch sind, fordern jedoch einen hohen Detaillierungsgrad des Plans, da die Möglichkeiten, die sie zur Anreicherung des Plans bieten, beschränkt sind.

- Bei der Ausführung eines Plans muss eine Verbindung (*causal connection*) zwischen dem Plan und der konkreten Situation, d.h. den im Plan genannten Objekten und Entitäten der Umgebung des Agenten hergestellt werden. Will z.B. ein Roboter in der Blockwelt[4] eine Aktion wie Move(b,x,y) ausführen, muss er eine Möglichkeit besitzen, die genannten Blöcke in seiner Umgebung zu identifizieren. Bei klassischen Planungsverfahren wurde von diesem Problem abstrahiert. Beim Plangebrauch in Realwelt-Umgebungen werden jedoch domänenspezifische Fähigkeiten benötigt, die eine Verbindung zwischen dem Plan und der aktuellen Situation herstellen, damit der Agent situationsabhängig handeln kann.[5]

Durch diese Probleme sehen sich Agre und Chapman veranlasst, ein neues Paradigma des Plangebrauchs vorzustellen, welches diese Schwierigkeiten umgeht. Die *Plan-als-Kommunikation-Sichtweise* spricht Plänen eine geringere Rolle bei der Auswahl von Aktionen zu. Sie ist Teil der Theorie der *situierten Aktivität* (*situated activity*), in der Handlungen immer situations- bzw. kontextbezogen vollzogen werden ([Suchman 1987]). Ein Plan wird nicht strikt ausgeführt, er wird vielmehr als eine Ressource für Information neben anderen Ressourcen gehandelt. Agre und Chapman sprechen in diesem Zusammenhang von dem *Gebrauch des Plans* (*plan use*) im Gegensatz zur *Ausführung eines Plans* (*plan execution*). Um zu bestimmen, welche Aktivität durch eine Instruktion im Plan beschrieben wird, ist ein kontinuierlicher interpretativer Arbeitsprozess notwendig, der Wissen über die Domäne und die in ihr ausführbaren Aktionen voraussetzt. Der Agent muss zur Improvisation fähig sein, d.h. er muss die Instruktion im Kontext der aktuellen Situation bewerten können ([Agre und Chapman 1990]). Dies kann auch weitere konkrete Aktionen oder Prozesse erfordern, die nicht im Plan erwähnt werden, wie Perzeptionsaktionen oder Verabeitungsprozesse, die die aktuellen Informationen manipulieren, um zu ermitteln, in welcher Beziehung sie zum Plan stehen.

Agre und Chapman beschränken den Begriff des *Plans* nicht nur auf den Kontext von automatischen Systemen. Pläne werden als soziale Konstrukte im alltäglichen Leben von Menschen verwendet, intern oder extern repräsentiert und eben auch kommuniziert, d.h. mittels eines Repräsentationssystems anderen Personen mitgeteilt. Pläne können dabei auf unterschiedliche Arten übermittelt werden (z.B. mit gesprochener Sprache, als Skizze oder als Betriebspläne).

[4] Siehe Abschnitt 4.2.2.

[5] Im Geometrischen Agenten wird diese Fähigkeit durch den Prozess der Koreferenzauflösung geleistet.

In [Agre und Chapman 1990] werden für die Erläuterung der Pläne als Kommunikations-Sichtweise drei Beispielszenarien vorgestellt. Für diese Arbeit ist das zweite Beispiel interessant, da es der Aufgabenstellung des Geometrischen Agenten sehr nahe kommt. In dem Beispiel wurden Personen mit der Aufgabe konfroniert, von dem Haus, in dem Agre wohnt, zur nächstgelegenen U-Bahnstation zu gelangen. Zur Bewältigung der Aufgabe konnten die Versuchspersonen (die sich in der Umgebung nicht auskannten) auf einen auf einen Zettel geschriebenen Routenplan zurückgreifen. Die Pläne, die Instruktionen wie „links den Berg hoch“ („*left up the hill*“) oder „nimm die erste rechts“ („*take the first right*“) beinhalteten, sind unterspezifiziert, da zum einen manche Aktionen gar nicht beschrieben sind (wie z.B. einen kleines Stückchen nach rechts zu gehen, bevor „links den Berg hoch“ ausgeführt werden kann)[6], und zum anderen Parameter von Anweisungen wie die Straße in „nimm die erste rechts“ erst während der Navigation identifiziert werden können.

Der Begriff *Pläne als Kommunikation* ist motiviert durch die Bedeutung (*meaning*), die einem Plan zugewiesen wird. Wie bei sprachlichen Entitäten, die kontextabhängig interpretiert werden, kann die Bedeutung einer Anweisung aus dem Plan ebenfalls erst im Kontext der aktuellen Situation verstanden und in adäquate primitive Aktionen umgesetzt werden. Anders als bei klassischen Plänen kann und soll ein Plan nicht vollständig die intendierte Aktivität beschreiben. Drei Aspekte machen die Nutzung derart unterspezifizierter Pläne möglich ([Agre und Chapman 1990]):

- **Indexikalität** (*indexicality*): Objekte aus der Instruktion weisen auf ein Objekt der Umgebung, das erst im Kontext der Situation in der Umgebung ermittelt werden kann. Dadurch kann die Beschreibung kompakt ausfallen. Wenn z.B. in einer Situation nur eine Straße vorhanden ist, die vom aktuellen Weg nach rechts abbiegt, reicht die Anweisung „geh nach rechts“ aus dem obigen Beispiel aus, ohne dass der Straßenname explizit genannt werden muss.

- Durch **Projektion** (*projection*) wird eine gewisse Voraussicht auf zukünftige Situationen ermöglicht. Sowohl der Instrukteur als auch der Instruierte müssen ein hinreichend gemeinsames Verständnis von den Effekten einer Aktion besitzen. Durch das Abschätzen der Effekte von Aktionen kann eine Anweisung auf eine Abfolge von Situationen projeziert werden.

- **Reflexivität** (*reflexivity*): Der Instrukteur und der Instruierte haben nicht nur ein gemeinsames Verständnis, sie setzen dieses gemeinsame Verständnis auch wechselseitig bei der Plangenerierung bzw. -nutzung voraus. So wäre es denkbar, dass die Anweisung „geh nach rechts“ in einer Situation sofort ausgeführt werden könnte, aber auf einen Hinterhof führen würde. Der Instruktor kann aber das Verständnis des Instruierten voraussetzen, die Anweisung nur auf Objekte zu beziehen, die ein gewisses Maß an Erfolg versprechen (in diesem Fall auf öffentliche Straßen oder Wege).

Die Unsicherheit wird trotz dieser Aspekte nicht vollständig eliminiert. So kann es zu Problemen beim Verstehen der Instruktion kommen, wenn der Instruierte eine Anweisung anders interpretiert als es der Instruktor vorausgesetzt hat.

[6] Dies würde beim Geometrischen Agenten zu einer Lücke im Aktionsplan führen.

Das oben genannte Beispielszenario besitzt eine hohe Ähnlichkeit mit der Aufgabenstellung des Geometrischen Agenten. Der Geometrische Agent erhält ebenfalls eine Routeninstruktion, die textlich verfasst ist. Sie kann als Plan aus der *Pläne-als-Kommunikations-Sichtweise* aufgefasst werden, da sie dem Agenten durch den Instrukteur mitgeteilt wird und hinsichtlich der intendierten Aktivität unterspezifiziert ist. Auch hier stellt das Instruktionsmodell, welches aus der Verarbeitung der natürlichsprachen Wegbeschreibung resultiert, nicht die einzige Informationsressource dar, durch die die Aktionen des Agenten bestimmt werden. Der Agent ist gleichzeitig auf Informationen aus der Perzeption angewiesen, um die Instruktionsanweisungen aus dem Aktionsplan in adäquate primitive Aktionen umsetzen zu können. Hierfür muss der Agent während der lokalen Planung Perzeptionsaktionen, die nicht im Plan genannt werden, sowie weitere Verarbeitungsprozesse zur Aktualisierung seines internen Zustands durchführen.

Eine Modellierung einer mit dem oben genannten Beispiel vergleichbaren Beispielumgebung und eines darin navigierenden Agenten soll durch das Projekt Geometrischer Agent realisiert werden.

4.4.5 Die Plan-as-behaviour-Sichtweise

In [Hayes-Roth et al. 1993] wird eine konkrete Agentenarchitektur vorgestellt, die für eine dynamische und unsichere Umgebung, wie die eines Wachroboters, geeignet ist. Die Autoren schließen sich im wesentlichen der im vorigen Abschnitt vorgestellten Sichtweise an[7], sprechen allerdings Plänen die Funktion zu, das intendierte Verhalten von Agenten (*intended course of behaviour*) zu beschreiben. Die zu Anfang des Artikels gestellte Frage „How should plans influence the behaviour of intelligent agents?“ macht deutlich, dass Pläne als Orientierung und nicht als strikte Vorgaben verstanden werden.

Im Gegensatz zum Konzept des kontinuierlichen Planens wird zwischen einem Prozess der Planung (*planning*) und einem Prozess des Planfolgens (*plan following*), welches bewusst dem einfacheren und starren Prozess der Planausführung (*plan execution*) gegenübergestellt wird, unterschieden. Da Pläne nur der Orientierung dienen, können sie nicht direkt von einem Agenten ausgeführt werden. Der Agent wählt während des Planfolgens Verhaltensweisen aus, die mehr oder weniger konsistent mit den Anweisungen des Plans sind.

Die in [Hayes-Roth et al. 1993] vorgestellte Agentenarchitektur besteht aus zwei funktionalen Modulen, die verschiedene Abstraktionsstufen (*level*) repräsentieren. Die *physikalische Stufe* (*physical level*) stellt die Schnittstelle zur Umgebung dar, indem sie Methoden zur Perzeption und Aktion bereitstellt. Die *kognitive Stufe* (*cognitive level*) implementiert abstrakte Prozesse, die für die Beurteilung der aktuellen Situation, die Planung oder das Lösen von Problemen verantwortlich sind. Die kognitive Stufe besitzt keinen direkten Zugang zur Umgebung und kann nur über die physikalische Stufe Einfluss auf die Umgebung vornehmen. Die Struktur der beiden Schichten ist isomorph zueinander, so dass vergleichbare funktionale Einheiten innerhalb der Stufen vorhanden sind. Die Repräsentation der Informationen innerhalb der Einheiten ist jedoch verschieden. So finden sich in beiden Stufen ein Weltmodell, eine Menge von Verhaltensweisen, ein Kontrollplan und ein *meta-controller* zur Steuerung

[7] Die beiden Planungsparadigmen werden hier *plans are programs* und *plans are not programs* genannt.

der Aktivitäten, wobei die Repräsentation dieser Einheiten verschieden abstrakt ist. In der kognitiven Stufe werden Informationen symbolisch repräsentiert, in der physikalischen Stufe metrisch. Pläne der kognitiven Ebene werden der physikalischen Ebene übermittelt; die physikalische Ebene liefert Informationen über die Perzeption oder den Erfolg von Aktionen an die kognitive Ebene. Jede Schicht besitzt einen *meta-controller*, der in einem kontinuierlichen Prozess Verhaltensweisen auswählt, die am besten zum aktuellen Schritt des Kontrollplans und der aktuellen Situation des Agenten passen.

Pläne, Planung und der Prozess des Planfolgens werden von den Autoren in diesem Kontext so definiert: Ein Plan ist eine Sequenz von Planschritten. Ein Planschritt ist ein Tripel <task, parameters, constraints>. ‚Task' gibt dabei an, welche Aufgabe durch den Planschritt erfüllt werden soll. Als Beispiel wird die Aufgabe eines Wachroboters genannt, in einer ihm bekannten Umgebung einer Route zur Überwachung zu folgen. Im Beispiel spezifiziert der Parameter den Zielpunkt der Route. Die Zusatzbedingungen (‚constraints') sind eine geordnete Liste von Performanzkriterien (wie z.B. ‚sicher' oder ‚schnell'). Ein Planschritt ist nicht direkt ausführbar, sondern dient nur der Beschreibung der intendierten Aktivität. Als *Planung* wird jeder Prozess verstanden, der eine bestimmte Sequenz von Planschritten erzeugt. *Planfolgen* ist die Ausführung von Verhaltensweisen, die am besten zu einem sukzessiven Abschnitt von Planschritten passt. Hierfür ermittelt der Agent eine Verhaltensweise, die durch eine Intantiierung mit den entsprechenden Parametern den Planschritt erfüllt. Kommen mehrere Verhaltensweisen in Frage, wird diejenige ausgewählt, die am besten die Zusatzbedingungen erfüllt. Findet der Agent keine passende Verhaltensweise, wird eine situationsabhängige Defaultstrategie (*default-policy*) angwendet, die sicherstellen soll, dass der Agent in jeder Situation handlungsfähig bleibt.

Durch die hierarchische Stufung, die zur Repräsentation von Plänen auf verschiedenen Abstraktionsebenen führt, besteht eine Ähnlichkeit zum Verfahren des hierarchischen Planens. Anders als bei hierarchischen Planern, bei denen abstrakte Aktionen deterministisch auf Sequenzen von primitiven Aktionen abgebildet werden, beschreiben Planschritte in den hier vorgestellten Kontrollplänen Aufgaben, die durch verschiedene Verhaltensweisen erfüllt werden können. Pläne, die das *intendierte* Verhalten von Agenten beschreiben, sind entkoppelt von den aktuellen *ausführbaren* Verhaltensweisen, so dass ein Plan sogar dazu führen kann, dass der Agent Wissen verwendet, das er zur Zeit der Generierung des Plans noch nicht besessen hat, oder Verhaltensweisen ausführt, über die er noch nicht verfügt hat (d.h. der Agent kann neue Verhaltensweisen *lernen*).

Auch hier lassen sich Ähnlichkeiten zum Aufbau des Geometrischen Agenten erkennen. In beiden Systemen greifen Kontrollmodule (meta-controller bzw. Aktionsmodul) auf Informationen über die Umgebung und die zu bearbeitenden Aufgaben (Kontrollplan bzw. Aktionsplan) zurück, um die Aktivität des Agenten zu bestimmen. Umgebungswissen liegt bei der oben dargestellten Architektur in zwei verschiedenen Repräsentationen vor: auf kognitiver Ebene in einem Umgebungsmodell, welches symbolische Informationen über die Konstellation und Eigenschaften der einzelnen Objekte der Umgebung beinhaltet. Auf physikalischer Ebene werden diese Eigenschaften nicht repräsentiert, dafür aber metrische Informationen über diese Objekte. Im Geometrischen Agenten sind auch zwei Modelle für die Repräsentation der Umgebung vorhanden: Das Instruktionsmodell, welches räumliche Informationen über

die Umgebung aus der Wegbeschreibung beinhaltet, sowie das Perzeptionsmodell, welches die Objekte und deren Eigenschaften aus der aktuell wahrgenommenen Szene repräsentiert. Anders als bei der oben dargestellten Architektur ist der Repräsentationsformalismus trotz unterschiedlichen Inhalts dieser beiden Modelle gleich (beide Modelle werden in CRIL spezifiziert), so dass eine Integration der beiden Modelle in ein erweitertes Umgebungsmodell vereinfacht wird. Der Kontrollplan ähnelt dem Aktionsplan des Geometrischen Agenten, auch er ist nicht direkt ausführbar. Die Repräsentation der Planschritte resp. der Instruktionsanweisungen ist ähnlich. Ein Planschritt wird im Kontrollplan als ein Tripel <task, parameters, constraints> dargestellt, eine Instruktionsanweisung im Aktionsplan als ein Tripel <Anweisungstyp, Parameter, Situation>. Die ersten beiden Parameter sind vergleichbar: Sowohl ‚task' wie auch ‚Anweisungstyp' geben den Typ der Anweisung an; an zweiter Stelle stehen die Argumente der Anweisung. Das dritte Argument ist jedoch verschieden. Durch ‚constraints' werden zusätzliche Bedingungen an die auszuwählende Verhaltensweise gestellt, ‚Situation' weist auf die Situation hin, in der die Anweisung stattfindet, und ermöglicht eine temporale Ordnung der Anweisungen.

Sowohl der Kontrollplan wie auch der Aktionsplan können zu verschiedenen Verhaltensweisen führen, die im Kontext der aktuellen Situation ausgewählt werden und die Ermittlung zusätzlicher Information voraussetzen. Ein auch für den Geometrischen Agenten interessantes Konzept ist das der Default-Strategie. In Abschnitt 5.7 *Folgeaktionen* wird gezeigt, dass auch für den Geometrischen Agenten eine derartige Strategie notwendig ist, die die Handlungsfähigkeit des Agenten in bestimmten Problemsituationen gewährleistet.

Um eine geeignte Route für eine Navigationsaufgabe zu ermitteln, verfügt der Roboter in dem in [Hayes-Roth et al. 1993] vorgestellten Projekt über eine vollständige und sichere topologische Repräsentation seiner Umgebung. Der Prozess, der eine geeignete Route auswählt (*plan routes*), durchsucht den topologischen Graph nach einer befahrbaren Route. Diese Route wird dann unter Einbeziehung der physikalischen Schicht abgefahren, wobei je nach Aufgabenstellung entweder von Koppelnavigationsmechanismen Gebrauch gemacht wird oder ein reaktiver Kontrollmechanismus mit Infrarotsensoren Kollisionen vermeidet. Das Hauptproblem des Geometrischen Agenten besteht jedoch darin, mit unvollständigem und unsicherem Wissen über seine Umgebung konfrontiert zu sein.

4.5 Verfahren für die instruktiongestützte Navigation eines Roboters

4.5.1 Das Robot Navigation-Projekt

Das *Robot-Navigation*-Projekt widmet sich der Entwicklung eines Navigationsverfahrens für autonome Roboter, durch das elementare Verhaltensweisen (*basic behaviours*) des Roboters, eine geeignete Repräsentation von Routen und bestimmte Navigationstechniken bereitgetellt werden ([Müller et al. 2000], [Röfer und Lankenau 2002]).

Für die Navigationsaufgabe nutzt der Roboter eine Beschreibung der Route (*route description*). Innerhalb des Projekts werden zwei Navigationsverfahren entwickelt, die durch eine unterschiedliche Generierung und Verwendung der Routenbeschreibung für verschiedene Szenarien geeignet sind. Im ersten Verfahren wird der Roboter durch einen Lernprozess mit

Informationen über die Umgebung versorgt. Er besitzt dadurch Wissen über die Umgebung, das er nach der Lernphase für die autonome Navigation verwenden kann. Das zweite Verfahren setzt kein Vorwissen über die Umgebung voraus. Die Routenbeschreibung wird hier als eine formale qualitative Repräsentation dem Roboter mitgeteilt. Im folgenden werden die beiden Verfahren vorgestellt.

Generierung einer Routenbeschreibung durch einen Lernprozess

Als Testumgebung des ersten in [Röfer und Lankenau 2002] beschriebenen Verfahrens fungiert der Campus der Unversität Bremen, in der ein autonomer Rollstuhl (der *Bremen Autonomous Wheelchair*) selbstständig navigieren soll. In den Routenbeschreibungen wird von der tatsächlichen Bewegung des Roboters abstrahiert, indem Routensegmente als gerade Kanten zwischen Entscheidungspunkten repräsentiert werden. Um diese Segmente sind rechteckige Kästen gelegt, die die Region beschreiben, innerhalb der die tatsächliche Bewegung des Roboters stattzufinden hat (*acceptance areas*). Da diese eindimensionale Repräsentation von Routen nicht für komplexe Navigationsaufgaben geeignet ist, wird der Formalismus um die Repräsentation von Routengraphen erweitert. Ein Routengraph repräsentiert in einer graphenartigen Struktur die topologischen und metrischen Eigenschaften (Länge von Routensegmenten und Winkel zwischen Routensegmenten) der Umgebung.

Eine Routenbeschreibung ist eine endliche Sequenz von Eckpunkten, den sogenannten *corners*. Eine *corner* beinhaltet sowohl den Winkel α, der durch das ein- und ausgehenden Routensegment des Eckpunkts eingeschlossen wird, sowie die Länge l des ausgehenden Routensegments. Als Beispiel für eine Routenbeschreibung wird die Sequenz „(0°, 800cm),(89°, 345cm),(-83°, 566cm)“ angegeben.

Die Routenbeschreibung wird in einer Lernphase gewonnen, in der der Roboter manuell gesteuert wird. Dabei werden die odometrischen Daten des Roboters gesammelt und durch einen Generalisierungsprozess in eine kompakte Form gebracht. Diese Information wird gespeichert und für zukünftige autonome Durchläufe verwendet.

Während eines autonomen Durchlaufs werden ebenfalls die odometrischen Daten gesammelt und wie in der Lernphase in die kompakte Form gebracht. Durch einen Vergleich der Routenbeschreibung mit den aktuellen Daten des Roboters, in der die Entfernung zum letzten Eckpunkt berücksichtigt wird, kann der Roboter eine Selbstlokalisation innerhalb der Route vornehmen.

Um die Robustheit des Navigationsverfahrens zu erhöhen, wird der Roboter zusätzlich mit elementaren Verhaltensweisen (*basic behaviours*) ausgestattet. Diese Verhaltensweisen beschreiben Aktivitäten des Roboters auf einer abstrakten Ebene, so dass die Ausführung einer Verhaltensweise in Abhängigkeit der aktuellen Situation erfolgt, was die Navigation von der exakten Struktur der Umgebung (die z.B. dynamische Objekte wie bewegliches Mobiliar beinhaltet) unabhängig macht. Beispiele für elementare Verhaltensweisen sind Aktivitäten wie einem Korridor zu folgen (*cooridor-following*) oder durch eine Tür zu gehen (*door-passage*). Die Verhaltensweisen werden zusätzlich in den Lernprozess miteinbezogen. Der Roboter speichert die Positionen innerhalb der Route, in denen er eine (vorgegebene) Änderung der Verhaltensweise vorgenommen hat. Während der autonomen Navigation kann der Roboter anhand des Selbstlokalisierungsverfahrens ermitteln, wann er sich an einer dieser Positionen befindet und die entsprechende Verhaltensweise aktivieren.

Hauptaufgabe des hier vorgestellten Navigationsverfahrens ist eine robuste Selbstlokalisation im Routengraphen. Hierfür wird die Routenbeschreibung, die durch einen Lernprozess zur Verfügung gestellt wird, mit der aktuellen Route des Roboters verglichen. Durch die Routenbeschreibung besitzt der Roboter bei der Navigation topologisches und metrisches Vorwissen über die Umgebung, d.h. er kennt die Umgebung.

Diese Vorbedingung ist im Projekt Geometrischer Agent nicht gegeben. Der Agent verfügt über keine Karte der Umgebung, die metrische Informationen über Winkel oder Längen von Routensegmenten repräsentiert. Als Informationsressource dient ihm nur das unterspezifizierte räumliche Modell, das er aus der sprachlichen Routeninstruktion gewonnen hat.

Verwendung von Routeninstruktionen

In [Müller et al. 2000] wird ein alternatives Verfahren für die Generierung von Routenbeschreibungen und deren Verwendung vorgestellt. Auch hier dient der autonome Rollstuhl als Testroboter, allerdings ist die Testumgebung auf die Etage eines Universitätsgebäudes beschränkt.[8]

Das System ist aus mehreren Komponenten aufgebaut, die verschiedene für die Navigation notwendige Kompetenzen bereitstellen. Um von der Hardware des Rollstuhls zu abstrahieren, wurde die Schnittstelle SAM (*Sensor/Actuator Module*) zu den höhergelegenen Modulen geschaffen. Ein *Situation Detector* ermittelt mit Hilfe von odometrischen Verfahren und den aktuellen Sensordaten die aktuelle Situation und ist zudem für die Selbstlokalisation verantwortlich. Die Planung, d.h. die Auswahl einer geeigneten Verhaltensweise in der aktuellen Situation, nimmt der *Navigator* vor. Diese Verhaltensweisen werden schließlich durch das *Behaviours*-Modul ausgeführt, indem sie an das SAM weitergeleitet werden.

Die formale Routenbeschreibung, die dem Roboter mitgeteilt wird, ist bei der Repräsentation auf vier Aspekte beschränkt. Durch sie können der Startpunkt eines Pfades, die Ausführung einer Aktion an einer bestimmten Landmarke, zusätzliche Informationen über einen Pfad wie zu passierende Landmarken und der Zielpunkt beschrieben werden. Dies führt zu einer Formalisierung, in der eine Routenbeschreibung durch eine Sequenz von Tupeln spezifiziert wird. Ein Tupel hat dabei folgende Gestalt:

< [{ *controlmarks* } *router*] *reorientation* >

Eine *reorientation* beschreibt eine direktionale Anweisung wie „TurnLeft“, „EnterRightDoor“ oder „FollowCorridor“. Ein *router* beschreibt eine Landmarke, an der eine *reorientation* stattfinden kann. Der letzte *router* ist das Ziel der Route. *Controlmarks* dienen der Beschreibung von Landmarken, die auf längeren Wegstücken passiert werden. An ihrer Position findet keine *reorientation* statt. Landmarken, die während der Navigation erkannt werden, werden entweder als *router* oder als *controlmarks* interpretiert.

Folgendes Beispiel stellt eine initiale Routenbeschreibung dar ([Müller et al. 2000]):

< RightHandBend TurnRight >
< CorridorRight CorridorLeft TurnLeft >
< CorridorRight DeadEnd Stop >

[8] Durch das zweite Verfahren soll ein Krankenrollstuhl mit der Kompetenz der autonomen, instruktionsgestützten Navigation in einem Krankenhaus ausgestattet werden.

Die initiale Routenbeschreibung wird abgearbeitet, indem das erste Tupel T aus der Beschreibung herangezogen und bearbeitet wird. Der Roboter ist jedoch nicht in der Lage, Anweisungen wie „TurnRight“ direkt auszuführen, da er nicht weiss, wie weit die Drehung zu erfolgen hat. Die Anweisungen werden bei der Bearbeitung eines Tupels auf elementare Verhaltensweisen abgebildet. Diese werden vor dem Erreichen des *routers* aktiviert, und nach dem Passieren wieder deaktiviert. Die Konvertierung der initialen Routenbeschreibung führt zu folgender Repräsentation:[9]

< FollowRightWall RightHandBend >
< FollowLeftWall CorridorRight CorridorLeft >
< FollowLeftWall CorridorRight DeadEnd >
< Stop >

Bei der Verarbeitung eines Tupels T aus der initialen Routenbeschreibung können abhängig vom Inhalt von T drei Fälle unterschieden werden:

- ***T* besteht nur aus einer *reorientation*.** In diesem Fall besitzt die entsprechende Verhaltensweise ein intrinsisches Ende, und es ist dem Roboter möglich, diesen Endzustand zu erkennen (beispielsweise „Stop“ oder „TurnRound“). Der *navigator* wartet, bis das Behaviour-Modul das Ende der Ausführung der Verhaltensweise gemeldet hat. Dann wird T gelöscht, und das nächste Tupel herangezogen.

- ***T* beinhaltet *controlmarks*.** Die Landmarken, die durch die *controlmarks* beschrieben werden, müssen erst passiert werden, bevor der nächste *router* erreicht wird, an dem eine *reorientation* ausgeführt wird. Dass der Roboter an einer *controlmark* keine Änderung der aktuellen Verhaltensweise durchführt, wird durch ein *default*-Verhalten, das für verschiedene Typen von *controlmarks* bereitgestellt wird, sichergestellt. Wenn eine *controlmark* passiert wurde, wird diese aus dem aktuellen Tupel gelöscht.

- ***T* beinhaltet keine *controlmarks*** oder diese wurden alle schon passiert, so dass sie alle gelöscht worden sind. In diesem Fall muss nach dem *router* gesucht werden. Wird die Position des *routers* erreicht, kann die *reorientation* ausgeführt werden. Hiernach wird T gelöscht und das nächste Tupel herangezogen.

Wenn die Routenbeschreibung leer ist, wird der Durchlauf gestoppt, und es wird davon ausgegangen, dass der Roboter sein Ziel erreicht hat.

Da der Roboter kein spezifisches metrisches und topologisches Vorwissen über die Umgebung besitzt, muss die Selbstlokalistion durch Perzeptionsprozesse unterstützt werden. Während der Navigation vergleicht der *Navigator* hierfür die Informationen über die Landmarken, die durch den *Situation Detector* zur Verfügung gestellt werden, mit der Routenbeschreibung. Durch ein bildverabeitendes Verfahren (*landmark category detection*) kann der Roboter erkennen, ob in der aktuellen Situation bestimmte Landmarkenkategorien erfüllt

[9] In dieser Repräsentation haben die Tupel folgende Gestalt: < *behaviour* [{ *controlmarks* } *router*] >. Der *router* beschreibt hier, wann die nächste Verhaltensweise aktiviert, d.h. das nächste Tupel herangezogen werden kann. Die Umwandlung erfolgt nicht im voraus für die gesamte Routenbeschreibung, sondern wird erst bei der Abarbeitung eines Tupels der initialen Repräsentation vorgenommen.

werden, die für die Spezifizierungen von Routen verwendet werden. Landmarken werden als boolsche Vektoren über Landmarkenkategorien gebildet. Aufgrund der eingeschränkten Umgebung sind nur fünf Kategorien von Landmarken vorgesehen: „wall in front“, „corridor left“, „corridor right“, „door left“ und „door right“. Der Vektor beschreibt, welche Landmarkenkategorien in der aktuell perzipierten Situation erfüllt werden. Durch eine Kombination von Landmarkenkategorien lassen sich komplexe Landmarken bilden. Ermittelt der Erkennungsprozess z.B., dass die Kategorien „wall in front“ und „corridor left“ erfüllt sind, liegt die komplexe Landmarke „LeftHandBend“ vor.

Vergleich mit dem Geometrischen Agenten

Im ersten Szenario wird der Roboter mit einer Karte der Umgebung ausgestattet, die metrische Informationen beinhaltet (Länge und Winkel), und die er für seine Navigationsaufgabe nutzen kann. Der Geomterische Agent verfügt jedoch nicht über eine derartige Karte. Das geometrische und topologische Wissen, das er für die Naviagtion verwendet, stammt aus der Routenintsruktion und beinhaltet keine metrischen Informationen.

Im zweiten Szenario des *Robot Naviagtion*-Projekts wird der Roboter mit einer unterspezifizierten Routeninstruktion ausgestattet. Durch die Einschränkung der Umgebung des autonomen Rollstuhls auf eine Etage eines Gebäudes findet jedoch eine Modellierung statt, die von der Modellierung der Umgebung des Geometrischen Agenten verschieden ist. Dies äußert sich insbesondere in der Spezifizierung von Landmarken, wie auch in den Möglichkeiten, eine Route zu beschreiben.

Landmarken werden im zweiten Szenario durch einen Vektor dargestellt, dessen einzelne Elemente bestimmte Landmarkenkategorien darstellen. Mehrere positive Einträge im Vektor repräsentieren eine Verknüpfung von verschiedenen Landmarkenkategorien zu einer komplexen Landmarke. Diese Landmarkenkategorien sind jedoch spezifisch für die Testumgebung, und nicht auf andere Umgebungen übertragbar. Sie werden durch die Lage von bestimmten Objekten gebildet, die typischerweise in einer Etage eines Gebäudes vorkommen (Türen, Korridore und Wände).

Die Spezifikation einer räumlichen Konstellation mehrerer Landmarken, wie sie im Projekt *Geometrischer Agent* z.B. durch die Relation BETWEEN möglich ist, kann im *Robot Navigation*-Projekt nicht getätigt werden. Da aber die adäquate Repräsentation geometrischer und topologischer Konzepte wesentliches Ziel des Projekts *Geometrischer Agent* ist, sind die Verfahren für die Modellierung und Abarbeitung einer Routeninstruktion im *Robot Navigation*-Projekt nicht auf den Geometrischen Agenten übertragbar.

4.5.2 Instruction-Based Learning

Ziel des IBL-Projekts ist es, ein System für die instruktionsgestützte Navigation eines Roboters zu entwickeln. Der Begriff des *Lernens* im Titel des Projekts bezieht sich auf die Fähigkeit des Sytems, bereits verarbeitete Routeninstruktionen für zukünftige Routen wiederverwenden zu können ([Lauria et al. 2002a]).[10]

[10] Weitere Details zum IBL-Projekt finden sich in [Lauria et al. 2002b], [Kyriacou et al. 2002] und [Bugmann et al. 2001].

Die Methode des *Instruction-Based Learning* macht sich die Tatsache zu Nutze, dass menschliche Instruktoren bei der Verfassung einer natürlichsprachlichen Wegbeschreibungen typischerweise eine Dekomposition der Instruktion in einzelene Aktionseinheiten (*action chunks*) vornehmen. Diese *action chunks* kodieren einzelne Aktionen, Sequenzen von Aktionen, oder Aktionen, die an Bedingungen gekoppelt sind, oder in Schleifen (*loops*) auszuführen sind ([Lauria et al. 2002b]). Diese Aktionen werden durch *primitive Prozeduren* repräsentiert, die der Roboter ausführen kann, indem diese primitiven Prozeduren als Subprogramme realisiert sind (*robot executable procedures*). Aufgabe des Planungsprozesses des IBL-Systems ist es, die in der natürlichsprachlichen Wegbeschreibung kodierten Aktionen auf ausführbare Prozeduren abzubilden. Durch eine geeignete Zusammenstellung von primitiven Prozeduren, die der Roboter ausführen kann, wird eine Abbildung der natürlichsprachlichen *action chunks* auf die primitiven Prozeduren begünstigt ([Lauria et al. 2002a]).

Verarbeitung der Wegbeschreibung

Da natürlichsprachliche Wegbeschreibungen im allgemeinen unterspezifiziert sind ([Lauria et al. 2002a]), soll der Roboter in der Lage sein, die Konsistenz der Wegbeschreibung verifizieren zu können. Hierfür besitzt er eine symbolische Repräsentation seiner ausführbaren Aktionen, in der die Vorbedingungen und Effekte der Aktionen berücksichtigt sind. Ein Problem tritt dann auf, wenn der Roboter durch einen Inferenzprozess feststellt, dass zwei in der natürlichsprachlichen Wegbeschreibung aufeinanderfolgende Aktionen inkonsistent sind. Inkonsistenz liegt vor, wenn die Vorbedingungen der zweiten Aktion nicht erfüllt sind. Dies kann seine Ursache darin haben, dass die Effekte der ersten Aktion die Vorbedingung der zweiten negativ beeinflussen, oder die Vorbedingungen von Anfang an nicht erfüllt sind und auch zwischenzeitlich nicht erfüllt werden. In diesem Fall geht der Roboter davon aus, dass die Wegbeschreibung unterspezifiziert ist, d.h. eine Aktion nicht explizit genannt wurde, und fragt beim Instruktor nach. Der Roboter fragt auch nach, wenn eine Aktion auf zu abstrakter Ebene formuliert ist, und er nicht weiss, wie er diese abstrakte Anweisung auf primitive Prozeduren abbilden soll. Dieser Fall tritt auf, wenn durch eine Anweisung ein komplexes Routensegment beschrieben ist, welches dem Roboter unbekannt ist. So kann z.B. die Aufforderung „Gehe zur Post!“(*„Go to the post office!“*) in der Routeninstruktion dazu führen, dass der Roboter feststellt, dass er keine geeignete Route zur Post kennt und den Instruktor auffordert, die Route zur Post näher zu spezifizieren: „Wie komme ich zur Post?“(*„How do I get to the post office?“*) ([Lauria et al. 2002b]). Ist die Antwort hinreichend spezifisch, gibt sich der Roboter damit zufrieden. Eine Instruktion ist hinreichend spezifisch, wenn die einzelnen Anweisungen direkt in primitive Prozeduren umgewandelt werden können, oder wenn dem Roboter die entsprechende Teilroute zu einem früheren Zeitpunkt auf spezifischer Ebene mitgeteilt wurde (er diese Route *gelernt* hat).

Um die oben genannten Aufgaben zu erfüllen, müssen durch das IBL-System eine geeignete symbolische Repräsentation der natürlichsprachlichen Wegbeschreibung und eine adäquate Menge von primitiven Prozeduren bereitgestellt werden. Weiter im Fokus des Projekts liegen die Schaffung eines Mechanismus zur Abbildung der natürlichsprachlichen Wegbeschreibung auf die primitiven Prozeduren des Roboters, sowie die Realisierung eines erfolgreich navigierenden Roboters in einer Testumgebung. Das IBL-System besteht aus zwei funktionalen Einheiten (*Dialog Manager* und *Robot Manager*), in denen die einzelnen Verarbeitungsschritte

der Umwandlung der natürlichen Wegbeschreibung in ausführbare Prozeduren vorgenommen werden. Angeschlossen an das System ist der Robotor, der bei seiner Navigation durch die Testumgebung durch den Robot Manager gesteuert wird.

Der Dialog Manager ist für die interaktive Schnittstelle zwischen dem Instruktor und dem System verantwortlich. Das Modul beinhaltet Submodule für die Spracherkennung und -verarbeitung sowie für die Erzeugung von sprachlichen Ausdrücken. Das System kann Fragen an den Instruktor richten, wenn der Robot Manager eine entsprechende Anfrage an den Dialog Manager gestellt hat. Die natürlichsprachliche Routeninstruktion wird durch mehrere Verarbeitungsprozesse in eine symbolische Repräsentation, die mit Hilfe eines DRS-Systems (*Discourse Representation Structure*) formuliert wird, umgewandelt. Diese Repräsentation dient dem Robot Manager als Eingabe.

Der Robot Manager ist für die Auswahl der ausführbaren Prozeduren, die Ansteuerung des Roboters sowie für den Lernprozess zuständig. Hierfür wandelt er die DRS-Repräsentation, die durch den Dialog Manager bereitgestellt wird, in die interne Sprache des Robot Managers um, welche Funktionen für die Ausführung von Aktionen und für Lernprozesse beinhaltet. Eine erfolgreiche Umwandlung hat entweder die Ausführung der entsprechenden Aktionen zur Folge, oder es wurde durch die Instruktion eine neue Route erklärt, die einen Lernprozess anstösst. Konnte keine erfolgreiche Umwandlung der DRS-Repräsentation in die entsprechenden Prozeduren vorgenommen werden, liegt eines der oben genannten Probleme vor und der Robot Manager sendet eine Anfrage an den Dialog Manager. Dieser gibt die Anfrage (nach Umwandlung in einen natürlichsprachlichen Ausdruck) an den Instruktor weiter und verarbeitet anschließend die Antwort des Instruktors.

Bei der Zusammenstellung der Menge an primitiven Prozeduren des Roboters wurde untersucht, welche Typen von Anweisungen typischerweise in Wegbeschreibungen auftreten. Die Analyse des Testkorpus hat eine Kategorisierung in 13 verschiedene *action chunks* ergeben, für die entsprechende primitive Prozeduren zur Verfügung gestellt werden. In der Menge sind z.B. primitive Prozeduren für Bewegungen (follow_road(), turn(), rotate()) enthalten, oder die Anweisung, einer früher erklärten Route zu folgen (go()).[11] Durch verschiedene Parametrisierungen lässt sich eine primitive Prozedur für verschiedene Anweisungstypen verwenden. So lässt sich die Prozedur `turn()` durch die Verwendung entsprechender Argumente z.B. für die Anweisung „Geh nach links“ aber auch „Geh die zweite links“, oder „Geh hinter der Post links“ nutzen. Aufgabe des Robot Managers ist es, die symbolische Repräsentation einer Anweisung wie „Geh die zweite hinter der Post rechts“ auf die entsprechende primitive Prozedur abzubilden und diese geeignet zu parametrisieren (in diesem Fall turn(second, right, after, post office)).

Der Lernprozess

Namensgebend für das Projekt ist das Vermögen des Sytems, durch die Beschreibung eines Routensegments auf spezifischer Ebene dieses Routensegment mit den genannten primitiven Prozeduren zu verknüpfen, d.h. durch die Beschreibung einen Teil der Route zu *lernen*. Mit Hilfe einer *Procedure Specification Language* (PSL) können neue Prozeduren spezifiziert

[11] In den Klammern der Prozeduren werden die Parameter der Prozeduren angegeben. Aus Gründen der Übersichtlichkeit habe ich die Parameter(-typen) weggelassen.

werden. Als initiale Prozeduren besitzt das System die Menge von 13 primitiven Prozeduren. Werden durch die Beschreibung einer Route die hierfür primitiven Prozeduren genannt, wird eine neue Prozedur erzeugt, die die primitiven Prozeduren beinhaltet. Dadurch kann der Roboter einen Weg (z.B. zur Post) erlernen, so dass in einer späteren Routeninstruktion die Anweisung „Geh zur Post“ ausreichend spezifisch ist, d.h. erfolgreich mit einer Prozedur verknüpft werden kann. Der Lernprozess des Systems findet auf der Stufe der symbolischen Repräsentation statt (*symbolic learning*). Dies hat den Vorteil, dass für die Generierung einer neuen Prozedur nur die symbolischen Referenzen auf die Prozeduren und nicht die Prozeduren selbst (die aus Programmcode bestehen) kopiert werden müssen.

Abarbeitung der primitiven Prozeduren durch den Roboter

Während der Navigation in der Testumgebung werden die primitiven Prozeduren auf elementare Perzeptions- und Bewegungsaktionen des Roboters abgebildet ([Kyriacou et al. 2002]). Die Testumgebung des Systems ist ein idealisiertes Miniaturmodell einer Stadt (mit den Maßen 1,20m x 1,70m). Der Roboter nimmt eine Fläche von 8cm x 8cm ein und besitzt eine CCD-Kamera, über die er seine Umgebung wahrnehmen kann, sowie einen Funksender, über den er die Videosignale an den Rechner weiterleitet, auf dem das IBL-System implementiert ist, und Steuersignale von diesem entgegennimmt. Um die Objekterkennung zu vereinfachen, entsprechen die Straßen des Modells einem vorgegebenen Schema und sind an den Seiten weiss markiert. Als Landmarken dienen hauptsächlich Straßensegmente, die bestimmte Eigenschaften aufweisen, durch die der Roboter feststellen kann, wann er eine bestimmte Prozedur auszuführen hat. So muss der Roboter z.B. erkennen können, an welcher Stelle er rechts abbiegen muss, um die Anweisung „Nimm die zweite rechts!“ abzuarbeiten. Hierfür vergleicht er den während der Navigation wahrgenommenen Teil der Straße mit vorgegebenen Schablonen, die bestimmte Aspekte von Straßensegmenten repräsentieren (ob es sich z.B. um ein gerades Wegstück, eine Kurve oder eine Kreuzung handelt). Wenn ein zweites Mal ein Straßenabschnitt erfolgreich mit einer Schablone, die eine Rechtsabbiegung beinhaltet, verglichen wurde, kann der Agent die Aktion, rechts abzubiegen, ausführen. Zusätzliche Landmarken (wie die Post) werden durch farbige Markierungen modelliert.

Aus Sicht des Moduls, das für die Ausführung der primitiven Prozeduren zuständig ist, handelt es sich bei diesen um Anweisungen auf einer abstrakten Ebene (*high-level task specifications*), die dynamisch während der Navigation abgearbeitet werden ([Kyriacou et al. 2002]). Hierfür führt der Roboter eine Reihe von Aktionssequenzen (*chain of search-and-act sensory-motor loops*) durch, die jeweils einer primitiven Prozedur entsprechen. Jede dieser Aktionssequenz besteht aus einem Programmabschnitt, der verschiedene Subroutinen aufruft. Die Abbildung 4.2 zeigt den Algorithmus für die primitive Prozedur „Nimm die n-te links/rechts“, der in [Kyriacou et al. 2002] als Beispiel vorgestellt wird.

Im ersten Schritt (`Define set of road features`) wird die Menge von Schablonen ausgewählt, die für den Erkennungsprozess in Frage kommen. Dies sind in diesem Fall gerade oder leicht gebogene Wegstücke für die Geradeausfahrt und Schablonen, in denen ein Wegstück nach links (resp. rechts) abbiegt. Die letztgenannten Schablonen dienen für die Identifizierung von Abbiegungen, die zuerst gezählt werden (wobei weiter geradeausgefahren wird), um dann an der n-ten Kreuzung abzubiegen.

Durch die zweite Subroutine (`Update internal map & localize robot`) wird die inter-

```
Define set of road features
Loop:
{
  Capture and process road image
  Update internal map & localize robot
  Find best matching template in the map
  Execute procedure (e.g. robot motion) associated with the
   winning template
}
```

Abbildung 4.2: Beispiel für einen Algorithmus für die Abarbeitung einer primitiven Prozedur.

ne Karte (*short-lived map*), die die perzipierte Umgebung repräsentiert, aktualisiert, und die Position des Roboters darin bestimmt. Diese interne Karte beinhaltet durch eine Zwischenspeicherung und Weiterverarbeitung der perzipierten Bilder auch Objekte, die nicht im aktuellen Sichtfeld des Roboters liegen (da sie hinter ihm oder im toten Winkel der CCD-Kamera liegen).

Der Roboter nutzt ein Repertoire von zwölf Schablonen, die verschiedene Aspekte der Straßen repräsentieren (*road-feature templates*), um eine Verbindung zwischen den Objekten der Routeninstruktion (den Parametern der primitiven Prozeduren) und der mentalen Karte herstellen zu können. Durch die Subroutine `Find best matching template in the map` wird ein Erkennungsprozess (*template matching*) angestoßen, der ermittelt, wie gut ein perzipiertes Wegstück mit dem Argument der aktuellen primitiven Prozedur übereinstimmt. So gibt es z.B. sechs Schablonen, die für den Erkennungsprozess für die Anweisung „geh nach rechts“ in Frage kommen, da in ihnen ein Wegstück nach rechts abzweigt.

Passt eine der sechs Schablonen zu der mentalen Karte, kann der Roboter durch die Subroutine `Execute procedure (e.g. robot motion) associated with the winning template` die Aktion ausführen (indem er an der entsprechenden Stelle eine Drehung vollführt und dann weiterfährt).

Vergleich mit dem Geometrischen Agenten

Die gemeinsame Aufgabenstellung der beiden Projekte hat zur Folge, dass mehrere Gemeinsamkeiten vorhanden sind. Durch beide Verfahren müssen die in Abschnitt 1.2 genannten Kompetenzen, die für die Navigation notwendig sind, bereitgestellt werden. So finden sich sowohl im Geometrischen Agenten, als auch im IBL-System eine Repräsentation der perzipierten Umgebung (Perzeptionsmodell und *short-lived map*). Da beide Verfahren eine natürlichsprachliche Routeninstruktion als Ressource verwenden, muss diese durch ein Modell repräsentiert werden (Instruktionsmodell und DRS-Repräsentation). Die in der Routeninstruktion beschriebenen Aktionen werden auf eine Sequenz von abstrakten Aktionen abgebildet (primitive Prozeduren und Instruktionsanweisungen), die während der Navigation in konkrete Aktionen des Roboters umgewandelt werden. Auch der Algorithmus zur Abarbeitung der abstrakten Aktionen ist vergleichbar aufgebaut, um eine dynamische Verarbeitung

zu ermöglichen. Der oben beschriebene Algorithmus besitzt im wesentlichen dieselbe Struktur wie der Algorithmus für die Ablaufsteuerung, der in Abschnitt 5.8.1 vorgestellt wird.

Die Art und Weise der Modellierung der Umgebung und der Routeninstruktion, sowie der resultierenden Repräsentationen, wie auch die Realisierung der Verarbeitungsprozesse weisen jedoch deutliche Unterschiede auf, die im jeweiligen Fokus des Projekts begründet sind.

Ziel des IBL-Projekts ist es, durch einen Lernprozess auf symbolischer Ebene die Generierung neuer Prozeduren zu schaffen, die aus Referenzen auf primitive Prozeduren (oder schon vorher gelernten Prozeduren) bestehen. Ziel des Projekts *Geometrischer Agent* ist es, eine Axiomatisierung räumlicher Konzepte zur Verfügung zu stellen. Hierfür wird ein Formalismus geschaffen, in dem durch eine Repräsentationsprache (CRIL) adäquate Modelle sowohl für natürlichsprachliche Routeninstruktionen als auch für Modelle aus der Perzeption formuliert werden können. Dem Aspekt, dass die Modellierung geometrischer Aspekte im Projekt des Geometrischen Agenten eine besondere Rolle spielt, wird dadurch Rechnung getragen, dass die Routeninstruktion auf den Instruktionsgraphen, der die geometrischen Eigenschaften der Objekte aus der Routeninstruktion repräsentiert, abgebildet wird. Die imperativen Anteile der Routeninstruktion werden getrennt in einem Aktionsplan gespeichert. Im IBL-Projekt hingegen werden die *action chunks* der natürlichsprachlichen Routeninstruktion direkt auf eine Sequenz von Prozeduren abgebildet. Diese Herangehensweise hat zur Folge, dass die Möglichkeit der Repräsentation geometrischer Konzepte beschränkt ist, da die Spezifikation von Objekten wie z.B. Landmarken nur im Kontext von Prozeduren erfolgen kann ([Lauria et al. 2002a]). Die Instruktionen verweisen hauptsächlich auf Eigenschaften der Route, die in Aspekten des zu begehenden Weges bestehen: So finden sich im Korpus häufig Formulierungen wie „gehe die n-te links/rechts“, „gehe geradeaus bis ...„, oder „überquere die Straße“. Aussagen, die die Lage eines Objekts beschreiben, werden auf die Prozedur `located()` abgebildet (die in etwa der Instruktionanweisung `BE_AT()` des Geometrischen Agenten entspricht). Eine geometrische Spezifikation wie sie z.B. durch die Aussage „zwischen Haus B und Haus C befindet sich ein Weg“ geleistet wird, ist im IBL-System nicht modellierbar.

Durch die Einschränkung bei der Repräsentation geometrischer Konzepte hat sich eine spezifische Modellierung der Umgebung im IBL-Projekt ergeben, in der andere Landmarken als Straßensegmente eine untergeordnete Rolle spielen. Der Erkennungsprozess (*recognition*), der im Geometrischen Agenten durch den Prozess der Auflösung von Koreferenzen bereitgestellt wird, besteht im IBL-Projekt im Wesentlichen in dem Prozess, der die interne Karte der Perzeption mit den infrage kommenden Schablonen vergleicht (*template matching*). Um eine hinreichend hohe Robustheit zu ereichen, sind die Strassen in der Testumgebung aus einzelnen Elementen zusammengesetzt, die eine hohe Ähnlichkeit mit den Schablonen besitzen und somit hinreichenden Erfolg beim Erkennungsprozess gewährleisten.

4.6 Planen im Geometrischen Agenten

Ein klassischer Planer verwendet das Planungsproblem als Vorgabe, das eine Spezifikation des Initialzustands, der ausführbaren Aktionen und des Zielzustandes repräsentiert, um in einem abgeschlossenen Planungsprozess den Plan von zu tätigenden Aktionen zu generieren. Dieser Plan wird dann in der Ausführungsphase strikt ausgeführt.

Für den Geometrischen Agent hingegen kann weder ein derartiges Planungsproblem for-

muliert werden, noch kann ein Plan von konkreten Aktionen im voraus erzeugt werden, der einfach strikt in der Navigationsphase ausgeführt werden kann. Allerdings besitzt der Geometrische Agent einen Aktionsplan als abstrakte Vorgabe, an dem er sich während der Navigationsphase bei der Auswahl der primitiven Aktionen orientieren kann.

An der Generierung des Aktionsplans sind mehrere Prozesse beteiligt. Den ersten Schritt macht der Verfasser der natürlichsprachlichen Routeninstruktion, indem er neben der geometrischen Beschreibung von Teilen der Umgebung eine Folge von auszuführenden Handlungen nennt. Diese explizit formulierten Handlungsanweisungen werden im Geometrischen Agenten während der Instruktionsphase aus der Routeninstruktion extrahiert und in eine formale Repräsentation, den Aktionsplan, umgewandelt.

4.6.1 Generierung und Nutzung des Aktionsplans

Generierung und Nutzung des Aktionsplans finden in zwei zeitlich getrennten Phasen statt: in der Instruktions- und Navigationsphase des Geometrischen Agenten. Wie in der Abbildung 4.3 schematisch dargestellt, wird der Aktionsplan während der Instruktionsphase erzeugt. Der Geometrische Agent orientiert sich während der Navigationphase am Aktionsplan, um im Prozes der lokalen Planung die primitiven Aktionen auszuwählen, die er dann in der simulierten Umgebung ausführt.

- **Generierung des Aktionsplans:** In der Instruktionsphase werden durch die Verarbeitung der Routeninstruktion die imperativen Anteile der sprachlichen Wegbeschreibung extrahiert, und es wird ein statischer[12] Aktionsplan erzeugt. Dieser Aktionsplan ist abstrakt, da die in ihm enthaltenen Aktionen bzw. Instruktionsanweisungen unterspezifiziert sind. Sie sind daher nicht direkt ausführbar, sondern dienen dem Agenten als Orientierung bei der Auswahl der ausführbaren primitiven Aktionen.

- **Abarbeitung des Aktionsplans durch lokale Planung:** Während der Navigationsphase findet eine an die Situation angepasste lokale Planung statt. Der Agent bewertet mit Hilfe von Informationen aus der Perzeption und dem Instruktionsgraphen und unter Berücksichtigung pragmatischer Annahmen die aktuelle Situation und aktualisiert seinen internen Zustand. Anhand des internen Zustands trifft er eine Auswahl von primitiven Aktionen, die einer Instruktionsanweisung aus dem Aktionsplan entsprechen. Diese Aktionen bilden die lokale Aktionssequenz und werden vom Agenten ausgeführt. In der schematischen Darstellung in der Abb. 4.3 wird z.B. die Instruktionsanweisung $!\mathsf{GO}(\mathsf{w}_2)$ auf die lokale Aktionssequenz, die die primitiven Aktionen A_1 und A_2 beinhaltet, abgebildet.

Im Geometrischen Agenten findet somit ein Verfahren zur Auswahl von primitiven Aktionen statt, das sich von den typischen Verfahren klassischer Planer unterscheidet. Wenn man aber den Begriff des *Planens* weiter fasst, und darunter jegliche Prozesse verstehen will, die bei der Auswahl der Aktionen eines rationalen Agenten beteiligt sind, kann auch der in Kapitel 5 vorgestellte Prozess für die Abarbeitung des Aktionsplans als ein Planungsprozess verstanden werden.

[12] *Statisch* bedeutet in diesem Kontext, dass der Plan nach der Generierung nicht mehr verändert wird.

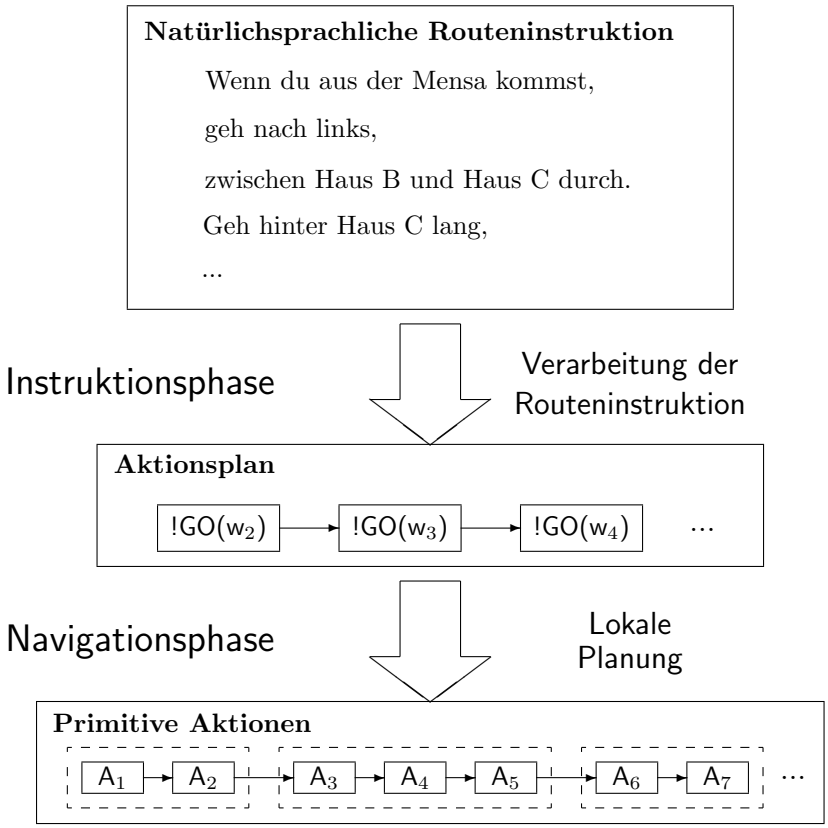


Abbildung 4.3: Die Prozesse zur Generierung und Nutzung des Aktionsplans im Geometrischen Agenten. Während der Instruktionsphase werden die imperativen Anteile aus der natürlichsprachlichen Routeninstruktion extrahiert und als Instruktionsanweisungen im Aktionsplan gespeichert. Eine Instruktionsanweisung wird während der Navigationsphase in einem lokalen Planungsprozess auf eine lokale Aktionssequenz abgebildet, die sich aus primitiven Aktionen des Agenten zusammensetzt.

4.6.2 Planungskonzepte im Geometrischen Agenten

Hierarchische Abstraktion

Im Geometrischen Agenten sind drei Abstraktionsstufen erkennbar, in denen Pläne formuliert bzw. als Lösung generiert werden. Auf höchster Stufe steht der Plan, der direkt durch die Aufgabenstellung gegeben ist. Beispielsweise kann für die in Abschnitt 3.1 gestellte Frage „Wie komme ich von der Mensa zu Haus E?“ direkt der abstrakte (und triviale) Plan ‚Gehe von der Mensa zu Haus E.‘ generiert werden. Den ersten Schritt bei der Konkretisierung des abstrakten Plans leistet der Instrukteur, der durch die natürlichsprachliche Formulierung der Wegbeschreibung eine Sequenz von auszuführenden Handlungen auf einer niedrigeren Abstraktionsstufe vorgibt. Die Handlungsanweisungen können aufgrund der Aspekte Indexikalität, Projektion und Reflexivität kompakt bzw. abstrakt ausfallen.[13] Bei der Abbildung

[13]Siehe Abschnitt 4.4.4 *Die Plan-as-communication-Sichtweise.*

der natürlichsprachlichen Routeninstruktion auf das formale Instruktionsmodell werden diese Handlungsanweisungen extrahiert und als Instruktionsanweisungen im Aktionsplan gespeichert. Die Instruktionsanweisungen können aus der Sicht des übergeordneten Ziels als Aktionen, die eine Lösung der Navigationsaufgabe darstellen, interpretiert werden. Aus Sicht des Aktionsmoduls stellen die Instruktionsanweisungen jedoch Teilziele auf einer mittleren Stufe dar, da sie hinsichtlich einer Ausführung unterspezifiziert sind. Die Instruktionsanweisungen werden durch den Prozess der lokalen Planung konkretisiert, indem eine Auswahl von primitiven Aktionen getroffen wird, die dann vom Geometrischen Agenten in der simulierten Umgebung ausgeführt werden können.

Ein wesentlicher Unterschied zum hierarchischen Planen, wie es in Abschnitt 4.3 beschrieben ist, besteht in der Art und Weise, in der die Konkretisierung des abstrakten Plans vorgenommen wird. Es gibt im Geometrischen Agenten keine statischen Regeln zur Planverfeinerung[14], die für jede mögliche Situation im voraus einen korrekten Plan konkreter Aktionen erzeugen. Aufgrund der Unsicherheit erfolgt die Abbildung auf primitive Aktionen dynamisch im Kontext der aktuellen Situation.

Ausführungsüberwachung

Der Geometrische Agent orientiert sich an seinem internen Zustand, um zu entscheiden, welche primitiven Aktionen in der aktuellen Situation auszuführen sind. Das Prinzip der Ausführungsüberwachung ist bei der lokalen Planung unvermeidlich, da die Perzeption der Umgebung die Informationen über die aktuelle Situation liefert, durch die der Agent seinen internen Zustand aktualisert.

Die Aktualisierung des internen Zustands spielt auch eine wichtige Rolle bei der Entscheidung, ob eine Instruktionsanweisung aus dem Aktionsplan als erfolgreich abgearbeitet werden kann. Hierfür wird geprüft, ob die Vorbedingung der folgenden Anweisung erfüllt ist.[15]

Desweiteren muss der Geometrische Agent vor der Ausführung einer Bewegungsaktion das Argument der Instruktionsanweisung mit einem potentiellen Pfad aus der Umgebung identifizieren. Hier erfüllt die Perzeption vor Ausführung einer Aktion nicht nur die Aufgabe, Vorbedingungen zu überprüfen; die Prozesse, die sich der Perzeption anschließen, bestimmen auch den wesentlichen Parameter der Bewegungsaktion.

Allerdings wird durch ein negatives Ergebnis der Überprüfung der Vorbedingungen der auszuführenden Aktion keine Revision oder Reparatur eines Plans wie bei dem Verfahren der Neuplanung vorgenommen, da der Agent über keinen im voraus erzeugten detallierten Plan verfügt. Stattdessen werden weitere primitive Aktionen ausgeführt, bis die aktuelle Instruktionsanweisung als erfolgreich abgearbeitet interpretiert werden kann. Das Verfahren, das diese primitiven Aktionen auswählt und deren Ausführung anstößt, wird in Kapitel 5 im Detail vorgestellt.

Kontinuierliches Planen

Das Verfahren der lokalen Planung kann als ein kontinuierlicher Planungsprozess verstanden werden. Während der Navigationsphase orientiert sich der Agent am Instruktionsmodell, um

[14] In [Russell und Norvig 2003] *Plandekompositionsregeln* genannt.
[15] Siehe Abschnitt 5.1.4 *Die Vorbedingung von Instruktionsanweisungen.*

eine Abarbeitung des Aktionsplans vorzunehmen. Hierfür wird zusätzlich das Wissen verwendet, welches durch die Ergebnisse der Perzeptionsprozesse bereitgestellt wird. In der Navigationsphase kann keine strikte Trennung zwischen den Phasen der lokalen Aktionsplanung und der Aktionsausführung (inklusive der Ausführung der Perzeptionsaktionen) vorgenommen werden. Diese beiden Prozesse wechseln einander ab, bedingen sich gegenseitig, und werden durch die Ablaufsteuerung des Aktionsmoduls kontrolliert.

Kapitel 5

Das Aktionsmodul

Das Aktionsmodul koordiniert durch eine Ablaufsteuerung das Verhalten des Geometrischen Agenten in der Navigationsphase. Das Verhalten des Agenten äußert sich durch die Ausführung von primitiven Aktionen in der simulierten Umgebung. Diese primitiven Aktionen werden in einem Prozess der lokalen Planung, der sich an dem internen Zustand des Agenten orientiert, ausgewählt. Der interne Zustand repräsentiert das Wissen und die Annahmen des Agenten über die aktuelle Situation. Um die Information aus der Routeninstruktion bei der lokalen Planung adäquat in der aktuellen Situation nutzen zu können, ist der Agent auf Hilfsprozesse angewiesen, die den aktuellen internen Zustand des Agenten ermitteln bzw. manipulieren. Diese Hilfsprozesse dienen der Verabeitung der Perzeption, der Selbstlokalistion im Instruktionsmodell und treffen mittels pragmatischer Annahmen Entscheidungen, die das Verhalten des Agenten beeinflussen (s.u.). Zusätzlich zur Auswahl und Ausführung der primitiven Aktionen ist das Aktionsmodul für die Koordinierung dieser Hilfsprozesse verantwortlich.

Die Abbildung 5.1 stellt die wesentlichen Komponenten des Geometrischen Agenten dar, die an der Verarbeitung während der Navigationsphase beteiligt sind. Zu unterscheiden sind die Objekte, die den internen Zustand des Agenten beschreiben, und die Prozesse, die den internen Zustand manipulieren. Der interne Zustand wird durch das Umgebungsmodell (welches das I-Netz, das P-Netz, die Verknüpfung der Netze sowie die Position des Agenten beinhaltet) und den Aktionsplan bestimmt.

5.1 Repräsentation des internen Zustands des Agenten

Das Verhalten des Geometrischen Agenten richtet sich nach seinem internen Zustand. Dieser repräsentiert das Wissen (bzw. die Annahmen) des Agenten über die aktuelle Situation und setzt sich aus dem Umgebungsmodell des Agenten und dem Aktionsplan, in dem die aktuell relevanten Instruktionsanweisungen markiert sind, zusammen.

5.1.1 Das Umgebungsmodell des Agenten

Das Umgebungsmodell repräsentiert die räumlichen Aspekte des aktuellen internen Zustands des Agenten. Es beinhaltet den Instruktionsgraphen, den Perzeptionsgraphen, die Verknüpfung dieser Graphen und die aktuelle Position des Agenten.

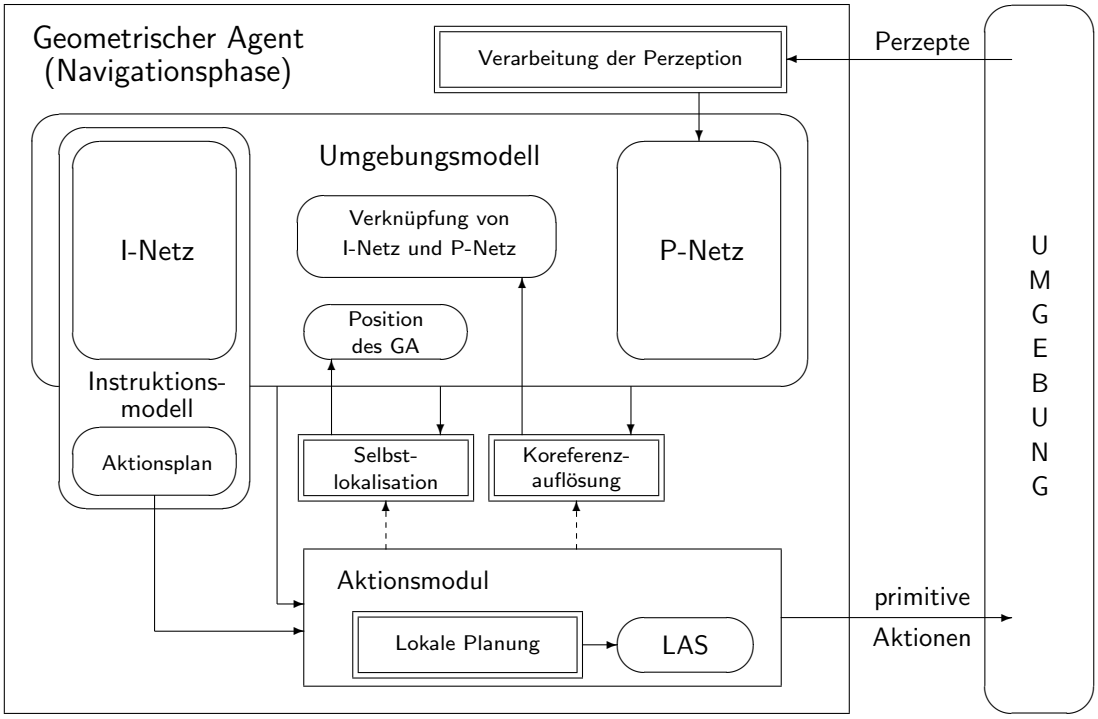


Abbildung 5.1: Die Architektur des Geometrischen Agenten in der Navigationsphase. Das Agentenprogramm aus Abb. 2.1 ist hier ersetzt worden durch die internen Repräsentationen des Geometrischen Agenten (ovale Kästen) sowie die Prozesse der Navigationsphase (hervorgehobene eckige Kästen).

- **Der Instruktionsgraph**: Der Instruktionsgraph (I-Netz) stellt eine deklarative Beschreibung des räumlichen Wissens der Routeninstruktion dar. Als Argumente der Instruktionsanweisungen des Aktionsplans fungieren Referenzobjekte des Instruktionsgraphen. Diese Objekte werden durch Attribute und geometrische bzw. topologische Relationen zu anderen Objekten spezifiziert.

- **Der Perzeptionsgraph**: Der Perzeptionsgraph (P-Netz) beinhaltet die aktuell wahrgenommenen Objekte des Agenten als Referenzobjekte, sowie die wahrgenommen räumlichen Eigenschaften dieser Objekte als geometrische oder topologische Relationen. Durch Ausführung einer Perzeptionsaktion und den Prozess der Verarbeitung der Perzeption wird der Perzeptionsgraph des Agenten aktualisiert.

- **Verknüpfung von Perzeption und Instruktion**: Durch den Prozess der Koreferenzauflösung wird eine Verknüpfung zwischen dem I-Netz und dem P-Netz vorgenommen, indem Koreferenzbeziehungen zwischen Objekten des I-Netz und Objekten des P-Netz ermittelt werden (vgl. Abschnitt 3.5 *Prozesse in der Navigationsphase*).

  Die Prozesse, die die Verarbeitung der Perzeption sowie die Koreferenzauflösung vornehmen, sind im Projekt schon prototypisch ausgearbeitet worden. Grundlegende Prinzipien und Funktionsweise des Koreferenzierungsverfahrens werden in [Helwich 2003] vorgestellt.

- **Position des Geometrischen Agenten**: Durch seine aktuelle Position innerhalb des Umgebungsmodells kann der Agent feststellen, an welcher Stelle er sich innerhalb des Instruktionsgraphen befindet, und welcher Teil des Instruktionsgraphen für ihn momentan relevant ist. Die Position des Agenten wird durch den Prozess der Selbstlokalisation ermittelt, der sich pragmatischer Annahmen bedient.

5.1.2 Der Aktionsplan und die aktuell relevanten Instruktionsanweisungen

Der Aktionsplan enthält die Informationen über die vom Agenten auszuführenden Handlungen, die in der Routeninstruktion explizit genannt werden. Er setzt sich aus Instruktionsanweisungen zusammen, die die Aktionen des Agenten auf abstrakter Ebene beschreiben und bezüglich einer Ausführung in der simulierten Umgebung unterspezifiziert sind. Im Prozess der lokalen Planung werden die Instruktionsanweisungen auf primitive Aktionen des Agenten abgebildet. Zwei Instruktionsanweisungen spielen im Prozess der lokalen Planung eine besondere Rolle:

- **Die aktuelle Instruktionsanweisung**: Welche Aktion als nächstes in der aktuellen Situation auszuführen ist, ermittelt der Agent im Prozess der lokalen Planung mit Hilfe der aktuellen Instruktionsanweisung. Sie schreibt durch den Anweisungstyp die Art der auszuführenden Aktion vor. Das Argument der aktuellen Anweisung bestimmt den wesentlichen Parameter der Aktion, in dem es spezifiziert, welche Objekte aus der Perzeption koreferenziert werden können und somit als Argumente für die Aktion in Frage kommen.

- **Die folgende Instruktionsanweisung**: Eine weitere Instruktionsanweisung ist in der aktuellen Situation ebenfalls von Relevanz und bestimmt den Zustand des Agenten mit: Um festzustellen, ob die aktuelle Instruktionsanweisung als erfolgreich bearbeitet interpretiert werden kann, wird durch das Aktionsmodul geprüft, ob die Vorbedingung der folgenden Instruktionsanweisung erfüllt ist (siehe Abschnitt 5.1.4). Ist dies der Fall, kann die aktuelle Instruktionsanweisung als erfolgreich abgearbeitet interpretiert werden und die nächste Anweisung herangezogen werden, die hierbei den Status der aktuellen Instruktionsanweisung erhält.

5.1.3 Anweisungstypen

CRIL unterscheidet vier Typen von imperativen Operatoren, durch die verschiedene Instruktionsanweisungen gebildet werden (Siehe Tabelle 5.1). Die Anweisungstypen GO und CH_ORIENT entsprechen Bewegungsanweisungen und haben als Argumenttypen Pfade. Der Anweisungstyp VIEW beschreibt die Handlung, nach einem Objekt Ausschau zu halten. Wie eine Anweisung vom Typ BE_AT abzuarbeiten ist, kann erst im Kontext der aktuellen Situation ermittelt werden, da diese Anweisung als die Aufforderung „Versichere Dich in der Region r zu sein, anderenfalls bewege Dich zu r.“ interpretiert wird.

Als Argumente der Operatoren fungieren Referenzobjekte des Instruktionsgraphen. Die geometrische Spezifikation dieser Objekte, die in der Routeninstruktion beschrieben wird, wird durch die Struktur des Instruktionsgraphen repräsentiert. Das Argument einer Bewegungsanweisung stellt somit den wesentlichen Parameter bei der Abbildung der Instruktions-

Typen von Instruktionsanweisungen	**Interpretation**
GO(w)	Bewege dich auf dem Pfad, der durch w spezifiziert ist.
CH_ORIENT(w)	Drehe dich, so dass deine Orientierung mit der Richtung des Pfades, der durch w spezifiziert ist, übereinstimmt.
BE_AT(r)	Versichere dich, in der Region r zu sein. Falls du nicht in der Region r bist, gehe dort hin.
VIEW(o)	Siehe nach dem Objekt o.

Tabelle 5.1: Typen von Instruktionsanweisungen, aus denen sich der Aktionsplan zusammensetzt.

anweisung auf die entsprechenden primitiven Aktionen dar, da es bestimmt, auf welchem Pfad die Bewegungsaktion ausgeführt wird.

5.1.4 Die Vorbedingung von Instruktionsanweisungen

Instruktionsanweisungen repräsentieren Aktionen des Agenten auf einer abstrakten Stufe, d.h. die beschriebenen Aktionen sind aufgrund des Problems der Unterspezifiziertheit nicht direkt in der Umgebung ausführbar, sondern müssen auf ausführbare Aktionen des Agenten abgebildet werden.

Aktionen werden in der KI klassischerweise durch ihre Vorbedingungen und Effekte spezifiziert.[1] Eine vollständige Spezifikation von Instruktionsanweisungen in diesem Sinne müsste alle Effekte berücksichtigen, die den durch die Abarbeitung der Anweisung resultierenden Folgezustand beeinflussen. Dies ist jedoch nicht möglich, da aufgrund der Unterspezifiziertheit des Aktionsplans nicht vorausgesagt werden kann, auf welche primitiven Aktionen eine Instruktionsanweisung abgebildet wird, und welche Zustände bei der Ausführung der primitiven Aktionen durchlaufen werden. Somit ist es auch nicht möglich, im voraus einen Plan von primitiven Aktionen zu erstellen, der dem Aktionsplan auf einer konkreten Stufe entspricht. Der Agent ist während des Prozesses der lokalen Planung auf die Perzeption der Umgebung angewiesen, um seinen internen Folgezustand nach Ausführung einer primitiven Aktion (und nach erfolgreicher Abarbeitung einer abstrakten Instruktionsanweisung, s.u.) zu ermitteln.

Eine Vorbedingung, die alle Typen von Instruktionsanweisungen gemein haben, spielt jedoch eine wichtige Rolle bei der lokalen Planung. Bei der Formulierung der sprachlichen Routeninstruktion zerlegt der Instruktor die Route in Routensegmente und verknüpft diese Segmente mit Handlungen ([Wunderlich und Reinelt 1982]). Diese Handlungen sollen während der Navigation nacheinander ausgeführt werden. Im Idealfall ist die Instruktionsanweisung so beschaffen, dass durch die erfolgreiche Ausführung einer Handlung die Ausführung der darauffolgenden Handlung ermöglicht wird.[2]

[1] Siehe Abschnitt 4.2 *Klassisches Planen.*

[2] Anderenfalls besitzt die Routeninstruktion Lücken.

Dies lässt sich auf die Abarbeitung von Instruktionsanweisungen übertragen, wobei dem Argument einer Instruktionsanweisung eine besondere Rolle zukommt. Bevor der Agent die der Anweisung entsprechende primitive Aktion ausführen kann, muss er eine Beziehung zu der aktuellen Situation herstellen. Hierfür muss er das Objekt im Perzeptionsmodell ermitteln, welches als Parameter der primitiven Aktion fungieren kann. Vorraussetzung für die Abarbeitung einer Instruktionsanweisung ist somit ein erfolgreicher Koreferenzierungsprozess. Jede Instruktionsanweisung stellt die Vorbedingung an ihren Vorzustand, dass in diesem ihr Argument erfolgreich mit einem Objekt aus der Umgebung koreferenziert wurde. Da der Prozess der Koreferenzauflösung in dem Fall, dass mehrere Objekte für eine Koreferenzierung in Frage kommen, eine Liste dieser Objekte als Ergebnis liefert, wird die Vorbedingung auch schon dann erfüllt, wenn diese Liste nicht leer ist, da der ggf. anschließende Prozess, der aus dieser Liste ein Objekt auswählt, immer ein Ergebnis liefert (siehe hierzu Abschnitt 5.4.2).

Der Prozess der Koreferenzauflösung ermittelt ein Ähnlichkeitsmaß zischen den relevanten Teilnetzen aus dem Instruktions- und Perzeptionsgraphen, dass sich an der Struktur der Teilnetze orientiert. Da diejenigen Teilnetze (und die darin enthaltenen zentralen Objekte) als koreferent interpretiert werden, die einen gewissen Schwellwert überschreiten, ist dieser Schwellwert von zentraler Bedeutung bei dem Prozess der Koreferenzauflösung (siehe Abschnitt 3.5.2 und [Helwich 2003]) und ist auch dann verantwortlich, wenn der Prozess eine leere Liste als Ergebnis liefert.

Andersherum kann eine Instruktionsanweisung genau dann als erfolgreich abgearbeitet interpretiert werden, wenn die Vorbedingung der folgenden Instruktionsanweisung erfüllt ist. Somit lässt sich eine schwache **Vorbedingung für Instruktionsanweisungen** formulieren:

Liefert der Prozess der Koreferenzauflösung mindestens ein Ergebnis für das Argument der Instruktionsanweisung (d.h. die Ergebnisliste ist nicht leer), dann ist die Vorbedingung der Anweisung erfüllt.

5.2 Die primitiven Aktionen des Agenten

Der Agent führt während der Navigation zwei grundlegende Typen von Handlungen aus. Zum einen bewegt er sich, um seinem Ziel entgegenzukommen oder an bestimmten Stellen seine Orientierung zu verändern, zum anderen nimmt er seine Umgebung während der Navigation wahr. Diese beiden Handlungstypen müssen durch primitive Aktionen realisiert werden. Aktionen sind *primitiv*, wenn sie direkt durch den Agenten ausführbar sind.

Eine Möglichkeit der Simulation besteht in der parallelen Ausführung von Bewegungsaktionen und der kontinuierlichen Wahrnehmung der Umgebung - so wie Menschen auch gleichzeitig ein Wegstück entlanggehen und dabei nach einem Gebäude Ausschau halten können. In der Simulation auf einem Rechner könnte eine kontinuierliche Perzeption durch die Ausführung von einzelnen Perzeptionsschritten mit einer hinreichend hohen Frequenz approximiert werden (dies entspricht einer quasi-kontinuierlichen Perzeption). Durch eine quasi-kontinuierliche Perzeption wird jedoch kein für den Agenten nutzbarer Informationsgewinn erzielt. Nur an den Stellen, an der der Agent eine Entscheidung bezüglich des nächsten auszuwählenden Wegstücks treffen muss (an den Entscheidungspunkten), ist der Agent auf die Information aus der Perzeption angewiesen.

In der Simulation wurde deshalb ein anderer Ansatz gewählt: Der Agent verfügt über explizit anzustoßende Bewegungs- und Perzeptionsaktionen, die nicht gleichzeitig ausgeführt werden können. Zwischen der Ausführung zweier Perzeptionsaktionen, also z.B. während der Ausführung einer Bewegungsaktion, ist der Agent ‚blind'. Erst nach Beendigung der Bewegungsaktion kann der Agent eine Perzeptionsaktion ausführen und dadurch sein Perzeptionsmodell aktualisieren.

Diese Herangehensweise stellt aufgrund der Implementation der Bewegungsaktionen (s.u.), bei denen von Problemen der Bahnplanung und Bewegungsausführung abstrahiert wird, eine adäquate Modellierung dar. Eine derartige Modellierung wäre allerdings auf weniger abstrakten Stufen eines mobilen Robotersystems (vgl. Abb. 1.1) anders zu bewerten. Die Komponenten eines mobilen Robotersystems, die die Kompetenzen der Bahnplanung oder Bewegungsausführung bereitstellen, sind typischerweise auf eine Aktualisierung der Informationen aus der Perzeption angewiesen, die in deutlich kürzeren Intervallen stattfindet, um angemessen auf die Umgebung reagieren zu können. Die Modellierung und Repräsentation dieser Informationen kann dabei gänzlich anders beschaffen sein (z.B. können Sensordaten direkt für ein reaktives Verhalten des Roboters verwendet werden, wobei auf komplexe symbolische Berechnungsprozesse verzichtet wird), so dass echtzeitfähiges Verhalten ermöglicht werden kann.

Der Agent verfügt über drei primitive Aktionen (Siehe Tabelle 5.2). Eine Perzeptionsaktion ermöglicht die Wahrnehmung des Agenten. Zwei Bewegungsaktionen (`follow(`$\mathtt{w}_P$`)` und `turn(`$\mathtt{w}_P$`)`) stellen die Bewegungen des Agenten zur Verfügung.[3] Zusätzlich sind in der Tabelle die Hilfsprozesse angegeben, die den internen Zustand des Agenten manipulieren. Diese Prozesse werden in Abschnitt 5.3 vorgestellt.

5.2.1 Perzeption

Die Perzeptionsaktion `Perzeption` nimmt den für den Agenten sichtbaren Bereich der simulierten Umgebung wahr. Bei der Ausführung einer Perzeptionsaktion werden automatisch drei Prozesse des Agenten angestoßen: `Verarbeitung der Perzeption`, `Selbstlokalisation` und `Koreferenzauflösung(`$\mathtt{o}_I$`)`. Diese Prozesse treten in der Simulation nur zusammen nach einer Perzeptionsaktion auf. Die `Verarbeitung der Perzeption` aktualisiert den Perzeptionsgraphen, indem die neu perzipierten Objekte und deren geometrischen Eigenschaften hinzugefügt werden. Durch die `Selbstlokalisation` wird die aktuelle Position des Agenten im Instruktionsmodell ermittelt. Der Prozess `Koreferenzauflösung(`$\mathtt{o}_I$`)` sucht nach Teilnetzen im Perzeptiongraphen, die eine hinreichende Ähnlichkeit zu dem Teilnetz des Instruktionsgraphen besitzen, welches das Objekt $\mathtt{o}_I$ spezifiziert, und liefert als Ergebnis eine Liste $\mathtt{l}_{o_P}$ von möglichen Kandidaten für eine Koreferenzierung, sowie deren Koreferenzwerte.[4] Aus dieser Liste wird dann ein Objekt ausgewählt. Handelt es sich bei dem Objekt um einen Pfad, wählt der Prozess `Auswahl(`$\mathtt{l}_{w_P}$`)` mittels pragmatischer Annahmen einen Pfad $\mathtt{w}_P$ aus der Menge

[3] Der Index P gibt an, dass es sich bei den Argumenten um Referenzobjekte des Perzeptionsgraphen handelt. Der Index I bezeichnet Objekte aus dem Instruktionsgraphen.

[4] Eigentlich liefert der Prozess als Ergebnis eine Liste von Teilnetzen aus dem Instruktions- und Perzeptionsgraphen, die erfolgreich koreferenziert wurden (sogenannte *matches*) und deren Koreferenzwerte (Siehe Abschnitt 3.5 *Prozesse in der Navigationsphase* und [Helwich 2003]). Da aber bei der Ausführung einer Aktion nur das zentrale Objekt $\mathtt{o}_P$ des entsprechenden Teilnetztes aus dem Perzeptionsgraphen als Parameter fungiert, wird an dieser Stelle und weiterhin in dieser Arbeit das zentrale Objekt $\mathtt{o}_P$ als Argument angegeben.

Primitive Aktionen	**(Hilfs-)Prozesse**	**Interpretation**
`Perzeption`		Wahrnehmung der aktuellen Szene in der Simulation.
	`Verarbeitung der Perzeption`	Aktualisierung des Perzeptionsgraphen.
	`Selbstlokalisation`	Selbstlokalisation im Umgebungsmodell.
	`Koreferenzauflösung(o`$_I$`)`	Suche nach koreferenzierbaren Objekten im Perzeptionsgraphen. Liefert als Ergebnis eine Liste `l`$_{o_P}$ von erfolgreich koreferenzierten Objekten.
	`Auswahl(l`$_{w_P}$`)`	Wählt aus der Liste von koreferenzierten Pfaden aus dem Perzeptionsgraphen einen aus.
`follow(w`$_P$`)`		Folgen des Pfads, der durch `w`$_P$ spezifiziert ist.
`turn(w`$_P$`)`		Drehen in Richtung des Pfades, der durch `w`$_P$ spezifiziert ist.
	`BestPfadeZuReg(r`$_P$`)`	Bestimmung der Pfade im Perzeptionsgraphen, die in die Region `r`$_P$ führen.

Tabelle 5.2: Die Aktionen des Agenten und zugehörige Prozesse. Dabei kann zwischen einer Perzeptionsaktion und zwei Bewegungsaktionen unterschieden werden. Die internen Prozesse manipulieren das Wissen (bzw. die Annahmen) des Agenten über die aktuelle Situation.

von möglichen Pfaden aus dem Perzeptionsgraphen aus. Anderenfalls wird das Objekt aus der Liste ausgewählt, das den größten Koreferenzwert besitzt.

5.2.2 Bewegung

Eine Bewegungsaktion, die der Geometrische Agent in der simulierten Umgebung ausführen kann, wird durch zwei Parameter bestimmt: durch ihren Typ und durch ihr Argument. Die Auswahl einer primitiven Bewegungsaktion durch den Prozess der lokalen Planung entspricht somit der Auswahl eines Typs und eines Pfades, der als Argument für die Aktion dient.

- **Der Typ:** Der Agent verfügt über zwei Typen von Bewegungsaktionen: Das Folgen eines Pfades (`follow(w`$_P$`)`) und die Drehung in Richtung der Orientierung eines Pfades (`turn(w`$_P$`)`). Diese zwei Typen von primitiven Bewegungsaktionen sind ausreichend, da sich der Agent im Modell des Campus nur auf Wegstücken bewegen (und nicht den Rasen überqueren) kann, und komplexe Bewegungsaktionen durch die Hintereinanderausführung von atomaren primitiven Bewegungsaktionen gebildet werden können. Der

Vorbedingungen	**Aktion**	**Effekte**
Der Agent befindet sich am Startpunkt des Pfades w_P.	`follow(`w_P`)`	Der Agent befindet sich am Endpunkt des Pfades w_P und besitzt eine Orientierung, die der Orientierung des Pfades entlang des letzten Wegstücks entspricht.
Der Agent befindet sich am Startpunkt des Pfades w_P.	`turn(`w_P`)`	Der Agent befindet sich am Startpunkt des Pfades w_P und besitzt eine Orientierung, die der Orientierung des Pfades w_P entspricht.

Tabelle 5.3: Vorbedingungen und Effekte von Bewegungsaktionen.

Typ der aktuell auszuführenden Bewegungsaktion orientiert sich an der aktuellen Instruktionsanweisung aus dem Aktionsplan. Ist die aktuelle Anweisung vom Typ GO, wird eine primitive Aktion vom Typ follow ausgewählt, ist die Anweisung vom Typ CH_ORIENT, ist die Aktion vom Typ turn. Der Anweisungstyp BE_AT kann ebenso eine Aktion vom Typ follow initiieren.[5]

- **Das Argument:** Als Argumente für Bewegungsaktionen fungieren Objekte aus dem Perzeptionsgraphen, die dem Typ ‚Pfad' entsprechen.[6] Dem Pfad, der als Argument der Aktion dient, kommt eine besondere Bedeutung zu. Bei Aktionen vom Typ follow bestimmt er, welchem Routensegment der Agent in der Umgebung folgt. Bei einer Drehung (turn) bestimmt das Argument, wie weit die Drehung erfolgt. Bevor der Agent eine Bewegungsaktion ausführen kann, muss der Pfad aus dem Perzeptionsgraphen ermittelt werden, der als Argument fungieren soll. Den ersten Schritt leistet der Prozess der Koreferenzauflösung, der eine Liste von möglichen Pfaden als Ergebnis liefert. Aus dieser Liste wird durch den Prozess ‚Auswahl eines Pfades bei Bewegungsaktionen' ein Pfad ausgewählt.

Die Auswahl und Ausführung von primitiven Bewegungaktionen findet bei der lokalen Planung in zwei unterscheidbaren Kontexten statt. Zum einen werden Bewegungsaktionen ausgewählt und ausgeführt, um eine Handlung, die durch die Instruktionsanweisung vorgegeben ist, zu realiseren. Zum anderen werden primitive Bewegungsaktionen auch im Kontext einer Folgeaktion ausgewählt und ausgeführt. Welche Typen von Folgeaktionen das Aktionsmodul bereitstellt und wie diese in Abhängigkeit der aktuellen Situation aktiviert werden, wird im Abschnitt 5.7 *Folgeaktionen* gezeigt.

Die Tabelle 5.3 zeigt die Vorbedingungen und Effekte der primitiven Bewegungsaktionen. Durch die Aktion `follow(`w_P`)` verändert der Agent seine Position vom Startpunkt zum Endpunkt des Pfades w_P. Dabei orientiert er sich in Richtung des letzten Wegstücks des Pfades. Durch die Aktion `turn(`w_P`)` verändert der Agent seine Orientierung auf die Ausrichtung des ersten Wegstücks des Pfades, welcher durch w_P spezifiziert wird.

[5] Siehe Abschnitt 5.6 *Abarbeitung verschiedener Anweisungstypen.*

[6] Eine Ausnahme hiervon wird in Abschnitt 5.7 *Folgeaktionen* vorgestellt.

5.3 Prozesse während der Navigation

5.3.1 Prozesse für die Zustandsbestimmung

Drei Prozesse, die während der Navigation ausgeführt werden, manipulieren den internen Zustand des Agenten (siehe Abbildung 5.1):

- **Verarbeitung der Perzeption:** Führt der Agent eine Perzeptionsaktion aus, wird der anschließende Prozess ‚Verarbeitung der Perzeption' aktiviert. Er aktualisiert das Perzeptionsmodell, welches die Objekte aus der Wahrnehmung des Agenten repräsentiert (P-Netz).

- **Selbstlokalisation:** Der Prozess für die Koreferenzauflösung setzt eine Selbstlokalisation des Agenten voraus, durch die geklärt wird, an welcher Stelle sich der Agent momentan im Umgebungsmodell befindet und welche Orientierung er besitzt. Welche pragmatischen Annahmen eine Selbstlokalisation ermöglichen, wird in Abschnitt 5.4.1 gezeigt.

- **Koreferenzauflösung:** Durch die Koreferenzauflösung wird die Verknüpfung von I-Netz und P-Netz vorgenommen. Um die geometrische Information aus dem Instruktionsmodell für sich nutzbar zu machen, muss der Agent in der Lage sein, Objekte aus der Wegbeschreibung (die als Referenzobjekte im Instruktionsgraphen repräsentiert werden) in seiner Umgebung wiederzuerkennen, d.h. die Objekte aus dem Perzeptionsgraphen zu ermitteln, die koreferent zu dem Objekt des Instruktionsgraphen sind. Hierfür vergleicht der Prozess der Koreferenzauflösung das relevante Teilnetz aus dem Instruktionsgraphen mit Teilnetzen aus dem Perzeptionsgraphen (siehe Abschnitt 3.5.2 *Auflösung von Koreferenzen* und [Helwich 2003]). Aufgrund der Unterspezifiziertheit der Routeninstruktion besteht die Möglichkeit, dass mehrere Objekte des Perzeptionsmodells für eine Koreferenzierung in Frage kommen. Als Ergebnis liefert der Prozess eine Liste der Teilnetze aus dem Perzeptionsgraphen, welche mit dem relevanten Teilnetz des Instruktionsgraphen koreferenziert werden konnten, wobei für die Abarbeitung der Anweisung nur das zentrale Objekt von Relevanz ist[7]. Ein Koreferenzwert gibt an, welches Ähnlichkeitsmaß jeweils ermittelt wurde, und lässt dadurch eine Beurteilung der Güte der Koreferenzauflösung zu. Handelt es sich bei dem zentralen Referenzobjekt um einen Pfad, wird durch den unten beschriebenen Prozess ‚Auswahl eines Pfades für die folgende Bewegungsaktion' eine Auswahl eines Pfades aus der Liste der koreferenzierten Pfade vorgenommen. Anderenfalls wird das Objekt mit dem größten Koreferenzwert ausgewählt. Da als Argumente von Bewegungsaktionen Pfade fungieren, der Agent aber nur in der Lage ist, Wegstücke bzw. Routensegmente in der simulierten Umgebung wahrzunehmen, muss bei der Koreferenzauflösung von Pfaden eine Verknüpfung der beiden Konzepte erfolgen. Diese Verknüpfung wird bei der Verarbeitung der Perzeption vorgenommen, indem die potentiellen Pfade, deren Startpunkt der aktuellen Position des

[7] Aus diesem Grund wird weiter unten in dieser Arbeit als Ergebnis des Prozesses der Koreferenzauflösung eine Liste von zentralen Objekten der Teilnetze angegeben. Eigentlich wird eine Liste von Teilnetzen als Ergebnis geliefert (sog. *matches*), aus denen das zentrale Element für die Weiterverarbeitung durch das Aktionsmodul herausgenommen wird.

Agenten entspricht und die entlang der perzipierten Routensegmente verlaufen, automatisch generiert werden (siehe Abb. 5.2).

Die Prozesse ‚Selbstlokalisation' und ‚Koreferenzauflösung' werden durch die Ablaufsteuerung direkt aufgerufen, wenn das Aktionsmodul auf eine Aktualisierung des internen Zustands, d.h. auf die Ergebnisse dieser Prozesse angewiesen ist. Die ‚Verabeitung der Perzeption' wird immer dann ausgeführt, wenn der Agent eine Perzeptionsaktion ausgeführt hat.

5.3.2 Prozesse für die Auswahl von Bewegungsaktionen

Der Agent kann die Aktionen, die als Instruktionsweisungen im Aktionsplan repräsentiert werden, aufgrund der Unterspezifiziertheit der Routeninstruktion nicht direkt ausführen. Während der Navigation wird eine Anweisung im Aktionsplan im Kontext der Wahrnehmung der aktuellen Situation interpretiert und in einem dynamischen Prozess der lokalen Planung auf eine Sequenz abgebildet, die sowohl primitive Aktionen als auch die Prozesse zur Manipulation des internen Zustands des Agenten beinhaltet. Bei der lokalen Planung von Bewegungsaktionen werden weitere Hilfsprozesse benötigt:

- **Auswahl eines Pfades für die folgende Bewegungsaktion**: Der Prozess für die Auswahl eines Pfades ist ein Hilfsprozess, der während der lokalen Planung ausgeführt wird. Da für den Prozess der Koreferenzauflösung aufgrund der Unterspezifiziertheit des Instruktionsmodells die Möglichkeit betsteht, mehrere Referenzobjekte des Perzeptionsgraphen als mögliche Kandidaten zu ermitteln, muss in diesem Fall eines dieser Objekte ausgewählt werden. Da der ausgewählte Pfad bei Anweisungen, die Bewegungen entsprechen, als Argument der Bewegungsaktion fungiert, stellt er den entscheidenen Parameter der Bewegungsaktion dar.

  Der Prozess der Koreferenzauflösung ermittelt die Koreferenzbeziehungen über die Struktur der Teilnetze und weist denjenigen Netzen einen hohen Koreferenzwert zu, die eine hohe strukturelle Ähnlichkeit besitzen ([Helwich 2003]). Da die Pfade, für deren Teilnetze der höchste Koreferenzwert ermittelt wurde, nicht immer geeignete Kandidaten darstellen, werden bei der Auswahl von Pfaden für Bewegungsaktionen weitere pragmatische Annahmen getroffen (siehe Abschnitt 5.4.2).

- **Bestimmung von Pfaden zu einer Zielregion:** Dieser Prozess wird für die Abarbeitung von Anweisungen vom Typ BE_AT benötigt. Die Anweisung !BE_AT(r) ist als die Aufforderung „Versichere Dich, in der Region r zu sein, anderenfalls begib dich zu r." zu interpretieren. Stellt der Agent bei der Abarbeitung der Anweisung fest, dass er sich nicht in r befindet, muss er ein Verfahren bereitstellen, durch das er sich zu r hinbewegt. Dies beinhaltet die Aufgabe, einen Pfad zu ermitteln, dessen Startpunkt mit der aktuellen Position des Agenten identisch ist und dessen Endpunkt in der Region r liegt. Der Prozess ‚Bestimmung von Pfaden zu einer Zielregion' ermittelt zu diesem Zweck alle Pfade aus dem Perzeptionsgraphen, die diesen Bedingungen entsprechen.

5.3.3 Prozesse für die Problembehandlung

Aufgrund der Unsicherheit besteht für den Geometrischen Agenten prinzipiell das Problem, dass während der Navigation Schwierigkeiten auftreten. Zwei weitere Verfahren sollen

gewährleisten, dass der Agent trotzdem handlungsfähig bleibt:

- **Umgang mit Lücken im Aktionsplan:** Da im Aktionsplan nur Anweisungen auftauchen, die in der Wegbeschreibung explizit durch einen imperativen Ausdruck (d.h. durch ein Verb, welches eine Aktion beschreibt) repräsentiert sind, kann eine Lücke im Aktionsplan entstehen, wenn eine Aktion auf eine andere Weise beschrieben ist. Beispielsweise kann die deklarative Formulierung wie „Hinter Haus C befindet sich ein Weg“ implizit die Aufforderung beinhalten, diesem Weg zu folgen. Hier muss ein Verfahren die Lücke erkennen und geeignete Aktionen und Prozesse des Agenten anstoßen können.
- **Der Agent hat sich verlaufen:** Desweiteren besteht aufgrund der Unsicherheit prinzipiell die Möglichkeit, dass der Agent sich verläuft, d.h. eine falschem Routensegment folgt. Soll der Agent in derartigen Situationen handlungsfähig bleiben, muss der Agent das Problem erkennen und die Ablaufsteuerung ein Verfahren bereitstellen, das den Agenten in einen definierten Zustand zurücksetzt, von dem aus er einen erneuten (Teil-) Durchlauf starten kann.

Um die lokale Planung vornehmen zu können, muss durch die Ablaufsteuerung des Aktionsmoduls die Koordinierung der Prozesse ‚Verarbeitung der Perzeption‘, ‚Selbstlokalisation‘, ‚Koreferenzauflösung‘, ‚Auswahl eines Pfades bei Bewegungsaktionen‘ und ‚Bestimmung von Pfaden zu einer Zielregion‘ realisiert sein. Die Verfahren für die Problembehandlung stellen zusätzliche Maßnahmen zur Kompensation der Unsicherheit dar und können für ein erstes Verfahren außer Acht gelassen werden. Der Abschnitt 5.5 *Die Ablaufsteuerung* stellt das Verfahren vor, welches eine Realiserung der Ablaufsteuerung vornimmt.

5.4 Pragmatische Annahmen für die Hilfsprozesse

5.4.1 Pragmatische Annahmen für die Selbstlokalisation

Der Prozess der Koreferenzauflösung stellt eine Verbindung zwischen dem Objekt, welches als Argument einer Instruktionsanweisung fungiert, und den Objekten der perzipierten Umgebung her. Der Prozess vergleicht hierfür das Teilnetz aus dem Instruktionsgraphen, das als aktuell relevant bewertet wird, mit entsprechenden Teilnetzen des Perzeptionsgraphen.

Um zu ermitteln, welches Teilnetz des Instruktionsgraphen relevant ist, muss vor dem Prozess der Koreferenzauflösung die Position des Agenten im Umgebungsmodell bestimmt werden. Dies leistet der Prozess der Selbstlokalisation. Vier Schritte führen zur Bestimmung der aktuellen Position des Agenten im Umgebungsmodell, von denen die ersten drei zwischen der Verarbeitung der Perzeption und der Koreferenzauflösung und der vierte nach der Ausführung einer Bewegungsaktion ausgeführt werden:

1. Durch die Verarbeitung der Perzeption wird der Perzeptionsgraph aktualisert. Neben der Generierung neuer Teilnetze wird dabei die aktuelle Position des Agenten im **Perzeptionsgraphen** bestimmt, indem ein Knoten eingefügt wird, der die aktuelle Position des Agenten repräsentiert.

2. Mittels einer pragmatischen Annahme wird anhand des Arguments der folgenden Routeninstruktion die aktuelle Position des Agenten im **Instruktionsgraphen** bestimmt. Entspricht die folgende Anweisung einer Bewegungsaktion, dann wird die aktuelle Position des

Agenten im Instruktionsmodell mit dem Startpunkt des Pfades, welches durch die Anweisung angegeben wird, gleichgesetzt. Anderenfalls wird die letzte Position des Agenten im Instruktionsmodell beibehalten.

3. Die beiden Knoten aus dem Instruktions- und Perzeptionsgraphen, die die aktuelle Position des Agenten repräsentieren, werden als koreferent markiert.

4. Zusätzlich wird nach erfolgreicher Abarbeitung einer Instruktionsanweisung, die dem Folgen eines Pfades entspricht, die Position des Agenten mit dem Endpunkt des Pfades im **Instruktionsgraphen** gleichgesetzt.

Die unter dem zweiten Punkt genannte Annahme ist optimistisch, da sie davon ausgehen, dass durch die Ausführung der letzten Bewegungsaktion die aktuelle Instruktionsanweisung erfolgreich abgearbeitet worden ist. Ob dies tatsächlich der Fall ist, wird aber erst durch den anschließenden Prozess der Koreferenzauflösung ermittelt, der prüft, ob das Argument der folgenden Instruktionsanweisung in der Umgebung identifiziert werden kann. Ist das Ergebnis positiv, hat sich die Annahme als korrekt erwiesen. Bei einem negativen Ergebnis werden ggf. weitere Bewegungs- und Perzeptionsaktionen des Agenten ausgeführt. Da aber die aktuelle (und somit auch die folgende) Instruktionsanweisung beibehalten wird, werden bei erneuter Ausführung der Selbstlokalisation nur die Schritte 1., 2. und 3. wiederholt.

5.4.2 Pragmatische Annahmen für die Auswahl eines Pfades

Der Prozess der Koreferenzauflösung stellt eine Verknüpfung zwischen dem Instruktions- und dem Perzeptionsmodell her, indem er die Teilnetze aus dem Perzeptionsgraphen ermittelt, die eine hinreichend hohe Ähnlichkeit zu dem relevanten Teilnetz aus dem Instruktionsgraphen besitzen (siehe Abschnitt 3.5.2 *Auflösung von Koreferenzen* und [Helwich 2003]). Das relevante Teilnetz des Instruktionsgraphen hat das Argument der aktuellen Instruktionsanweisung als zentrales Objekt und beinhaltet weitere Referenzobjekte, die mit dem zentralen Objekt in einer relevanten Beziehung stehen. Das Verfahren, das die Koreferenzen auflöst, ermittelt jeweils die strukturelle Ähnlichkeit der Teilnetze und ordnet den Verknüpfungen (den sog. *matches*) einen Ähnlichkeits- oder Koreferenzwert zu. Alle Teilnetze aus dem Perzeptionsgraphen, deren Ähnlichkeitswert bei der Verknüpfung mit dem Teilnetz aus dem Instruktionsgraphen einen gewissen Schwellwert überschreiten, werden als koreferent interpretiert ([Helwich 2003]).

Als Ergebnis liefert der Prozess der Koreferenzauflösung eine Liste von Teilnetzen aus dem Perzeptionsgraphen, die als koreferent gewertet wurden. Diese Teilnetze beinhalten jeweils das für eine Koreferenzierung in Frage kommende Objekt als zentrales Referenzobjekt. Aus diesen zentralen Referenzobjekten muss noch ein Objekt ausgewählt werden, welches als Argument der primitiven Aktion dient, die für die Abarbeitung der aktuellen Instruktionsanweisung ausgeführt wird.

Ist der Objekttyp eine Landmarke, wird das Objekt ausgewählt, dessen Teilnetz die höchste strukturelle Ähnlichkeit zu dem relevanten Teilnetz des Instruktionsgraphen und somit den größten Koreferenzwert besitzt.

Handelt es sich bei der aktuellen Instruktionsanweisung um eine Anweisung, die eine Bewegungsaktion beschreibt, ist das Argument der Anweisung ein Pfad aus dem Instruktionsgraphen. Der entsprechende Pfad aus dem Perzeptionsgraphen stellt den wesentlichen Parameter der folgenden primitiven Bewegungsaktion dar, die im Prozess der lokalen Pla-

nung ermittelt wird. Der Auswahl des Pfades aus der Liste von Alternativpfaden kommt eine besondere Bedeutung zu, da sie entscheidend für performantes Handeln des Agenten ist: Wird der falsche Pfad ausgewählt, kann das dazu führen, dass der Agent die falsche Entscheidung in einem Entscheidungspunkt trifft und einer falschen Route folgt.

Die einfache Strategie, den Pfad auszuwählen, dessen *match* den höchsten Koreferenzwert besitzt, ist nicht angebracht, da die größte strukturelle Ähnlichkeit der Teilnetze nicht immer auf einen adäquaten Pfad verweist. Die Performanz des Agenten wird z.B. in dem Fall beinträchtigt, wenn der Pfad, dessen *match* die höchste Ähnlichkeit besitzt, auf einem relativ langen Routensegment verläuft, und der Agent durch die Wahl dieses Pfades ‚zu weit' geht, d.h. an einem relevanten Entscheidungspunkt vorübergeht und diesen ignoriert.

Da die Anzahl der vom Agenten während der Navigationsphase ausgeführten Aktionen das Performanzkriterium nicht beeinflusst, wohl aber das wiederholte Folgen derselben Wegstücke, ist eine defensive Strategie bei der Auswahl des Pfades angebracht, die ein ‚zu-weit-gehen' des Agenten und das damit verbunden wiederholte Folgen eines Wegstücks (da der Agent zu einem früheren Entscheidungspunkt zurückkehren muss) vermeidet. Diese defensive Strategie, kombiniert mit dem Abarbeitungsverfahren (siehe Abschnitt 5.5), sorgt auch zwischen zwei explizit genannten Entscheidungspunkten für ein adäquates Verhalten des Agenten, indem weitere Bewegungsaktionen ausgeführt werden können, wenn sich herausstellt, dass der gegangene Pfad zu kurz war.[8]

Der Prozess ‚Auswahl eines Pfades für die folgende Bewegungsaktion' (mit der Prozedur `Auswahl(`$\mathtt{l}_{wP}$`)`) ist für diese defensive Auswahl verantwortlich. Technisch werden die koreferenzierten Objekte weiter in einer Liste gespeichert. Die Objekte sollen dabei so geordnet sein, dass das erste Objekt der beste Kandidat ist, und die folgenden Objekte absteigend bewertet werden.[9] Die Auswahl erfolgt dann nach der Sortierung, indem das erste Element der Liste herangezogen wird. Die Sortierung orientiert sich an drei Kriterien:

- **Der Koreferenzwert**: Der Prozess der Koreferenzauflösung liefert als Ergebnis eine Liste von möglichen Teilnetzen, mit den in Frage kommenden Pfaden als zentrales Objekt ([Helwich 2003]). Da der Ähnlichkeitswert die strukturelle Ähnlichkeit der Teilnetze aus dem Perzeptions- und dem Instruktionsgraphen beschreibt, ist der Ähnlichkeitswert als alleiniges Kriterium für die Auswahl problematisch. Dennoch sind Objekte mit einem hohem Koreferenzwert Objekten mit einem niedrigeren vorzuziehen.

- **Enthaltensein / Auswahl eines möglichst kurzen Pfades**: Für den Abarbeitungsmechanismus, der in Abschnitt 5.5 vorgestellt wird, ist eine defensive Strategie bei der Auswahl des nächsten Wegstücks sinnvoll, die sich an den topologischen Eigenschaften der Routensegmente orientiert. Diese Strategie soll gewährleisten, dass der Agent zwar zu kurze Pfade auswählen kann, aber nie zu weit geht. Der Prozess der Koreferenzauflösung liefert Pfade als Ergebnis, die durch topologische Beziehungen miteinander verknüpft sind (wie Enthaltensein oder Berührung). Interessant sind die Pfade, die in anderen Pfaden enthalten sind. Die defensive Strategie besteht darin, die Pfade so zu

[8] Dies sind sog. Folgeaktionen vom Typ 3, siehe Abschnitt 5.7 *Folgeaktionen*.

[9] Diese Herangehensweise ist für ein Verfahren sinnvoll, das angestoßen wird, wenn der Agent sich auf einer falschen Teilroute wähnt, d.h. annimmt, er habe sich verlaufen (siehe Abschnitt 5.10.2). Dabei kehrt der Agent zu einem früheren Entscheidungspunkt zurück und wählt einen anderen Pfad aus der Liste aus.

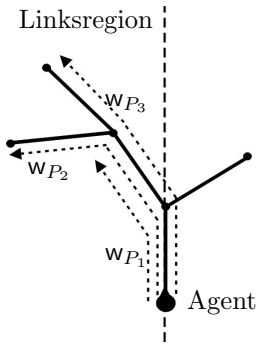


Abbildung 5.2: Sortierung nach dem Kriterium ‚Enthaltensein': Als Lösungen für eine Koreferenzauflösung mit dem Pfad w_I, der durch die Relation $\mathsf{TO}(\mathsf{w}_I,\mathsf{LEFT}(\mathsf{rsys}(\mathsf{Agent})))$ im Instruktiongraphen spezifiziert ist, kommen die Pfade in Frage, die ihren Startpunkt in der Position des Agenten haben und deren letztes Pfadstück nach links zeigt (relativ zum Referenzsystem des Agenten). Dies sind die drei Pfade w_{P_1}, w_{P_2} und w_{P_3}. Die Sortierung stellt den kürzesten Pfad w_{P_1} nach vorne und lässt die beiden anderen unsortiert, da sie bezüglich des Kriteriums gleichwertig sind.

sortieren, dass Pfade, die in möglichst vielen anderen Pfaden enthalten sind, vorzuziehen sind (siehe Abb. 5.2.). Als Ergebnis werden Pfade preferiert, die kurz sind und möglichst viele *Fortsetzungspfade* besitzen (siehe Abschnitt 5.7 *Folgeaktionen*).

- **Vermeidung zuvor gegangener Wegstücke**: Da in der Liste $\mathtt{l}_{w_P}$ auch Pfade auftauchen können, die entlang von Wegstücken verlaufen, denen der Agent in Teilen schon gefolgt ist (z.B. durch die letzte Bewegungsaktion), müssen diese Pfade gesondert behandelt werden. Hierfür werden die gegangenen Pfade gespeichert und mit der aktuellen Liste $\mathtt{l}_{w_P}$ verglichen, wobei von der Richtung des Pfades abstrahiert wird.[10] Überlappen sich die Pfade (d.h. besitzen sie gemeinsame Wegstücke), werden die Pfade hinten in der Liste eingeordnet und kommen praktisch nur für die Auswahl in Betracht, wenn keine Pfade vorliegen, denen der Agent noch nicht gefolgt ist (diese Situation kann z.B. auftreten, wenn der Agent an das Ende einer Sackgasse gelangt ist).

Die Strategie zur Auswahl eines Pfades für die folgende Bewegungsaktion kombiniert die drei oben genannten Kriterien bei der Sortierung der Pfade:

1. Es werden zwei Teillisten gebildet. Die erste Teilliste beinhaltet die Pfade, denen der Agent noch nicht gefolgt ist. Die zweite Teilliste beinhaltet alle Pfade, denen er schon (in Teilen) gefolgt ist. Innerhalb dieser Teillisten wird nach 2. sortiert.

2. Bilden von Teillisten anhand des Koreferenzwerts. Die Teillisten beinhalten Objekte, für deren Teilnetze bei dem Prozess der Koreferenzauflösung derselbe Ähnlichkeitswert ermittelt wurde. Innerhalb dieser Teillisten wird dann nach 3. sortiert.

[10] Das heisst, es wird nicht zwischen Start- und Endpunkt unterschieden.

3. Sortierung der Objekte nach dem Kriterium des Enthaltenseins. Die Pfade, die bei der Prüfung auf Enthaltensein miteinbezogen werden, können auch aus den anderen Teillisten stammen, d.h. andere Ähnlichkeitswerte aufweisen.

Diese Sortierung führt zu einer partiellen Ordnung innerhalb der Teillisten, da bei allen drei Sortierkriterien zu erwarten ist, dass Elemente gleich bewertet werden. Die resultierende Liste hat die folgende Form:

$$\underbrace{\Big[\underbrace{\underbrace{[w_{111}, w_{112}, w_{113}, ...]}_{\text{Sortierung nach 3.}}, [w_{121}, w_{122}, w_{123}, ...], ...}_{\text{Sortierung nach 2.}}\Big], \Big[[w_{211}, w_{212}, w_{213}, ...], [w_{221}, w_{222}, w_{223}, ...], ... \Big]}_{\text{Sortierung nach 1.}}$$

Durch die beiden äußeren Teillisten werden die Pfade, denen der Agent zuvor gefolgt ist (Pfade mit dem Index w_{2ij}), von denen getrennt, auf denen er sich noch nicht bewegt hat (w_{1ij}). Innerhalb dieser zwei Teillisten findet eine Unterscheidung nach dem Kriterium ‚Koreferenzwert' statt. Resultat der Sortierung ist eine weitere Unterteilung in Teillisten, in denen alle Pfade denselben Koreferenzwert besitzen (dargestellt durch den zweiten Index j eines Pfades w_{ijk}). Zuletzt wird innerhalb dieser Teillisten nach dem Kriterium ‚Enthaltensein' sortiert (die Pfade, die für das das Kriterium miteinbezogen werden, müssen jedoch nicht denselben Koreferenzwert besitzen).

Die hier vorgeschlagene Anordnung der Sortierungskriterien hat ein spezifisches Verhalten des Geometrischen Agenten zur Folge: Durch eine erste Sortierung nach dem Aspekt, ob der Agent einem Wegstück zuvor gefolgt ist, kommen diese Pfade nur für die folgende Bewegungsaktion in Betracht, wenn keine anderen Pfade vorliegen. Dies führt dazu, dass der Agent nur einen Weg zurückgeht, wenn er keine anderen Möglichkeiten besitzt.[11] Das zweite Sortierungskriterium hat zur Folge, dass für die Pfadauswahl nur Pfade in die engere Auswahl gelangen, die den höchsten Koreferenzwert besitzen. Aus diesen Pfaden wird dann der kürzeste Pfad ausgewählt. Diese Strategie ist dann sinnvoll, wenn der Prozess der Koreferenzauflösung typischerweise für die kurzen Pfade, die in anderen, längeren Pfaden enthalten sind, denselben Ähnlichkeitswert wie für die längeren Pfade ermittelt.[12] Das Vertauschen der Kriterien ‚Enthaltensein' und ‚Koreferenzwert' ist jedoch sinnvoll oder gar notwendig, wenn für die längeren Pfade typischerweise ein höherer Koreferenzwert ermittelt wird, und diese somit kürzeren Pfaden vorgezogen werden.

Das Sortierungsverfahren ist ein Parameter, durch dessen Veränderung (z.B. durch ein Vertauschen der Sortierungskriterien) sich das Verhalten des Agenten steuern lässt (siehe Abschnitt 5.10.2). Ob eine andere Anordnung der Sortierungskriterien die Performanz des Geometrischen Agenten erhöhen kann, kann erst nach einer (noch ausstehenden) Testphase evaluiert werden.

[11] Eine Ausnahme besteht, wenn der Agent gezielt zu einem früheren Entscheidungspunkt zurückkehrt. Dabei ist aber der Prozess zur Auswahl eines Pfades nicht beteiligt.

[12] Und dieses Verhalten zeigt sich bei der Parametrisierung der aktuellen Implementation.

5.5 Die Ablaufsteuerung

Die Ablaufsteuerung des Aktionsmoduls wählt während der Navigationsphase die primitven Aktionen des Agenten aus und stößt deren Ausführung an. Für die Auswahl der primitiven Aktionen ist der Prozess der lokalen Planung verantwortlich. Zusätzlich müssen durch den Prozess auch die Hilfsprozesse angestoßen werden, die zur Aktualisierung des internen Zustands dienen. Wie bei einem kontinuierlich planenden Agenten[13] findet keine zeitliche Trennung zwischen dem Prozess der Auswahl und der Ausführung der Aktionen statt. Nach der Auswahl einer Aktion bzw. eines Prozesses wird sofort die Ausführung angestoßen.

Der grundlegende Ablauf

Eine einfache Möglichkeit in der Abarbeitung des Aktionsplans besteht darin, die Anweisungen aus dem Aktionsplan nach und nach abzufragen und in entsprechende primitive Aktionen zu übersetzen, bis alle Aktionen bearbeitet wurden. Vor der Ausführung einer Bewegungsaktion wird eine Aktualisierung des internen Zustands vorgenommen, wodurch der Agent feststellen kann, welche Pfade der aktuell wahrgenommenen Umgebung als Parameter der Bewegungsaktion in Frage kommen. Aus dieser Liste von koreferenzierten Pfaden wird dann einer ausgewählt. Nach Ausführung der Bewegungsaktion wird dann die nächste Anweisung aus dem Aktionsplan abgearbeitet.

Dieses einfache Verfahren wird jedoch nicht dem Problem der Unsicherheit gerecht, dass aus der Unterspezifiziertheit der Routeninstruktion folgt. Die Auswahl des Pfades für die Bewegungsaktion geschieht unter pragmatischen Annahmen. Es kann nicht als sicher gelten, dass der ausgewählte Pfad demjenigen entspricht, den der Instruktor bei der Formulierung der Routeninstruktion im Sinn hatte. Dies kann insbesondere aufgrund der defensiven Strategie dazu führen, dass der Agent eine Bewegung auf einem Pfad vornimmt und die entsprechende Anweisung als abgearbeitet interpretiert, die Vorbedingung der nächsten Instruktionsanweisung jedoch noch nicht erfüllt ist, da der Agent das Argument der folgenden Anweisung nicht in der Umgebung erkennen kann. Eine andere Strategie zur Auswahl des Pfades für die folgende Bewegungsaktion ist mit vergleichbaren Problemen behaftet. Durch eine offensive Strategie (d.h. der Agent würde immer möglichst weit laufen) besteht die Gefahr, dass der Agent die Position, an der er die nächste Anweisung abzuarbeiten hat, verpasst, was besonders bei Anweisungen fatal wäre, die den Agenten auffordern, auf einen anderen Weg abzubiegen.

Eine Möglichkeit, diesem Problem gerecht zu werden, besteht in der Kombination der defensiven Strategie zur Wegauswahl und dem Konzept der Aktionsüberwachung. Die Aktionsüberwachung prüft nach jeder Ausführung einer primitiven Bewegungsaktion, ob die aktuelle Anweisung erfolgreich abgearbeitet wurde. Eine Instruktionsanweisung gilt als erfolgreich abgearbeitet, wenn der Agent eine Position im geometrischen Modell des Campus erreicht hat, in der er die folgende Instruktionsanweisung abarbeiten kann. Hierfür muss er das Argument der folgenden Instruktionsanweisung in der Umgebung identifizieren können. Nur wenn er das dem Argument entsprechende Objekt aus der Umgebung erkannt hat, kann er die Anweisung abarbeiten, was der in Abschnitt 5.1.4 genannten Vorbedingung einer Instruktionsanweisung entspricht.

[13] Siehe Abschnitt 4.4.2 *Kontinuierliches Planen.*

Die Prozesse und Aktionen, die während der lokalen Planung ausgeführt werden, können in drei übergeordneten Prozessen gekapselt werden, denen dynamisch in Abhängigkeit der Situation die Kontrolle übergeben wird: *Ausführung der Bewegungsaktion*, *Aktualisierung des internen Zustands* und *Folgeaktion*. In dem übergeordneten Prozess ‚Ausführung der Bewegungsaktion' sind der Prozess zur Auswahl des Pfades und die Bewegungsaktion zusammengefasst. Der übergeordnete Prozess ‚Aktualisierung des internen Zustands' beinhaltet die Perzeptionsaktion sowie die Prozesse, die sich der Perzeption anschließen (‚Verarbeitung der Perzeption', ‚Selbstlokalisation' und ‚Koreferenzauflösung'). Durch den Prozess ‚Koreferenzauflösung' wird versucht, das Argument der folgenden Instruktionsanweisung in der Umgebung zu identifizieren (d.h. koreferenzierbare Teilgraphen im Perzeptionsgraphen zu ermitteln). Durch einen Test kann nach der Aktualisierung des internen Zustands festgestellt werden, ob die Koreferenzauflösung erfolgreich war. Dies ist genau dann der Fall, wenn die Liste, die der Prozess als Ergebnis liefert, nicht leer ist (siehe Abschnitt 5.1.4).

Bei erfolgreicher Koreferenzauflösung ist die Vorbedingung der folgenden Aktion erfüllt und die aktuelle Anweisung gilt als abgearbeitet. War die Koreferenzauflösung nicht erfolgreich, ist der Agent in der aktuellen Situation noch nicht in der Lage, die folgende Instruktionsanweisung zu bearbeiten. Um in dieser Situation entsprechend zu handeln, wird der komplexe Prozess ‚Folgeaktion' angestoßen. Bei der Folgeaktion handelt es sich um einen komplexen Prozess und nicht um eine primitive Aktion, da sich eine Folgeaktion aus primitiven Aktionen und Hilfsprozessen zusammensetzt (siehe Abschnitt 5.7 *Folgeaktionen*).

Im folgenden Abschnitt wird gezeigt, wie die unterschiedlichen Anweisungstypen abgearbeitet werden, indem den oben genannten Prozessen dynamisch, d.h. in Abhängigkeit der Situation, die Kontrolle übergeben wird.

5.6 Abarbeitung verschiedener Anweisungstypen

Der Aktionsplan beinhaltet Instruktionsanweisungen, für die vier Typen unterschieden werden können, die unterschiedlich abgearbeitet werden: GO, CH_ORIENT, BE_AT und VIEW.[14]

5.6.1 Anweisungen vom Typ GO und CH_ORIENT

Die Abarbeitung der Typen GO und CH_ORIENT unterscheidet sich nur an zwei Punkten, so dass sie in einem Abschnitt behandelt werden können: in dem Typ der Bewegungsaktion, die angestoßen wird, sowie in der Folgeaktion. Die Abarbeitung dieser Typen wird in der Abbildung 5.3 schematisch dargestellt.

Als erster Schritt bei der Abarbeitung wird die Komponente ‚Ausführung der Bewegungsaktion' aktiviert. Damit eine Anweisung aus dem Aktionsplan abgearbeitet werden kann, muss ihre Vorbedingung erfüllt sein, d.h. es wurde eine nichtleere Liste von Kandidaten, die als Argument der folgenden primitiven Aktion dienen können, generiert. Diese Liste wird dem Prozess ‚Auswahl des Pfades für die folgende Bewegungsaktion' (in der Abbildung AW abgekürzt) übergeben. Aus dieser Liste wird mittels pragmatischer Annahmen[15] ein Pfad ausgewählt und die entsprechende Bewegungsaktion mit dem ermittelten Pfad als Argument

[14] Siehe Abschnitt 5.1.2 *Der Aktionsplan und die aktuell relevanten Instruktionsanweisungen.*
[15] Siehe Abschnitt 5.4.2 *Pragmatische Annahmen für die Auswahl eines Pfades.*

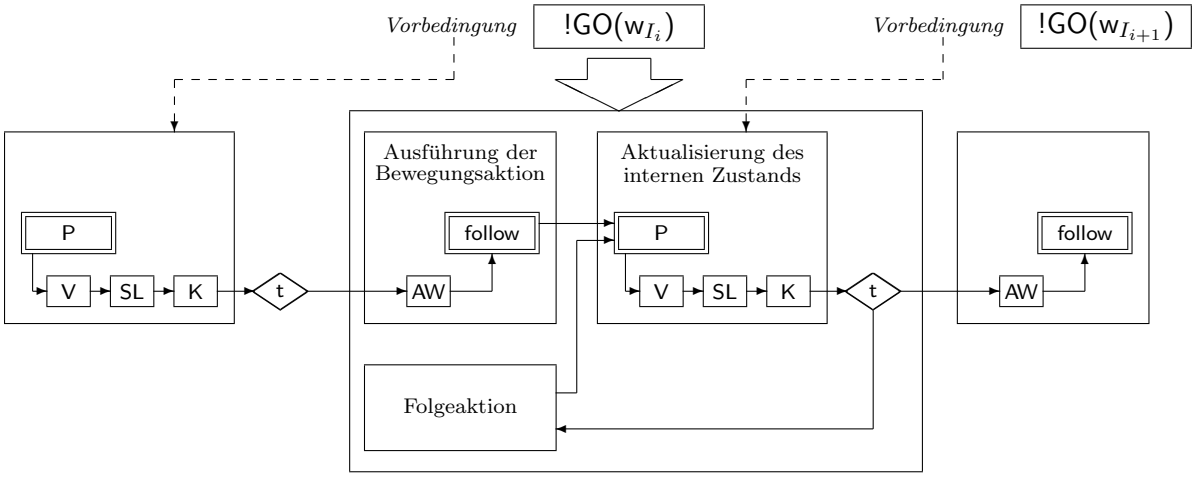


Abbildung 5.3: Abarbeitung einer Anweisung des Typs GO (Pseudo-Flussdiagramm zur schematischen Darstellung). Bei der Abarbeitung werden die übergeordneten Prozesse ‚Ausführung der Bewegungsaktion', ‚Aktualisierung des internen Zustands' und ggf. ‚Folgeaktion' angestoßen. Die primitiven Aktionen sind hervorgehoben (doppelt umrandete Kästen). Die Abarbeitung von CH_ORIENT erfolgt entsprechend, jedoch wird eine primitive Bewegungsaktion vom Typ turn ausgeführt.

angestoßen. Ist die aktuelle Anweisung vom Typ GO, wird eine Bewegungsaktion vom Typ follow angestoßen. Findet eine Abarbeitung einer Anweisung vom Typ CH_ORIENT statt, ist die Aktion vom Typ turn.

Nachdem die Bewegungsaktion ausgeführt wurde, hat der Agent entweder seine Position, seine Orientierung, oder beides verändert, so dass er sich in einer neuen Situation befindet. Nun wird der Prozess ‚Aktualisierung des internen Zustands' angestoßen. Ziel der Aktualisierung ist es, Informationen über die neue Situation zu erlangen, so dass festgestellt werden kann, ob die Vorbedingung der folgenden Instruktionsanweisung aus dem Aktionsplan in der neuen Situation erfüllt ist. Der Agent führt hierfür zuerst eine Perzeptionsaktion aus (P). Nach dem Prozess ‚Verarbeitung der Perzeption' (V) wurde das Perzeptionsmodell des Agenten aktualisiert, und der Agent führt den Prozess der Selbstlokalisation (SL) aus. Dieser Prozess erhält als Eingabe das Argument der folgenden Instruktionsanweisung (siehe Abschnitt 5.4.1).

Hiernach ermittelt der Prozess ‚Koreferenzauflösung' (K) die Objekte, die mit dem Argument der folgenden Instruktionsanweisung korrespondieren. Nach der Koreferenzauflösung wird das Ergebnis überprüft (in der Abbildung 5.3 dargestellt als bedingte Verzweigung t). War die Koreferenzauflösung erfolgreich, gilt die aktuelle Anweisung als abgearbeitet. Es wird die folgende Anweisung aus dem Aktionsplan zur Abarbeitung herangezogen und die Liste von koreferenzierten Objekten übergeben.

War die Koreferenzauflösung jedoch nicht erfolgreich, gilt die aktuelle Anweisung noch nicht als abgearbeitet und es wird der Prozess ‚Folgeaktion' angestoßen, der gewährleisten soll, dass der Agent durch die Ausführung einer primitiven Bewegungsaktion in eine Situation gelangt, in der die Vorbedingung der folgenden Anweisung erfüllt wird. Nach Ausführung

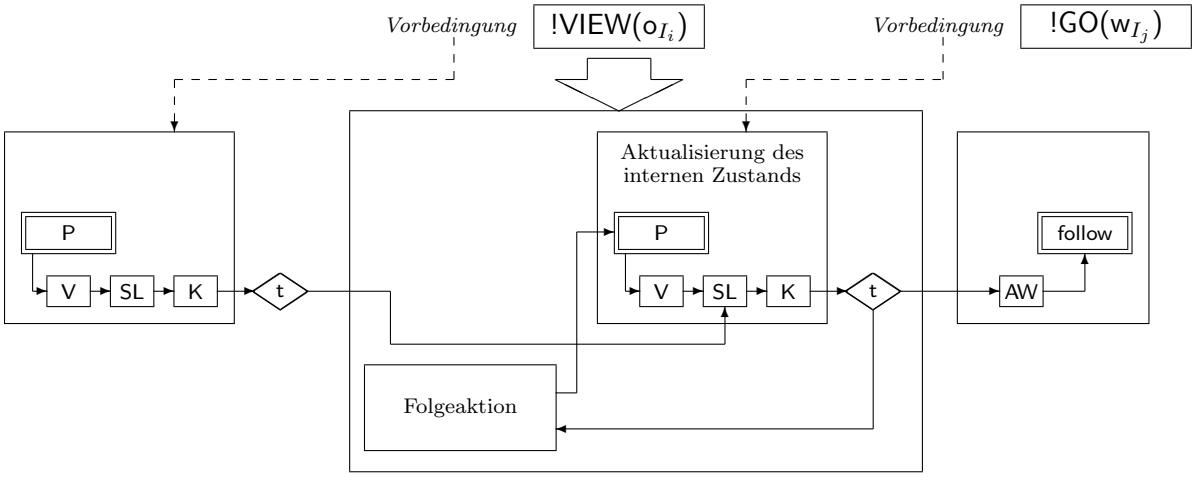


Abbildung 5.4: Abarbeitung einer Anweisung des Typs VIEW. Durch Prüfen der Vorbedingung während der Abarbeitung der vorhergehenden Anweisung wurde die intendierte Handlung schon ausgeführt. Als erster Schritt bei der Verarbeitung der aktuellen Anweisung vom Typ VIEW wird die Selbstlokalisation bei der Aktualisierung des internen Zustands ausgeführt, um die Vorbedingung der folgenden Anweisung zu überprüfen.

der Folgeaktion wird erneut eine Aktualisierung des internen Zustands vorgenommen, und die Vorbedingung der folgenden Instruktionsanweisung geprüft. Falls die Überprüfung abermals fehlschlägt, wird wieder der Prozess ‚Folgeaktion' angestoßen. Diese Schleife wird sooft durchlaufen, bis die Überprüfung der Vorbedingung erfolgreich war, oder aber ein Abbruch stattfindet (siehe Abschnitt 5.10.2 *Problemfall: der Agent hat sich verlaufen*). Verschiedene Strategien für die Komponente ‚Folgeaktion' werden in Abschnitt 5.7 vorgestellt.

5.6.2 Anweisungen vom Typ VIEW

Eine Anweisung vom Typ VIEW dient bei dem hier vorgestellten Verfahren nicht zur Aktivierung einer besonderen Handlung, sondern erfüllt den Zweck, zu bestimmen, wann die nächste Instruktionsanweisung herangezogen werden kann.

Durch den Anweisungstyp VIEW wird die Handlung beschrieben, nach einem Objekt Ausschau zu halten. Damit eine Anweisung in dem hier vorgestellten Verfahren bearbeitet werden kann, muss ihre Vorbedingung erfüllt sein. Das bedeutet, dass für die Aktivierung einer Anweisung vom Typ VIEW das Objekt, das als Argument der Anweisung fungiert, erfolgreich koreferenziert wurde. Somit ist die eigentliche Handlung, nach dem Objekt Ausschau zu halten, schon vor der Aktivierung der Anweisung durch den Prozess ‚Aktualisierung des internen Zustands' während der Abarbeitung der vorhergehenden Anweisung geleistet worden (siehe Abbildung 5.4).

Somit muss keine eigentliche Aktion mehr für die Abarbeitung einer Anweisung vom Typ VIEW ausgeführt werden. Dies führt dazu, dass bei der Abarbeitung der Anweisung gleich der übergeordnete Prozess ‚Aktualisierung des internen Zustands' angestoßen wird, um

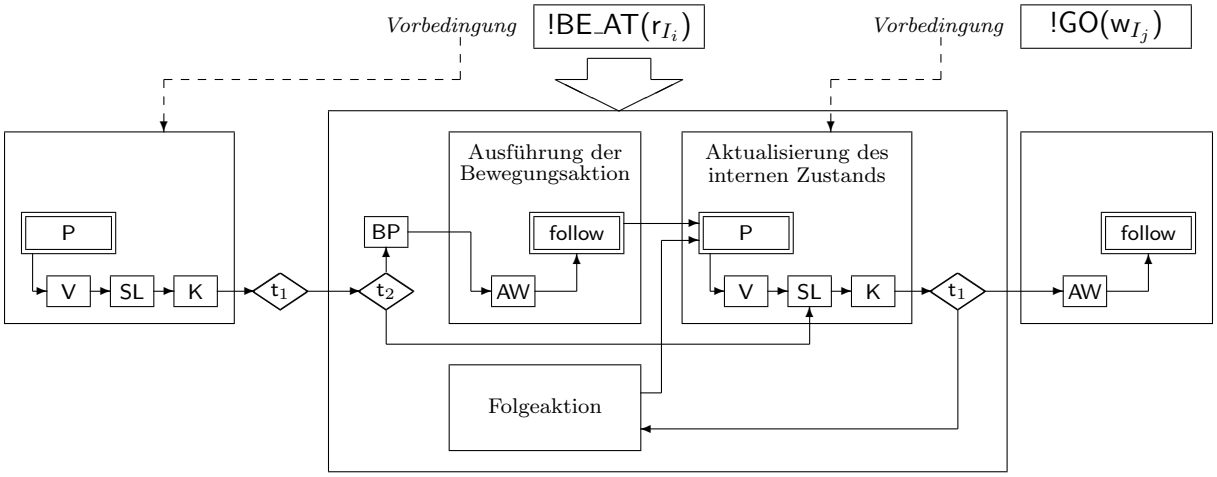


Abbildung 5.5: Abarbeitung einer Anweisung des Typs BE_AT.

die Vorbedingung der folgenden Anweisung zu überprüfen. Bei der erstmaligen Aktivierung können die Perzeptionsaktion sowie der Prozess ‚Verarbeitung der Perzeption' übersprungen und direkt der Prozess ‚Selbstlokalisation' ausgeführt werden, da sich der Agent seit der letzten Perzeption nicht bewegt hat. Der Prozess der Koreferenzauflösung erhält als Eingabe das Argument der folgenden Instruktionsanweisung. Ist die Koreferenzauflösung erfolgreich, wird auch hier die nächste Anweisung herangezogen und die Liste von koreferenzierten Objekten übergeben, schlägt sie jedoch fehl, wird die Komponente ‚Folgeaktion' aktiviert.

Konnte das Argument der aktuellen Anweisung vom Typ BE_AT erfolgreich koreferenziert werden (bei der Abarbeitung der vorherigen Anweisung), ist dadurch noch nicht gewährleistet, dass die letzte Anweisung zur Genüge abgearbeitet wurde, wenn durch die letzte Anweisung eine Bewegungsaktion vom Typ follow intendiert wird: Kann das Argument der folgenden Anweisung noch nicht perzipiert und koreferenziert werden, wird evtl. eine Folgeaktion angestoßen, die eine weitere Abarbeitung der letzten Anweisung darstellt, indem der Agent einem Pfad folgt, der eine Fortsetzung des zuletzt gefolgten Pfades ist (siehe Abschnitt 5.7).

5.6.3 Anweisungen vom Typ BE_AT

Welcher Handlung eine Anweisung vom Typ BE_AT entspricht, kann erst im Kontext der aktuellen Situation ermittelt werden (siehe Abb. 5.5). Da die Anweisung zur Abarbeitung herangezogen wurde, lieferte die Prüfung der Vorbedingung während der Abarbeitung der vorhergehenden Anweisung (in der Abbildung durch die bedingte Verzweigung t_1 gekennzeichnet) ein positives Ergebnis, d.h. der Agent konnte das Objekt, das dem Argument der aktuellen Anweisung entspricht, in der Umgebung erkennen. Durch die Anweisung !BE_AT(r) wird der Agent dazu aufgefordert, sich zu der Region r hinzubewegen, wenn er sich nicht schon dort befindet. Um die Anweisung adäquat interpretieren zu können, muss an dieser Stelle ein Test erfolgen, der ermittelt, ob sich der Agent in der Region r befindet (gekennzeichnet durch die bedingte Verzweigung t_2).

Ergibt der Test t_2, dass sich der Agent in der entsprechenden Region befindet, muss keine Bewegung erfolgen und es kann mit der Aktualisierung des internen Zustands und der Prüfung der Vorbedingung der folgenen Anweisung fortgefahren werden (in diesem Fall entspricht die Abarbeitung der des Typs VIEW).

Liefert der Test jedoch ein negatives Ergebnis, d.h. der Agent befindet sich nicht in der Region r, muss eine Bewegungsaktion zu dieser Region initiiert werden. Für die Bestimmung von Pfaden zu der Region ist der Prozess ‚Bestimmung von Pfaden zu einer Region' (BP) verantwortlich. Er ermittelt im Perzeptionsgraphen alle Pfade, die ihren Startpunkt in der aktuellen Position des Agenten und ihren Endpunkt in der Region r haben. Diese Pfade werden dann an den Prozess ‚Auswahl eines Pfades' weitergereicht. Anschließend wird eine Bewegungsaktion vom Typ follow mit dem ermittelten Pfad als Parameter ausgeführt.

Nach Ausführung wird die Komponente ‚Aktualisierung des internen Zustands' aktiviert. Die weitere Abarbeitung findet wie bei den anderen Anweisungstypen statt.

Initialisierung vor Abarbeitung der ersten Instruktionsanweisung

Die Abarbeitung einer Anweisung setzt voraus, dass ihr Argument erfolgreich koreferenziert werden konnte. Diese Koreferenzierung wird bei der Abarbeitung der vorhergehenden Anweisung getätigt. Da die erste Instruktionsanweisung aus dem Aktionsplan keinen Vorgänger besitzt, bei dessen Abarbeitung diese Koreferenzauflösung geleistet wird, muss vor Abarbeitung der ersten Anweisung eine initiale Aktualisierung des internen Zustands erfolgen.

Bearbeitung der letzten Anweisung aus dem Aktionsplan

Da die letzte Instruktionsanweisung aus dem Aktionsplan keinen Nachfolger besitzt, muss für ihre Abarbeitung ein modifiziertes Verfahren verwendet werden: Die letzte Anweisung wird abgearbeitet, indem eine Abarbeitung wie in den oben vorgetellten Verfahren vorgenommen wird, ohne dass jedoch nach der evtl. ausgeführten primitiven Aktion eine Zustandsaktualisierung geschieht. Dies stellt für die Routeninstruktionen keine Einschränkung dar, in denen die letzte Anweiung vom Typ VIEW oder BE_AT ist, da bei der Aktivierung der letzten Instruktionsanweisung das entsprechende Objekt schon in der Umgebung perzipiert und koreferenziert werden konnte, der Agent also (zumindest fast) sein Ziel erreicht hat.

Besitzt eine Routeninstruktion eine letzte Instruktionsanweisung von einem anderen Typ, könnte es sinnvoll sein, eine zusätzliche Anweisung !BE_AT(<Zielregion>) als letzte Anweisung einzufügen. Das Argument <Zielregion> repräsentiert die Region, in der sich Ziel der Route befindet.

Abarbeitung und lokale Planung

Unter Planung wird typischerweise ein Prozess verstanden, der als Ergebnis für eine Problemstellung eine Liste von auszuführenden Aktionen generiert. Bei der lokalen Planung durch das Aktionsmodul wird jedoch keine Liste von primitiven Aktionen im voraus generiert, die dann zu einem späteren Zeitpunkt abgearbeitet wird. Nach der Auswahl einer primitiven Aktion wird deren Ausführung direkt angestoßen. Damit ähnelt das Verfahren der Ablaufsteuerung einem kontinuierlichem Planer, bei dem auch nicht zwischen einer Planungs- und

Ausführungsphase unterschieden werden kann.[16]

Welche primitive Aktion als nächstes auszuwählen ist, kann erst nach Beendigung der zuletzt getätigten Aktion oder nach Erhalt des Ergebnisses des zuletzt ausgeführten Prozesses ermittelt werden. Somit liegt während der Abarbeitung einer Instruktionsanweisung kein langfristiger Plan von primitiven Aktionen in einer deklarativen Form vor, der nach und nach abgerabeitet werden kann. Dennoch ist es sinnvoll, von einem Prozess der Planung zu sprechen, da man annehmen kann, dass bei der Abarbeitung des abstrakten Aktionsplans ein Aktionplan aus konkreten Aktionen generiert wird: Werden die primitiven Aktionen, die der Agent während eines Durchlaufs ausführt, in einer Liste gespeichert, so liegt im Nachhinein ein Aktionsplan in deklarativer Form vor, der exakt beschreibt, welche Aktionen während der Navigation zu welchen Zeitpunkten vom Agenten ausgeführt worden sind.

5.7 Folgeaktionen

Eine Folgeaktion wird immer dann aktiviert, wenn die Überprüfung der Vorbedingung der folgenden Instruktionsanweisung ein negatives Ergebnis liefert. Die Folgeaktion soll zum einen gewährleisten, dass der Agent trotz dieses Problems handlungsfähig bleibt, und zum anderen ein adäquates Verhalten des Agenten bereitstellen, durch das er einen Zustand erreicht, in dem die Vorbedingung erfüllt wird.

Zwei Gründe können dafür verantwortlich sein, dass in der aktuellen Situation die Vorbedingung der folgenden Anweisung nicht erfüllt ist, d.h. der Agent das entsprechende Objekt aus dem Instruktionsmodell in der Umgebung nicht erkennen kann:

- Das in Frage kommende Objekt aus der Umgebung liegt außerhalb des Sichtfelds des Agenten.
- Der Agent ist zu weit von dem Objekt entfernt.

Das erste Problem kann leicht behoben werden, indem der Agent ‚sich umsieht'. Dies leistet die Folgeaktion 1. Das zweite Problem kann nur behoben werden, wenn der Agent sich zu einer Position hinbewegt, an der er die in Frage kommenden Objekte perzipieren kann. Die Folgeaktionen 2 und 3 stoßen hierfür entsprechende Prozesse und Bewegungsaktionen an, durch die der Agent seine Position in der Umgebung verändert.

Das Aktionsmodul besitzt einen Zähler, der bei jeder Aktivierung einer Folgeaktion inkrementiert wird. In Abhängigkeit des aktuellen Anweisungstyps und des aktuellen Zählerwerts wird eine der unten beschriebenen Folgeaktionen aktiviert. Nach erfolgreicher Abarbeitung der aktuellen Anweisung wird der Zähler auf null zurückgesetzt.

5.7.1 Typen von Folgeaktionen

Folgeaktion 1: Umherblicken

Diese Folgeaktion wird durch die Ausführung einer Bewegungsaktion geleistet, die vom Typ turn ist, aber als Parameter eine Winkel besitzt. Im ersten Schleifendurchlauf wird eine Drehung im Uhrzeigersinn um 30° getätigt. Danach wird die Komponente ‚Folgeaktion' deak-

[16] Siehe Abschnitt 4.4.2 *Kontinuierliches Planen.*

tiviert und die Komponente ‚Aktualisierung des internen Zustands' aktiviert. Befinden sich nun die entsprechenden Objekte im Sichtfeld des Agenten, führt dies zu einer erfolgreichen Überprüfung und die aktuelle Anweisung gilt als abgearbeitet. Bei einem negativen Ergebnis wird die Schleife ein zweites Mal durchlaufen. Dieses Mal dreht sich der Agent um 60° gegen den Uhrzeigersinn (um die erste Drehung rückgängig zu machen) und aktiviert danach erneut die Komponente für die Zustandsaktualisierung. Führt die Prüfung der Vorbedingung abermals zu keinem Ergebnis, wird ggf. eine der beiden anderen Folgeaktionen angestoßen.

Folgeaktion 2: zufällige Exploration

Eine einfache Möglichkeit, die momentane Position zu verändern, besteht in der Ausführung einer Bewegungsaktion des Typs follow auf einem zufällig ausgewählten Pfad. Durch diese zufällige Exploration besteht prinzipiell die Möglichkeit, dass durch die Positionsänderung ein Entscheidungspunkt erreicht wurde, an dem die Vorbedingung der folgenden Anweisung erfüllt wird. Der Pfad, der für die zufällige Exploration ausgewählt wird, sollte dabei nur entlang *eines* Wegstücks verlaufen[17], und der Agent sollte ihm noch nicht gefolgt sein. Nach der Ausführung wird auch hier der interne Zustand des Agenten aktualisiert und die Vorbedingung der folgenden Anweisung geprüft. Ist das Ergebnis negativ, sind zwei weitere Verfahrensweisen denkbar:

1. Der Agent führt von der aktuellen Position eine weitere Bewegungsaktion aus (Folgeaktion 2a). 2. Der Agent kehrt zu dem letzten Entscheidungspunkt zurück und wählt einen anderen Pfad für die zufällige Exploration aus (Folgeaktion 2b).

Folgeaktion 3: Folgen der Fortsetzung eines Pfades

Die dritte Folgeaktion stellt ein aufwändigeres Verfahren bereit. Für sie werden die *Fortsetzungspfade* des zuletzt gefolgten Pfades verwendet:

Der Prozess der Koreferenzauflösung liefert als Ergebnis eine Liste von Objekten. Ist die aktuelle Anweisung vom Typ GO, handelt es sich bei der als nächstes auszuführenden primitiven Aktion um eine Bewegungsaktion vom Typ follow, und es wird im Prozess ‚Auswahl eines Pfades für die folgende Bewegungsaktion' ein Pfad aus dieser Liste von Pfaden ausgewählt. Da der Agent bei der Auswahl eine defensive Strategie verfolgt, ist es möglich, dass der ausgewählte Pfad nicht bis zu dem Entscheidungspunkt reicht, an dem die Vorbedingung der nächsten Instruktionsanweisung erfüllt ist. Um diesem Problem gerecht zu werden, wird durch die Folgeaktion 3 eine weitere Bewegung auf einem Pfad vorgenommen, der eine Fortsetzung des vorangegangenen Pfades darstellt, so dass nach der Bewegung ein Zustand erreicht wird, der ebenfalls erreicht worden wäre, wenn der Agent eine weniger defensive Strategie verfolgt hätte. Hierfür wird nochmals die Liste von Pfaden verwendet, die für die schon ausgeführte Bewegungsaktion genutzt wurde. In dieser Liste werden die Teilpfade ermittelt, die eine Fortsetzung des letzten Pfades darstellen, d.h. ihren Startpunkt in der aktuellen Situation des Agenten haben und die nicht entlang des Routensegments verlaufen, dem der Agent durch die letzte Bewegungsaktion gefolgt ist (siehe Abbildung 5.6). Aus dieser Liste von Fortsetzungs-

[17] Dies entspricht der defensiven Strategie aus Abschnitt 5.4.2. Auch hier ist es besser, zuerst ein kurzes Wegstück auszuwählen, und ggf. erneut eine Bewegung auszuführen, als den nächsten relevanten Entscheidungspunkt zu übergehen.

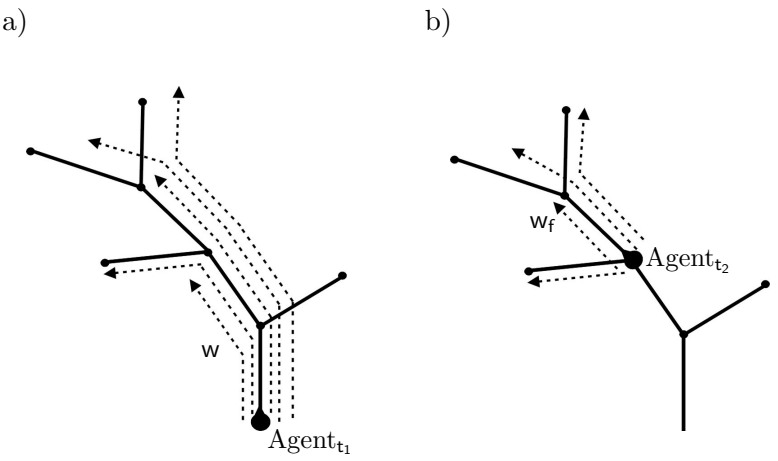


Abbildung 5.6: Auswahl eines *Fortsetzungspfades* für die Folgeaktion 3 (Beispiel für den Anweisungstyp GO): Bild a) zeigt die Pfade, die der Prozess der Koreferenzauflösung für die Ausführung der ursprünglichen Bewegungsaktion ermittelt hat (l_{wP}). Der gekennzeichnete Pfad w wird ausgewählt. b) Die Fortsetzungspfade von l_{wP} und w: Nach der Ausführung der Bewegung befindet sich der Agent am Endpunkt von w. Wird an dieser Stelle die Folgeaktion 3 aktiviert, werden alle Pfade ermittelt, die eine Fortsetzung des letzten Pfades w darstellen und Teilpfade der vorher ermittelten Alternativpfade sind (l_{fP}). Aus diesen wird der Pfad w_f ausgewählt, der als Argument der anschließenden Bewegung fungiert, und die Liste der Fortsetzungspfade aktualisiert.

pfaden wird dann durch den Prozess ‚Auswahl eines Pfades für die folgende Bewegungsaktion' ein Pfad ausgewählt und eine Aktion vom Typ follow mit dem Pfad als Argument ausgeführt. Danach wird die Liste der Fortsetzungspfade aktualisiert, indem die Fortsetzungspfade des ausgewählten Fortsetzungspfades ermittelt werden (in der ursprünglichen Liste von Pfaden).

Im Fall einer Anweisung vom Typ BE_AT wird die Liste der ursprünglichen Alternativpfade nicht durch den Prozess ‚Koreferenzauflösung' ermittelt, sondern durch den Prozess ‚Bestimmung der Pfade zu einer Region', der diejenigen Pfade aus dem Perzeptionsgraphen zurückgibt, die ihren Startpunkt in der aktuellen Position des Agenten und ihren Endpunkt in der Region haben. Die Fortsetzungspfade werden in diesem Fall mit Hilfe dieser Liste ermittelt.

Nach jeder Ausführung einer Bewegungsaktion vom Typ follow wird die Liste der Fortsetzungspfade aktualisiert. Findet die Bewegungsaktion im Kontext einer Folgeaktion statt, werden die Fortsetzungspfade in der ursprünglichen Liste von Alternativpfaden bestimmt. Anderenfalls ist die Bewegungsaktion vom Typ follow die erste primitive Aktion, die bei der Abarbeitung einer Anweisung ausgeführt wird, was nur bei der Abarbeitung von Anweisungen vom Typ GO und BE_AT geschehen kann. Hierbei wird die Liste der Fortsetzungspfade komplett neu ermittelt (und die alte Liste von Fortsetzungspfaden kann ‚vergessen' werden). Nach einer Ausführung einer Bewegungsaktion vom Typ turn wird die Liste der Fortsetzungspfade geleert. Hier macht es keinen Sinn, die Fortsetzungspfade der letzten Bewegungsaktion vom Typ follow weiter zu verwenden, da der Entscheidungspunkt erreicht wurde, an dem die

Drehung ausgeführt werden konnte. Nicht aktualisiert bzw. geleert wird die Liste von Fortsetzungspfaden jedoch bei der Abarbeitung einer Anweisung vom Typ VIEW oder BE_AT (wenn sich herausgestellt hat, dass sich der Agent in der Region befindet), da das Perzipieren eines Objektes nicht unbedingt bestimmt, dass die letzte Bewegung ausreichend war (da Objekte von mehreren Positionen aus gesichtet werden können). Ebenso verhält es sich mit dem Aspekt, ob der Agent sich in einer spezifizierten Region befindet: Auch hier kann die Feststellung des Agenten, er befinde sich in der Region, an mehreren Positionen getroffen werden (da Regionen ausgedehnt sind).

Die Folgeaktion 3 ist immer dann ausführbar, wenn die Liste von Fortsetzungspfaden nicht leer ist, d.h. es noch Pfade gibt, denen noch nicht gefolgt wurde und die (verknüpft mit früher gefolgten Pfaden) als potentielle Pfade für die letzte Bewegungsaktion vom Typ follow ermittelt wurden.

Die Folgeaktion 3 kann jedoch nicht angestoßen werden, wenn die Liste der Fortsetzungspfade leer ist. Dies kann passieren, wenn von Anfang an keine Fortsetzungspfade existieren, die letzte Bewegungsanweisung vom Typ turn war, oder aber die Folgeaktion 3 so oft angestoßen wurde, dass dem entferntesten Teilpfad, den der Agent ursprünglich perzipiert und koreferenziert hatte, schon gefolgt wurde. Da dieser keine Fortsetzungspfade besitzt, und die Liste der Fortsetzungspfade bei der Aktualisierung im Kontext einer Folgeaktion auch nicht um Teilpfade erweitert wird, die entlang weiterer (erst später perzipierter) Routensegmente verlaufen, ist die Liste schließlich leer.

Ein weiterer Typ einer Folgeaktion (Folgeaktion 4) wird angestoßen, wenn der Agent annimt, der Aktionsplan besitze eine Lücke. Dieses Verfahren wird in Abschnitt 5.10.1 *Test auf Lücken im Aktionsplan* vorgestellt.

5.7.2 Verwendung der Folgeaktionen

Der grundlegende Ablauf ist für die verschiedenen Anweisungstypen gleich. Bei den ersten zwei Durchläufen wird die Folgeaktion 1 ausgeführt, d.h. der Agent prüft nach Drehung und erneuter Perzeption, ob die Objekte, die dem Argument der folgenden Instruktionsanweisung entsprechen, nun in seinem Sichtfeld liegen: Zuerst wird die Folgeaktion 1a ausgeführt (Drehung um 30° im Uhrzeigersinn) und danach erneut eine Aktualisierung des internen Zustands vorgenommen. Ist die Vorbedingung der folgenden Anweisung nicht erfüllt, wird die Folgeaktion 1b ausgeführt (Drehung um 60° gegen den Uhrzeigersinn) und der interne Zustand aktualisert. Das Drehen des Agenten um einen vorgegebenen Winkel und erneutes Perzipieren der Umgebung ist den anderen Folgeaktionen zuerst vorzuziehen, da es den geringsten Aufwand erfordert (es muss kein Pfad für die Bewegungsaktion vom Typ turn ermittelt werden), und zum Erfolg führen kann, ohne dass der Agent seine Position verändert.

War das Umhersehen des Agenten erfolglos, muss eine der Folgeaktionen angestoßen werden, durch die der Agent seine Position verändert, d.h. durch die eine primitive Bewegungsaktion vom Typ follow angestoßen wird.

Hier ist die Folgeaktion 3 der Folgeaktion 2 vorzuziehen, da die Pfade, die für die primitive Bewegungsaktion in Frage kommen, Fortsetzungspfade des koreferenten Pfades sind (und verknüpft mit dem zuletzt gefolgtem Pfad einem Pfad entsprechen, der ebenfalls koreferent zu dem Argument der aktuellen Anweisung ist) und nicht zufällig ausgewählt werden.

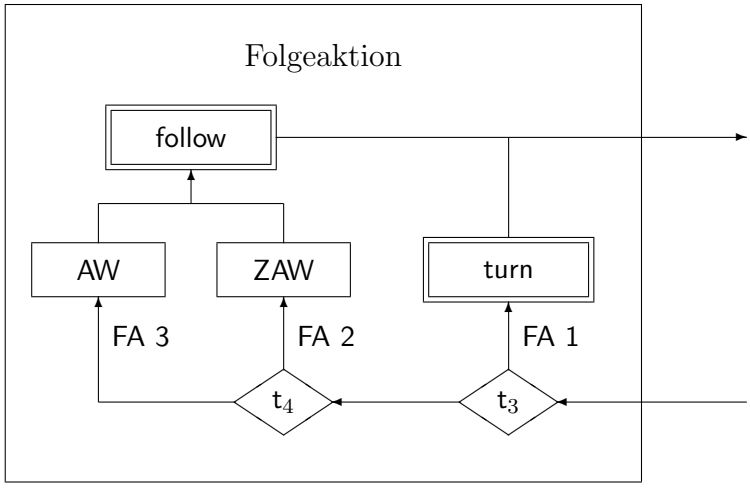


Abbildung 5.7: Der komplexe Prozess ‚Ausführung der Folgeaktion'. In der bedingten Verzweigung t_3 wird geprüft, ob der Zähler z_f kleiner als 3 ist. Falls ja, wird eine Folgeaktion vom Typ 1 ausgeführt (FA 1). Falls nicht, wird in der bedingten Verzweigung t_4 geprüft, ob die Liste l_{f_P} nicht leer ist. Ist sie leer, wird eine Folgeaktion vom Typ 2 angestoßen (FA 2), d.h. ein zufälliger Pfad ausgewählt (ZAW), dem dann durch die primitive Aktion follow gefolgt wird. Ist die Liste nicht leer, wird eine Folgeaktion vom Typ 3 angestoßen (FA 3). Hier wird zuerst aus der Liste von Fortsetzungspfaden einer ausgewählt (AW), der dann als Argument der Bewegungsaktion vom Typ follow fungiert.

Durch die Folgeaktion 3 wird die defensive Strategie der Pfadauswahl kompensiert (siehe Abschnitt 5.4.2).

Erst wenn die Liste der Fortsetzungspfade leer ist, wird die Folgeaktion 2 angestoßen (zufällige Exploration). Hierfür nutzt der Agent keine Informationen aus dem Instruktionsmodell, um den Pfad auszuwählen (deshalb *zufällige* Auswahl).

Die Abbildung 5.7 stellt den komplexen Prozess ‚Folgeaktion' dar (vgl. Abbildungen 5.3, 5.4 und 5.5, dort ist der Prozess ‚Folgeaktion' als *black box* dargestellt). Der Prozess wird angestoßen (eingehender Pfeil unten rechts), wenn die Prüfung der Vorbedingung der folgenden Anweisung ein negatives Ergebnis geliefert hat. Innerhalb des Prozesses wird in zwei bedingten Verzweigungen geprüft, welcher Typ von Folgeaktion ausgeführt wird. In den ersten beiden Durchläufen, nachdem der Agent seine Position verändert hat, d.h. nach der Ausführung einer Bewegungsaktion vom Typ follow, sieht der Agent sich um (Folgeaktion 1). Nach jedem dieser Durchläufe wird ein Zähler z_f inkrementiert. In der bedingten Verzweigung t_3 wird der Zähler abgefragt: Ist $\mathsf{z}_f = 1$, wird die Folgeaktion 1a ausgeführt und danach der Zähler inkrementiert. Ist $\mathsf{z}_f = 2$, wird die Folgeaktion 1b ausgeführt und danach der Zähler inkrementiert. Ist $\mathsf{z}_f = 3$, wird in der zweiten bedingten Verzweigung t_4 geprüft, ob die Liste der Fortsetzungspfade leer ist. Falls ja, wird der Prozess für die zufällige Auswahl eines Pfades angestoßen und danach die Bewegungsaktion vom Typ follow angestoßen. Ist die Liste der Fortsetzungspfade nicht leer, kann die Folgeaktion 3 ausgeführt werden. Nach Ausführung der Folgeaktionen 2 und 3 wird der Zähler z_f auf 1 zurückgesetzt, da der Agent seine Position verändert hat.

5.8 Algorithmen für die lokale Planung

5.8.1 Die Ablaufsteuerung

Der Algorithmus für die Ablaufsteuerung (Algorithmus 1) wird in der Abbildung 5.8 dargestellt. Bei der Abarbeitung der Instruktionsanweisungen werden die komplexen Prozesse ‚Aktualisierung des internen Zustands', ‚Ausführung der Aktion, und ‚Ausführung einer Folgeaktion' angestoßen (siehe Abschnitt 5.5), deren Prozeduren durch die Algorithmen 2 bis 4 beschrieben werden.

In den Zeilen 1 bis 7 findet die Initialisierung statt. Während der Initialisierung wird die Aktualisierung des internen Zustands angestoßen (Zeile 6) . Als Ergebnis liegt eine Liste koreferenzierter Objekte aus dem Perzeptionsgraphen vor (l_{o_P}), die bei den weiteren Instruktionsanweisungen (die einen Vorgänger besitzen) bei der Abarbeitung der vorherigen Anweisung in Zeile 13 bzw. 20 ermittelt wird (siehe Abschnitt 5.5).[18]

Die Zeilen 8 bis 26 beschreiben die Hauptschleife, die durchlaufen wird, bis die letzte Anweisung aus dem Aktionsplan erreicht wurde. Diese wird anders behandelt, da sie keinen Nachfolger besitzt (Zeile 28). In der Schleife werden zuerst die aktuelle Anweisung und deren Nachfolger aus dem Aktionsplan bestimmt (Zeile 10 und 11). Die aktuelle Anweisung bestimmt den Typ der primitiven Aktion, die der Agent zur Abarbeitung der aktuellen Anweisung auswählt. Als weiteres Argument erhält die Prozedur `Ausführung der Aktion` die Liste der Objekte, die während der Abarbeitung der vorherigen Instruktionsanweisung im Prozess der Koreferenzauflösung bei der Aktualisierung des internen Zustands ermittelt wurde.[19] Diese Liste enthält mindestens ein Objekt, da sonst die vorherige Anweisung nicht als erfolgreich abgearbeitet interpretiert und die aktuelle Anweisung zur Abarbeitung herangezogen worden wäre. In der Zeile 12 wird eine der aktuellen Anweisung entsprechende primitive Aktion ausgewählt und angestoßen, indem die Prozedur `Ausführung der Aktion` aufgerufen wird (Algorithmus 3). Diese Prozedur liefert als Rückgabe ggf. eine Liste von Fortsetzungspfaden (siehe Abschnitt 5.7.1).

Nach Ausführung der Aktion wird der interne Zustand aktualisiert (Zeile 13). Danach wird geprüft, ob die Vorbedingung der folgenden Anweisung erfüllt wird (Zeile 16). Ist die Vorbedingung nicht erfüllt, d.h. die Liste koreferenzierter Objekte $\mathtt{l}_{o_P}$ ist leer, wird die Schleife der Folgeaktion betreten, die erst verlassen wird, wenn die Vorbedingung erfüllt ist. In der Schleife wird in Zeile 18 die Prozedur `Ausführung einer Folgeaktion` (Algorithmus 4) aufgerufen, die als Eingabe den Zähler $\mathtt{z}_f$, die aktuelle Anweisung und ggf. eine Liste der Fortsetzungspfade erhält (siehe Abschnitt 5.7). Nach der Ausführung der primitiven Aktion, die durch die Prozedur `Ausführung einer Folgeaktion` angestoßen wurde, wird die Aktualisierung des internen Zustands vorgenommen (Zeile 20).

Nach erfolgreicher Abarbeitung einer Anweisung (d.h. wenn der Test in Zeile 16 ergibt, dass $\mathtt{l}_{o_P}$ nicht leer ist), kann der Zähler der Instruktionsanweisungen inkrementiert und der Zähler der Folgeaktionen zurückgesetzt werden (Zeile 24 und 25).

[18] An dieser Stelle wird kein Test wie in Zeile 15 vorgenommen, da angenommen wird, dass im Initialzustand die Vorbedingung der ersten Instruktionsanweisung erfüllt ist. Der Algorithmus kann aber modifiziert werden, so dass auch schon bei der Abarbeitung der ersten Anweisung ein Test erfolgt und ggf. Folgeanweisungen angestoßen werden.

[19] Bzw. bei der ersten Instruktionsanweisung bei der Initialisierung.

Algorithmus 1: Ablaufsteuerung
Eingaben:
Der Aktionsplan - eine Liste von Anweisungen [anw_1,...,anw_m]
Das initiale Umgebungsmodell (Instruktionsgraph)
l_{o_P} - die Liste koreferenzierter Objekte aus dem P-Netz

```
1:   /* Initialisierung */
2:   i := 1; /* i: Index der Instruktionsanweisungen */
3:   z_f := 1; /* z_f: Zähler für die Folgeaktion */
4:   l_fP := [] /* l_fP: Liste von Fortsetzungspfaden ist zu Beginn leer */
5:   a_anw := anw_1; /* die erste Anweisung aus dem Aktionsplan */
6:   l_oP := Aktualisierung des internen Zustands(a_anw.arg);
7:       /* Algorithmus 2 */
8:   while (i <= m - 1) do
9:       {
10:        a_anw := anw_i; /* die aktuelle Anweisung */
11:        f_anw := anw_i+1; /* die folgende Anweisung */
12:        l_fP := Ausführung der Aktion(a_anw,l_oP); /* Algorithmus 3 */
13:        l_oP := Aktualisierung des internen Zustands(f_anw.arg);
14:           /* Algorithmus 2 */
15:        /* Prüfe Vorbedingung der folgenden Anweisung */
16:        while (l_oP ist leer) do
17:           { /* Vorbedingung ist nicht erfüllt */
18:              l_fP,z_f := Ausführung einer Folgeaktion(z_f,a_anw,l_fP);
19:                 /* Algorithmus 4 */
20:              l_oP := Aktualisierung des internen Zustands(f_anw.arg);
21:                 /* Algorithmus 2 */
22:           }
23:        /* Vorbedingung ist erfüllt */
24:        i := i+1; /* nächste Instruktionsanweisung */
25:        z_f := 1; /* Zähler wird zurückgesetzt */
26:     }
27:  /* Abarbeitung der letzten Anweisung */
28:  Ausführung der Aktion(f_anw,l_oP); /* Algorithmus 3 */
```

Abbildung 5.8: Algorithmus für die Ablaufsteuerung.

Nach Abarbeitung der letzten Instruktionsanweiung (Zeile 28) wird angenommen, dass der Agent sein Ziel erreicht hat.

5.8.2 Aktualisierung des internen Zustands

Algorithmus 2: `Aktualisierung des internen Zustands`

`Eingaben:`

`f_anw.arg - das Argument der folgenden Anweisung`

`Rückgaben:`

$\texttt{l}_{o_P}$ `- die Liste koreferenzierter Objekte aus dem P-Netz`

```
1:  Ausführung der Perzeptionsaktion;
2:  Verarbeitung der Perzeption;
3:  Selbstlokalisation(f_anw.arg);
4:  l_oP := Koreferenzauflösung(f_anw.arg);
```

Abbildung 5.9: Algorithmus für die Aktualisierung des internen Zustands.

Der Algorithmus 2 (siehe Abb. 5.9) beschreibt die Prozedur, die die Aktualisierung des internen Zustands vornimmt. In ihr werden eine primitive Perzeptionsaktion (Zeile 1), sowie drei Hilfsprozesse angestoßen (Zeile 2 bis 4).

Durch den Prozess `Verarbeitung der Perzeption` werden die Objekte aus der aktuell wahrgenommenen Situation als Referenzobjekte und Relationen im Perzeptionsmodell repräsentiert (siehe Abschnitt 3.5.1 *Verarbeitung der Perzeption*).

Der Prozess `Selbstlokalisation` bestimmt die aktuelle Position des Agenten im Umgebungsmodell und erhält als Eingabe das Argument der folgenden Instruktionsanweisung (siehe Abschnitt 5.4.1 *Pragmatische Annahmen für die Selbstlokalisation*).

Der Prozess `Koreferenzauflösung` ermittelt schließlich die Objekte aus dem Perzeptionsgraphen, die für eine Koreferenz mit dem Argument der folgenden Anweisung (`f_anw.arg`) in Frage kommen (siehe Abschnitt 3.5.2 *Auflösung von Koreferenzen*).

5.8.3 Auswahl und Ausführung von primitiven Bewegungsaktionen

Die Prozedur `Ausführung der Aktion`, die für die Auswahl und Ausführung der primitiven Bewegungsaktionen verantwortlich ist, wird durch den Algorithmus 3 in der Abbildung 5.10 dargestellt. Entscheidende Parameter sind der Typ der aktuellen Anweisung (`anw.typ`) und die Liste koreferenzierter Objekte ($\texttt{l}_{o_P}$). Der Anweisungstyp bestimmt, welche Art von primitiver Aktion in Frage kommt.

Ist die aktuelle Anweisung vom Typ GO, wird eine Bewegungsaktion vom Typ `follow` angestoßen (Zeile 1 bis 7). Hierfür ermittelt die Prozedur `Auswahl des Pfades` den zu folgenden Pfad `w` (siehe Abschnitt 5.4.2). Hiernach wird die Liste von Pfaden $\texttt{l}_{f_P}$ bestimmt, die einen Fortsetzungspfad zu `w` darstellen. Diese Liste wird bei der evtl. anzustoßenden Folgeaktion benötigt (siehe Abschnitt 5.7). Nach Ausführung der primitiven Aktion wird die Prozedur `Ausführung der Aktion` verlassen und die Prozedur zur Aktualisierung des internen Zustands in Algorithmus 1 angestoßen (Zeile 13 in Algorithmus 1).

Algorithmus 3: Ausführung der Aktion

Eingaben:

anw - die aktuelle Anweisung

l_{oP} - die Liste koreferenzierter Objekte

Rückgaben:

l_{fP} - die Liste von Fortsetzungspfaden (nur nach Aktualisierung)

```
1:   if (anw.typ == GO)
2:   then {
3:            w := Auswahl des Pfades(l_oP);
4:            l_fP := Fortsetzungspfade(l_oP,w);
5:            follow(w);
6:            /* Rückkehr zu Alg. 1 */
7:        }
8:   if (anw.typ == CH_ORIENT)
9:   then {
10:           w := Auswahl des Pfades(l_oP);
11:           turn(w);
12:           l_fP := []; /* Liste ist leer */
13:           /* Rückkehr zu Alg. 1 */
14:       }
15:  if (anw.typ == BE_AT)
16:  then {
17:           if (Position des Agenten bei anw.arg)
18:           then {
19:                   /* Abbruch & Rückkehr zu Alg. 1 */
20:               }
21:           else {
22:                   l_rP := BestPfadeZuReg(anw.arg);
23:                     /* l_rP : die Liste von Pfaden zu Region r */
24:                   w := Auswahl des Pfades(l_rP);
25:                   l_fP := Fortsetzungspfade(l_rP,w);
26:                   follow(w);
27:                   /* Rückkehr zu Alg. 1 */
28:               }
29:       }
30:  if (anw.typ == VIEW)
31:  then {
32:           /* Abbruch & Rückkehr zu Alg. 1 */
33:       }
```

Abbildung 5.10: Algorithmus für die Auswahl und Ausführung der Aktion, die der aktuellen Anweisung entspricht.

Ist die aktuelle Anweisung vom Typ CH_ORIENT, wird nach Auswahl des Pfades eine primitive Aktion vom Typ `turn` ausgeführt (Zeile 8 bis 14). Hier wird die Liste der Fortsetzungspfade geleert (siehe Abschnitt 5.7).

Anweisungen vom Typ BE_AT (Zeile 15 bis 29) erfordern einen zusätzlichen Test, durch den der Agent feststellt, wie die Anweisung in der aktuellen Situation zu interpretieren ist: In Zeile 17 wird geprüft, ob der Agent sich in der durch das Argument der aktuellen Anweisung (`anw.arg`) spezifizierten Region befindet. Ist das der Fall, muss keine primitive Bewegungsaktion, die zu der Region führt, angestoßen werden, und es wird zu Algorithmus 1 (Zeile 13) zurückgekehrt. Anderenfalls ermittelt der Agent mittels der Methode `BestPfadeZuReg(anw.arg)` eine Liste von Pfaden $\mathtt{l}_{w_P}$, die von der aktuellen Position des Agenten zu der Region hinführen (siehe Abschnitt 5.3.2). Nach der Auswahl eines dieser Pfade wird dann die primitive Bewegungsaktion vom Typ `follow` angestoßen. Da der Agent eine primitive Bewegungsaktion vom Typ follow ausgeführt hat, wird die Liste der Fortsetzungspfade neu bestimmt (Zeile 25).

Bei einer Anweisung vom Typ VIEW wurde die intendierte Handlung, nach einem Objekt Ausschau zu halten, schon bei der Abarbeitung der letzten Anweisung ausgeführt, so dass keine weitere primitive Aktion angestoßen werden muss, und die Prozedur `Ausführung der Aktion` direkt verlassen werden kann.[20]

5.8.4 Ausführung einer Folgeaktion

Konnten bei der Aktualisierung des internen Zustands durch den Prozess der Koreferenzauflösung keine koreferenzierbaren Objekte im Perzeptionsgraphen ermittelt werden, ist die Vorbedingung der folgenden Instruktionsanweisung nicht erfüllt, und es wird die Schleife der Folgeaktionen betreten. In der Schleife wird in Zeile 17 des Algorithmus 1 die Prozedur `Ausführung einer Folgeaktion` angestoßen. Die Prozedur erhält als Eingabe den Zähler $\mathtt{z}_f$, der mitbestimmt, welche Art von Folgeaktion auszuführen ist, den Typ der aktuellen Anweisung `anw.typ`, sowie eine Liste von Fortsetzungspfaden $\mathtt{l}_{f_P}$ (die auch leer sein kann).

Handelt es sich um die erste Folgeaktion, die nach der Ausführung einer primitiven Bewegungsaktion vom Typ follow angestoßen wird, nimmt der Agent eine Drehung um 30° im Uhrzeigersinn vor (Folgeaktion 1a). Hiernach wird $\mathtt{z}_f$ inkrementiert und die Prozedur verlassen, um in Zeile 20 des Algorithmus 1 eine erneute Aktualisierung des internen Zustands vorzunehmen (Zeile 1 bis 6).

Beim zweiten Durchlauf nach Ausführung einer Bewegungsaktion vom Typ follow (Zeile 7 bis 12) wird die Folgeaktion 1b durchgeführt. Diese entspricht der Folgeaktion 1a, nur dreht sich der Agent diesmal um 60° gegen den Uhrzeigersinn.

Die weiteren Folgeaktionen orientieren sich daran, ob eine nichtleere Liste von Fortsetzungspfaden vorliegt (Zeile 13 bis 30). Ist diese Liste nicht leer, kann die Folgeaktion 3 ausgeführt werden (Zeile 14 bis 22). Hierfür wählt der Prozess `Auswahl des Pfades` einen der Fortsetzungspfade aus (Zeile 17), der als Argument der primitiven Bewegungsaktion vom Typ follow fungiert (Zeile 19). In Zeile 18 wird die Liste der Fortsetzungspfade aktualisiert,

[20] An dieser Stelle, wie auch bei der Rückkehr zu Algorithmus 1 in der Zeile 18 des Algorithmus 3 können bei der darauffolgenden Aktualisierung des internen Zustands die Perzeptionsaktion sowie die folgende Verarbeitung ausgelassen, d.h. die Zeilen 1 und 2 in Algorithmus 2 übersprungen werden (siehe Abschnitt 5.6). Aus Gründen der Übersichtlichkeit wurde dies in den Algorithmen nicht berücksichtigt.

Algorithmus 4: Ausführung einer Folgeaktion

```
Eingaben:
z_f      - Zähler für die Folgeaktionen bei der Abarbeitung
           der aktuellen Anweisung
anw.typ  - der Typ der aktuellen Anweisung
l_fP     - die Liste der Fortsetzungspfade
Rückgaben:
z_f      - Zähler für die Folgeaktionen bei der Abarbeitung (aktualisiert)
l_fP     - die Liste der Fortsetzungspfade (aktualisiert)
```

```
1:   if (z_f = 1) /* erster Durchlauf nach Aktion follow */
2:   then {
3:            turn(30); /* Folgeaktion 1a */
4:            z_f := 2;
5:            /* Rückkehr zu Alg. 1 */
6:        }
7:   else if (z_f = 2) /* zweiter Durchlauf nach Aktion follow */
8:   then {
9:            turn(-60); /* Folgeaktion 1b */
10:           z_f := 3;
11:           /* Rückkehr zu Alg. 1 */
12:       }
13:  else { /* dritter Durchlauf nach Aktion follow */
14:           if (l_fP ist nicht leer)
15:           then { /* es gibt (noch) Fortsetzungspfade */
16:                   /* Folgeaktion 3 */
17:                   w := Auswahl des Pfades(l_fP);
18:                   l_fP := Fortsetzungspfade(l_fP,w);
19:                   follow(w);
20:                   z_f := 1;
21:                   /* Rückkehr zu Alg. 1 */
22:                }
23:           else { /* l_fP ist leer */
24:                   /* Folgeaktion 2 */
25:                   w := Wähle zufälligen Pfad;
26:                   follow(w);
27:                   z_f := 1;
28:                   /* Rückkehr zu Alg. 1 */
29:                }
30:       }
```

Abbildung 5.11: Algorithmus für die Auswahl und Ausführung einer Folgeaktion.

indem die Fortsetzungspfade des ausgewählten Fortsetzungspfades ermittelt werden. Nach Ausführung der Bewegungsaktion kann erneut eine Aktualisierung des internen Zustands vorgenommen werden (Rückkehr zu Zeile 20 in Algorithmus 1).

Ist die Liste der Fortsetzungspfade leer, führt der Agent die Folgeaktion 2 aus, die in der zufälligen Exploration der Umgebung besteht (Zeile 23 bis 29). Durch den Prozess `Wähle zufälligen Pfad` wird ein geeigneter Pfad aus der Umgebung für die Exploration ausgewählt (siehe Abschnitt 5.7).[21]

Sowohl nach der Ausführung der Fortsetzungsaktion 2, wie auch nach der Fortsetzungsaktion 3 hat der Agent seine Position in der Umgebung verändert (da er eine primitive Bewegungsaktion vom Typ follow ausgeführt hat). Bei der Ausführung dieser Folgeaktionen wird z_f auf 1 zurückgesetzt, da der Agent sich in den nächsten beiden Folgeaktionen wieder umsehen soll (Folgeaktion 1).

5.9 Ein Beispiel

In dem hier beschriebenen Beispiel hat der Geometrische Agent die Aufgabe, von der Mensa zu Haus P3 zu gelangen. Die Tabelle 5.9 zeigt eine natürlichsprachliche Routeninstruktion, die eine Route von der Mensa zu Haus P3 beschreibt[22], sowie den Aktionsplan, der aus der Verarbeitung der Routeninstruktion resultiert. Der Instruktionsgraph, der die geometrischen Informationen aus der Routeninstruktion repräsentiert, ist hier nicht dargestellt. Die Abbildung 5.12 zeigt die beschriebene Route.

Beispiel 1: Route von der Mensa zu Haus P3

	Routeninstruktion	**Aktionsplan**
a)	Geh aus der Mensa.	r9: !GO(w_5)
b)	Dreh dich nach links.	r21: !CH_ORIENT(w_6)
c)	Dann siehst du Haus C.	r33: !VIEW(b_1)
d)	Geh zwischen Haus B und Haus C durch.	r48: !GO(w_7)
e)	Geh rechts.	r106: !GO(w_8)
f)	Geh links.	r152: !GO(w_9)
g)	Dann stehst du vor Haus P3.	r192: !BE_AT(b_2)

Tabelle 5.4: Routeninstruktion für eine Route von der Mensa zu Haus P3 und resultierender Aktionsplan.

In der Abbildung 5.13 werden die einzelnen Schritte während der Navigation veranschaulicht. Die Ablaufsteuerung beginnt mit der Initialisierung, in der die erste Instruktionsanweisung aus dem Aktionsplan herangezogen wird (!GO(w_5)). Danach wird der interne Zustand

[21] Das hier beschriebene Verfahren entspricht der Folgeaktion 2a, bei der der Agent nach der Bewegungsaktion von der neuen Position aus weitere Folgeaktionen ausführt.

[22] Diese Routeninstruktion habe ich zu Demonstrationszwecken erstellt; sie stammt nicht aus dem Korpus.

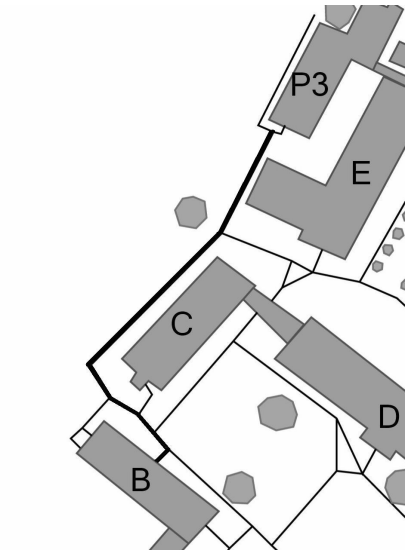


Abbildung 5.12: Die Route aus dem Beispiel.

des Agenten aktualisiert. Bei der Aktualisierung nimmt der Agent die Umgebung wahr und versucht mit Hilfe der Koreferenzauflösung die Pfade aus dem Perzeptionsgraphen zu ermitteln, die dem Argument der ersten Anweisung w_5 entsprechen. Dies sind alle Pfade, die aus der Mensa herausführen. Aufgrund der defensiven Strategie der Pfadauswahl wählt der Agent den Pfad aus, der in der Abbildung 5.13a hervorgehoben dargestellt ist. Da die aktuelle Anweisung vom Typ GO ist, führt der Agent eine primitive Bewegungsaktion vom Typ follow aus, für die als Argument der ausgewählte Pfad fungiert. Nach Ausführung der primitiven Bewegungsaktion befindet sich der Agent am Endpunkt des Pfades. In dieser Position aktualisiert der Agent seinen internen Zustand, um zu ermitteln, ob er das Argument der folgenden Instruktionsanweisung ($!\mathsf{CH\_ORIENT}(\mathsf{w}_6)$) koreferenzieren kann. Hierfür muss er in seiner Umgebung einen Pfad sehen können, der den Startpunkt in der aktuellen Position des Agenten hat und dessen Orientierung nach links weist (relativ zu dem Referenzsystem des Agenten).

Da der Agent einen Pfad hierfür koreferenzieren kann, kann die Anweisung ($!\mathsf{GO}(\mathsf{w}_5)$) als erfolgreich abgearbeitet interpretiert werden. Der Agent führt als nächstes eine primitive Bewegungsaktion vom Typ turn aus, da die aktuelle Anweisung vom Typ CH_ORIENT ist (Abbildung 5.13b/c), und die als Argument den Pfad aus dem Perzeptionsmodell erhält, der nach links zeigt. Nach der Bewegungsaktion aktualisiert der Agent den internen Zustand. Dabei wird das Argument der folgenden Instruktionsanweisung $!\mathsf{VIEW}(\mathsf{b}_1)$ herangezogen, das die Landmarke ‚Haus C' repräsentiert. Da der Agent in der aktuellen Situation das Haus perzipieren und koreferenzieren kann, gilt die Anweisung $!\mathsf{CH\_ORIENT}(\mathsf{w}_6)$ als erfolgreich abgearbeitet.

Bei der Abarbeitung einer Instruktionsanweisung vom Typ VIEW wird keine primitive Bewegungsaktion angestoßen, sondern gleich eine Aktualisierung des internen Zustands vorgenommen. In diesem konkreten Fall prüft der Agent, ob der Perzeptionsgraph Pfade enthält, die er mit dem Argument w_7 aus der folgenden Anweisung $!\mathsf{GO}(\mathsf{w}_7)$ koreferenzieren kann. Der

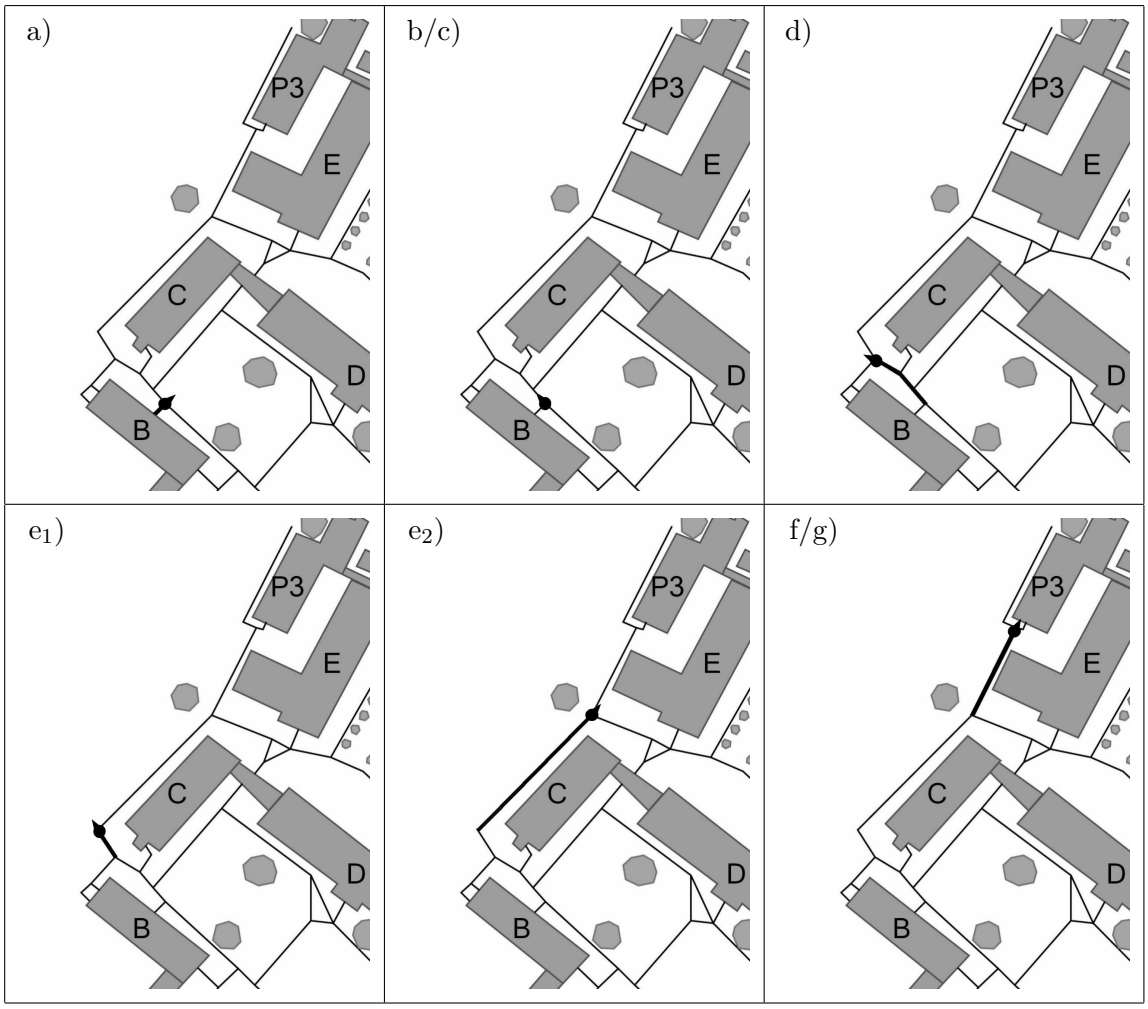


Abbildung 5.13: Abarbeitung der Routeninstruktion aus dem Beispiel. Die Abbildungen veranschaulichen, welche Aktionen der Agent ausführt, um die jeweils aktuelle Instruktionsanweisung aus dem Aktionsplan abzuarbeiten, und welche Pfade er ggf. dabei auswählt.

Pfad w_7 aus dem Instruktionsmodell wird im Instruktionsgraphen spezifiziert: er muss seinen Startpunkt in der aktuellen Situation des Agenten und einen internen Punkt (das muss nicht unbedingt ein Entscheidungspunkt sein) in der Region haben, die zwischen Haus B und Haus C liegt. Zusätzlich darf sein Endpunkt nicht in dieser Region liegen (da der Agent zwischen den Häusern *durchgehen* soll). Auch hier kann der Agent mehrere Pfade, die diese Bedingungen erfüllen, koreferenzieren, aus denen er dann den in Abbildung 5.13d hervorgehobenen Pfad auswählt. Nach der primitiven Bewegungsaktion (Folgen des Pfads) befindet sich der Agent am Endpunkt des Pfads.

Bei der anschließenden Aktualisierung des internen Zustands versucht der Agent die Pfade in seiner Umgebung zu ermitteln, die nach rechts führen. Hierbei kann er zwei Pfade erfolgreich koreferenzieren: Der erste Pfad wird in der Abbildung 5.13e_1 dargestellt. Der zweite koreferenzierte Pfad setzt sich aus dem Pfad aus Abbildung 5.13e_1, gefolgt von dem Pfad aus Abbildung 5.13e_2 zusammen. Aufgrund der defensiven Strategie der Pfadauswahl wählt

er den Pfad aus Abbildung 5.13e$_1$ und folgt diesem in der anschließenden primitiven Bewegungsaktion, um die Anweisung !GO(w_8) abzuarbeiten. Danach befindet sich der Agent am Endpunkt des Pfades. Die Aktualisierung des internen Zustands in dieser Situation ergibt jedoch, dass die Vorbedingung der folgenden Instruktionsanweisung !GO(w_9) nicht erfüllt ist: Der Agent kann keinen Pfad ausmachen, der in seiner Position beginnt und nach links führt (Abbildung 5.13e$_1$).

An dieser Stelle wird eine Folgeaktion angestoßen. Da auch nach zweimaligem Umhersehen (Folgeaktion 1a und 1b) kein entsprechender Pfad koreferenziert werden kann, wird eine Folgeaktion angestoßen, durch die der Agent seine Position verändert (Folgeaktion 2 oder 3). Da die Liste von Fortsetzungspfaden nicht leer ist, sondern einen Pfad beinhaltet, führt der Agent die Fortsetzungsaktion 3 aus, die im Folgen dieses Fortsetzungspfades besteht (Abbildung 5.13e$_2$).[23]

Nach Ausführung der Folgeaktion wird erneut der interne Zustand des Agenten aktualisiert und dabei geprüft, ob die Vorbedingung der folgenden Anweisung !GO(w_9) erfüllt ist, d.h. (mindestens) ein Pfad ermittelt werden kann, der nach links führt (der Agent befindet sich jetzt in der Position, die in der Abbildung 5.13e$_2$ dargestellt wird). Hier liefert der Prozess der Koreferenzauflösung wieder mehrere Pfade als Ergebnis, aus denen der Agent den in Abbildung 5.13f/g hervorgehobenen Pfad auswählt. Hiernach wird geprüft, ob der Agent das Argument der folgenden Anweisung !BE_AT(b_2) koreferenzieren kann, d.h. ob er die Landmarke ‚Haus P3' sehen und erkennen kann. Dies ist problemlos möglich, so dass nun die Anweisung !BE_AT(b_2) abgearbeitet werden kann.

Bei der Abarbeitung wird zuerst geprüft, ob der Agent sich in der Region befindet (die dadurch gekennzeichnet ist, dass sie sich bei Haus P3 befindet). Da der Test ein positives Ergebnis liefert, gilt die Anweisung als erfolgreich abgearbeitet, und der Navigationsdurchlauf ist beendet.

5.10 Weitere Verfahren für die Ablaufsteuerung

Die im folgenden vorgestellten Verfahren werden hier nur skizzenhaft vorgestellt. Die Entwicklung eines detaillierten Verfahrens sowohl für die Handhabung von Lücken, als auch für das Verhalten, das aktiviert wird, wenn der Agent annimmt, er habe sich verlaufen, kann hier nicht geleistet werden. In beiden Verfahren erfolgt eine hohe Interaktion (und ggf. Revision) der CRIL-Netze, die nicht direkt im Fokus dieser Arbeit liegen.

5.10.1 Test auf Lücken im Aktionsplan

Im Aktionsplan können sogenannte Lücken auftreten, wenn der Instruktor eine während der Navigation zu tätigende Handlung in der natürlichsprachlichen Routeninstruktion in deklarativer Art beschreibt, so dass der Ausdruck bei der Verarbeitung der Routeninstruktion nicht als imperative Anweisung interpretiert wird und keine entsprechende Instruktionsanweisung erzeugt und im Aktionsplan abgelegt wird.

[23] In dieser Situation hätte die Ausführung der Fortsetzungsaktion 2 zu demselben Verhalten geführt: Da der Pfad, der in der Abbildung 5.13e$_2$ hervorgehoben ist, der einzige Pfad ist, der für eine zufällige Exploration in Frage kommt, wäre der Agent ebenfalls diesem Pfad gefolgt.

Aktionen können z.B. implizit in deklarativer Form formuliert werden: So kann z.B. die Aussage „Zwischen Haus B und Haus C befindet sich ein Weg.“ auch als die Anweisung, diesem Weg zu folgen, verstanden werden (vgl. Abschnitt 3.3.4 *Imperative Anteile des Instruktionsmodells: der Aktionsplan*).

Die deklarativ formulierte Handlung in dem oben genannten Beispiel beschreibt, wie der Agent zu dem Entscheidungspunkt kommt, an dem er die folgende Handlung ausführen kann. Da diese Handlung nicht als Instruktionsanweisung im Aktionsplan repräsentiert wird, hat der Aktionsplan eine Lücke zwischen der letzten und der folgenden Instruktionsanweisung. Für den Geometrischen Agenten besteht somit das Problem, diese Lücke zu schließen.

Hierfür muss er eine derartige Lücke erkennen können. Werden folgende Bedingungen allesamt erfüllt, ist das ein Indiz dafür, dass ein Lücke vorliegen könnte (aber nicht muss!):

- Die Vorbedingung der folgenden Anweisung ist nicht erfüllt. Ist sie erfüllt, liegt auch keine Lücke vor, da der Agent ‚weiss, was zu tun ist‘.
- Es gibt einen Pfad w_{l_I} im Instruktionsgraphen, der
  - nicht als Argument einer Instruktionsanweisung fungiert
  - in der aktuellen Situation erfolgreich identifiziert werden kann, d.h. es gibt einen oder mehrere Pfade im Perzeptionsgraphen, die mit dem Pfad w_{l_I} koreferenziert werden können.
- Desweiteren kommen nur diejenigen Pfade aus dem Perzeptionsgraphen als Hinweis auf eine Lücke in Frage, die folgende Bedingung erfüllen: Der Agent befindet sich entweder im Startpunkt des Pfades oder kann einen Pfad von seiner aktuellen Position zu dem Startpunkt ausmachen. Dadurch werden Pfade ausgeschlossen, die unerreichbar sind, wie z.B. die Überführung von Haus D zu Haus F, die in einer Routeninstruktion des Korpus genannt wird: „Wenn du beim Pförtner stehst, dann siehst du das höchste Gebäude auf dem Gelände, Haus F. Von Haus F führt im ersten Stock ein Übergang zu Haus D. Gehe zuallererst zwischen Haus D und F unter dem Übergang durch.“.

Die erste Bedingung lautet, dass die Vorbedingung der folgenden Anweisung nicht erfüllt ist. Daher werden die weiteren Bedingungen erst überprüft, wenn der Prozess der Folgeaktion angestoßen wird, und auch erst nach der zweimaliger Ausführung der Folgeaktion 1 (Umhersehen), da der Test weitere rechenaufwändige Prozesse (z.B. den Prozess der Koreferenzauflösung) anstößt.

Sind die Bedingungen erfüllt, nimmt der Agent an, dass eine Lücke vorliegt und führt die Folgeaktion 4 aus.

Die Folgeaktion 4

Aus der Liste $\mathsf{l}_{w_l P}$ der Pfade, die erfolgreich mit dem Pfad w_{lI} aus dem Instruktionsmodell koreferenziert wurden, wird durch den Prozess ‚Auswahl eines Pfades bei Bewegungsaktionen‘ einer ausgewählt (w_{lP}). Eine der oben genannten Bedingungen ist, dass der Agent sich entweder im Startpunkt des ausgewählten Pfades befindet oder einen Pfad zu dem Startpunkt in der Umgebung ausmachen kann. Befindet er sich im Startpunkt, kann direkt eine primitive

Bewegungsaktion vom Typ follow ausgeführt werden, die den Pfad w_{lP} als Argument erhält. Danach wird die Liste der Fortsetzungspfade ermittelt (siehe Abschnitt 5.7 *Folgeaktionen*), die Folgeaktion verlassen und eine Aktualisierung des internen Zustands vorgenommen.

Befindet sich der Agent nicht im Startpunkt des Pfades w_{lP}, muss er einen weiteren Pfad ermitteln, der von seiner aktuellen Position zu dem Startpunkt führt, und diesem in einer anschließenden Bewegungsaktion folgen. Danach kann er wie oben beschrieben dem Pfad w_{lP} folgen und den Prozess der Folgeaktion verlassen.

5.10.2 Problemfall: der Agent hat sich verlaufen

Aufgrund der Unsicherheit bei der Navigation (die mitunter in der Unterspezifiziertheit der Routeninstruktion und der Ambiguität des Ergebnisses der Koreferenzauflösung begründet ist) besteht für den Geometrischen Agenten prinzipiell die Gefahr, dass er sich verläuft, d.h. an einem Entscheidungspunkt eine falsche Entscheidung trifft und dadurch einer falschen Teilroute folgt.

Um nicht endlos ‚umherzuirren', muss der Agent ein Verfahren besitzen, durch das er zum einen die Problemsituation erkennen kann, und das zum anderen dafür sorgt, dass der Agent zu einem ‚sicheren' Teil der Route zurückkehrt.

An dieser Stelle wird kein ausgearbeitetes Verfahren vorgestellt. Zwei Beispielverfahren sollen zeigen, welche Möglichkeiten es gibt, dem Problem gerecht zu werden.

Ein Indiz dafür, dass er sich verlaufen hat, ist die wiederholte Ausführung einer Folgeaktion: Wurden schon mehrere Folgeaktionen erfolglos ausgeführt, d.h. wurde nach deren Ausführung noch immer nicht die Vorbedingung der folgenden Instruktionsanweisung erfüllt, könnte dies darauf hindeuten, dass der Agent zu einem früheren Zeitpunkt einer falschen Teilroute gefolgt ist und auch durch das wiederholte Ausführen von Folgeaktionen nicht den Entscheidungspunkt erreichen kann, an dem er die nächste Instruktionsanweisung abarbeiten kann.

Hierfür wird in dem oben vorgestellten Verfahren durch einen zusätzlichen Zähler mitgezählt, wie oft eine Folgeaktion bei der Abarbeitung einer Instruktonsanweisung angestoßen wurde. Wird ein Schwellwert erreicht, wird angenommen, dass der Agent sich verlaufen hat, und es wird ein Verfahren zur Kompensation des Problems angestoßen. Zwei Verfahren bieten sich hierfür an:

Rückkehr zu einem früheren Entscheidungspunkt

Der Agent kehrt zu einem früheren Entscheidungspunkt zurück und wählt an dieser Stelle eine andere Route. Hierfür muss er verschiedene Informationen über die schon abgelaufene Route in einem Gedächtnis behalten:

- Er muss sich die **Entscheidungspunkte** merken, an denen er eine Entscheidung bezüglich der nachfolgenden Route getroffen hat.
- Er muss sich merken, welche Entscheidungen er an welchen Entscheidungspunkten getroffen hat, d.h. welche **Pfade** er ausgewählt hat, so dass er bei einem erneuten Versuch eine andere Entscheidung treffen kann.

- Um einen geeigneten Entscheidungspunkt für einen Neuversuch zu ermitteln, ist es sinnvoll, die **Ergebnisse der Koreferenzauflösung** zu speichern. Für eine Rückkehr bieten sich besonders die Entscheidungspunkte an, in denen durch den Prozess der Koreferenzauflösung entweder relativ niedrigre Koreferenzwerte ermittelt wurden, oder die einzelnen Pfade, die der Prozess als Ergebnis geliefert hat, einen vergleichbar hohen Koreferenzwert besitzen. In beiden Fällen ist die Chance, dass der Agent einem falschem Weg gefolgt ist, höher.

Rückkehr zum Startpunkt der Route

Eine andere Möglichkeit besteht darin, den aktuellen Durchlauf komplett abzubrechen und einen neuen Durchlauf vom Startpunkt aus zu starten. Dies macht nur Sinn, wenn durch die Veränderung von Parametern, die das Verhalten des Agenten steuern, ein anderer Ablauf möglich wird, d.h. der Agent andere Entscheidungen in den vergleichbaren Situationen treffen kann.

Folgende Parameter bieten sich für eine Veränderung an:

- Der **Schwellwert der Koreferenzauflösung**: Kann der Agent in bestimmten Situationen keine Koreferenzen herstellen, kann dies an einem zu niedrigen Schwellwert der Koreferenzauflösung liegen. Da dieser Schwellwert festlegt, ab welchem Ähnlichkeitsmaß zwei Teilnetze als koreferent gelten, wird durch einen niedrigeren Schwellwert die Zahl potentieller Kandidaten für eine Koreferenz erhöht. Dies kann dazu führen, dass das Verfahren zur Koreferenzauflösung in einem Fall (höherer Schwellwert) keine koreferenten Objekte im Perzeptionsgraphen ausmachen kann, da alle Ähnlichkeitswerte der *matches* unter dem Schwellwert liegen. Wird der Schwellwert abgesenkt, so dass er knapp unter einem auftretenden Ähnlichkeitswert liegt, liefert der Prozess hingegen (mindestens) ein positives Ergebnis.

- Das Verfahren zur **Auswahl eines Pfades aus Alternativpfaden**: Im Verfahren zur Pfadauswahl werden pragmatische Annahmen getroffen, deren Modifizierung sinnvoll sein könnte. Ein wesentliches Auswahlkriteriun ist der Ähnlichkeitswert, der im Prozess der Koreferenzauflösung ermittelt wird. Da eine Interpretation des Ähnlichkeitsmaßes schwierig ist ([Helwich 2003]), kann erst nach einer Testphase beurteilt werden, welche Sortierung und die damit verbundene Auswahl eines Pfades als adäquat einzustufen ist.

- Der **Schwellwert der Folgeaktionen**, der bestimmt, wann der Durchlauf abgebrochen wird. Ist der Schwellwert zu niedrig gewählt, kann das zur Folge haben, dass der Agent zu früh den Navigationsdurchlauf abbricht. Ist der Schwellwert jedoch zu hoch angestezt, kann dies dazu führen, dass der Agent ein wenig performantes Verhalten zeigt (da er ‚planlos umherirrt').

Bei den (noch austehenden) Tests muss sich zeigen, welche Wahl der oben genannten Parameter zu einem performanten Verhalten des Agenten führen.

Kapitel 6

Schlussbetrachtung und Ausblick

6.1 Schlussbetrachtung

Das in Kapitel 5 vorgestellte Aktionsmodul stellt eine Ablaufsteuerung bereit, die die primitiven Aktionen des Geometrischen Agenten während der Navigationsphase auswählt und anstößt. Um dem Problem der Unterspezifiziertheit der Routeninstruktion gerecht zu werden, werden die primitiven Aktionen in einem Prozess der lokalen Planung im Kontext der aktuellen Situation selektiert. Die Ablaufsteuerung orientiert sich bei der Auswahl der primitiven Aktionen am internen Zustand des Agenten, der das Wissen (und die Annahmen) des Agenten über die aktuelle Situation repräsentiert, sowie am Aktionsplan, der die in der natürlichsprachlichen Routeninstruktion explizit genannten Aktionen auf einer abstrakten Stufe beinhaltet. Für die Auswahl einer primitiven Aktion, die in der aktuellen Situation angemessen ist, muss der Agent sowohl den Typ der primitiven Aktion, wie auch ihr Argument bestimmen. Der Typ der primitiven Aktion orientiert sich an dem Typ der Anweisung aus dem Aktionsplan, die gerade bearbeitet wird. Für das Argument der primitiven Aktion muss im Prozess der lokalen Planung ein Referenzobjekt aus dem Perzeptionsgraphen bestimmt werden, das auf ein Objekt verweist, das der Agent in der aktuellen Szene wahrgenommen hat und koreferent zu dem Argument der aktuellen Anweisung ist. Routeninstruktionen sind typischerweise bezüglich der beschriebenen Route unterspezifiziert. Dies hat zur Folge, dass der Prozess der Koreferenzauflösung mehrere Objekte als Lösung ermittelt, was den Geometrischen Agenten vor das Problem stellt, eines dieser Objekte auszuwählen.

Diese Auswahl ist insbesondere bei der Abarbeitung von Anweisungen kritisch, die eine Bewegungsaktion beschreiben. Hier hat das Aktionsmodul die Aufgabe, einen Pfad aus der Umgebung für die folgende (der Anweisung entsprechenden) primitiven Bewegungsaktion auszuwählen. Das Abarbeitungsverfahren kombiniert zwei Konzepte, um dem Problem der Ambiguität gerecht zu werden: Eine defensive Strategie der Pfadauswahl soll gewährleisten, dass der Agent bei erstmaligem Folgen einen möglichst kurzen koreferenzierten Pfad auswählt. Stellt sich heraus, dass der ausgewählte Pfad zu kurz war, wird eine entsprechende Folgeaktion ausgeführt. Ein Pfad wird als zu kurz interpretiert, wenn der Agent in dem Entscheidungspunkt, an dem er sich nach der Bewegung auf dem Pfad befindet, das Argument der folgenden Instruktionsanweisung nicht erfolgreich in der Umgebung (d.h. im Perzeptionsgraphen) koreferenzieren kann. Durch die Folgeaktion wird entweder eine Drehung ausgeführt, was dem

Problem des beschränkten Sichtfelds des Agenten gerecht wird, oder eine weitere Bewegung auf einem Pfad angestoßen, der eine Fortsetzung des letzten Pfades darstellt. Durch dieses Verfahren wird implizit ein Pfad aus dem Instruktionsmodell sukzessive mit aufeinanderfolgenden Pfaden aus der Umgebung des Agenten identifiziert.

6.2 Ausblick

Die Integration folgender Konzepte bzw. Kompetenzen wären für den Geometrischen Agenten sinnvoll:

Lernen von Teilrouten: Soll der Agent wiederholt verschiedene Navigationsaufgaben in derselben Umgebung ausführen, ist es sinnvoll, dass der Agent Routen *lernen* kann. Hierfür muss er sich die primitiven Aktionen merken, die er für die Abarbeitung einer Routeninstruktion ausführt. War der Navigationsdurchlauf erfolgreich, kann die Liste von ausgeführten primitiven Aktionen mit der entsprechenden Route verknüpft und in einer Routenbibliothek gespeichert werden. Ist der Agent beispielsweise in einem Navigationsdurchlauf der Route vom Pförtner zu Haus C erfolgreich gefolgt, speichert er die primitiven Aktionen, die ihn zu Haus C geführt haben, in einem Gedächtnis ab. Danach ‚kennt' er die Route, und in einer ihm später mitgeteilten Routeninstruktion, die beispielweise eine Route vom Pförtner zu Haus P3 beschreibt, wäre eine erste Anweisung wie „Gehe zuerst vom Pförtner zu Haus E." hinreichend spezifisch, um erfolgreich vom Agenten abgearbeitet werden zu können.[1]

Planüberwachung: Planüberwachung kann die Performanz des Geometrischen Agenten zusätzlich erhöhen, da es Fälle von ‚glücklichem Zufall' (*serendipity*) erkennen kann (vgl. Abschnitt 4.4.1). Dieser könnte beispielsweise auftreten, wenn der Agent einer anderen Teilroute als der vorgeschlagenen gefolgt ist und dadurch eine Region erreicht hat, die auf der Route der Routeninstruktion erst zu einem späteren Zeitpunkt erreicht werden sollte. Wenn er dann in dieser Situation eine Landmarke sehen kann, zu der er sich bewegen soll, kann er den Teilplan, der ihn zu dieser Landmarke führt, ignorieren und sich zu der Landmarke hinbewegen. Um einen Fall von *serendipity* zu erkennen, würde der Geometrische Agent alle Argumente noch folgender Instruktionsanweisungen bei jeder Zustandsaktualisierung daraufhin prüfen, ob sie erfolgreich koreferenziert werden können. Wird ein Objekt o_s früher als erwartet perzipiert und efolgreich koreferenziert, kann ein Teil des Aktionsplans übersprungen werden (und durch die Anweisung $!\mathsf{BE\_AT}(\mathsf{o}_s)$ ersetzt werden).

[1] Dies entstspricht dem Vermögen des IBL-Systems (siehe Abschnitt 4.5.2), Routen zu erlernen.

Literaturverzeichnis

[Agre und Chapman 1990] Agre, P. E. und D. Chapman (1990). *What are plans for?*. Robotics and Autonomous Systems, 6:17–34.

[Aylett et al. 2000] Aylett, R. S., A. M. Coddington und G. J. Petley (2000). *Agent-based Continuous Planning*. In: *Proceedings of the 19-th Workshop of the UK Planning and Scheduling Special Interest Group (PLANSIG 2000)*.

[Blum und First 1997] Blum, A. und M. First (1997). *Fast Planning through planning graph analysis*. Artificial Intelligence, 90(1-2):281–300.

[Bugmann et al. 2001] Bugmann, G., S. Lauria, T. Kyriacou, E. Klein, J. Bos und K. Coventry (2001). *Using Verbal Instruction for Route Learning: Instruction Analysis*. In: *Proc. TIMR 01 - Towards Intelligent Mobile Robots*, Department of Computer Science, Manchester University, ISSN 1361 - 6161. Report number UMC-01-4-1.

[Denis 1997] Denis, M. (1997). *The description of routes: A cognitive approach to the production of spatial discourse*. Cahiers de Psychologie Cognitive, 16:409–458.

[Dennett 1987] Dennett, D. C. (1987). *The Intentional Stance*. MIT Press, Cambridge, Mass.

[Eschenbach 1988] Eschenbach, C. (1988). *SRL im Rahmen eines textverarbeitenden Systems*. Technischer Bericht, GAP-AP 3, Hamburg.

[Eschenbach et al. 2000] Eschenbach, C., L. Tschander, C. Habel und L. Kulik (2000). *Lexical Specifications of paths*. In: Freksa, C., W. Bauer, C. Habel und K. F. Wender, Hrsg.: *Spatial Cognition II*, S. 127–144. Springer, Berlin.

[Ferber 2001] Ferber, J. (2001). *Multiagentensysteme*. Addison Wesley, München.

[Fikes und Nilsson 1971] Fikes, R. E. und N. J. Nilsson (1971). *STRIPS: A new approach to the application of theorem proving to problem solving*. Artificial Intelligence, 2(3-4):189–208.

[Firby 1994] Firby, J. (1994). *Task Networks for Controlling Continuous Processes*. In: *Proceedings of the Second International Conference on AI Planning Systems*, Chicago.

[Firby 1995] Firby, J. (1995). *The RAP Language Manual*. Technischer Bericht, University of Chicago. Animate Agent Project Working Note AAP-6.

[Habel 1986] Habel, Ch. (1986). *Prinzipien der Referentialität*. Springer, Berlin.

[Hayes-Roth et al. 1993] Hayes-Roth, B., K. Pfleger, P. Morignot und P. Lalanda (1993). *Plans and Behaviour in Intelligent Agents*. Technischer Bericht KSL-95-35, Knowledge Systems Laboratory, Stanford.

[Helwich 2003] Helwich, J. (2003). *Graphenbasierte Navigation eines Geometrischen Agenten: Integration von Perzeption und Instruktion*. Diplomarbeit, Universität Hamburg, Fachbereich Informatik.

[Kowalski und Sergot 1986] Kowalski, R. und M. Sergot (1986). *A logic-based calculus of events*. New Generation Computing, 4(1):67–95.

[Kyriacou et al. 2002] Kyriacou, T., G. Bugmann und S. Lauria (2002). *Vision-Based Urban Navigation Procedures for Verbally Instructed Robots*. In: *Proceedings of the 2002 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems (IROS'02)*, S. 1326–1331, EPFL, Lausanne, Schweiz.

[Lauria et al. 2002a] Lauria, S., G. Bugmann, T. Kyriacou und E. Klein (2002a). *Mobile Robot Programming Using Natural Language*. Robotics and Autonomous Systems, 38(3-4):171–181.

[Lauria et al. 2002b] Lauria, S., T. Kyriacou, G. Bugmann, J. Bos und E. Klein (2002b). *Converting Natural Language Route Instructions into Robot-Executable Procedures*. In: *Proceedings of the 2002 IEEE Int. Workshop on Robot and Human Interactive Communication (Roman'02)*, S. 223–228, Berlin.

[Lazanas 1995] Lazanas, A. (1995). *Reasoning About Uncertainty in Robot Motion Planning*. Doktorarbeit, Stanford University.

[Liu und Daneshmend 2004] Liu, J. und L. K. Daneshmend (2004). *Spatial reasoning and planning: geometry, mechanism, and motion*. Springer, Berlin.

[McCarthy und Hayes 1969] McCarthy, J. und P. J. Hayes (1969). *Some philosophical problems from the standpoint of artificial intelligence*. Machine Intelligence, 4:463–502.

[Müller et al. 2000] Müller, R., T. Röfer, A. Lankenau, A. Musto, K. Stein und A. Eisenkolb (2000). *Coarse Qualitative Descriptions in Robot Navigation*. In: Freksa, C., W. Bauer, C. Habel und K. F. Wender, Hrsg.: *Spatial Cognition II*, S. 265–276. Springer, Berlin.

[Nehmzow 2002] Nehmzow, U. (2002). *Mobile Robotik*. Springer, Berlin.

[Pednault 1986] Pednault, E. P. D. (1986). *Formulating multiagent, dynamic-world problems in the classical planning framework*. In: Georgeff, M. P. und A. L. Landsky, Hrsg.: *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop*, S. 47–82. Morgan Kaufman, Timberline, Oregon.

[Poole et al. 1998] Poole, D., A. Mackworth und R. Goebel (1998). *Computational Intelligence*. Oxford University Press, New York.

[Röfer und Lankenau 2002] Röfer, T. und A. Lankenau (2002). *Route-Based Robot Navigation*. In: Freksa, C., Hrsg.: *Künstliche Intelligenz - Themenheft Spatial Cognition*, S. 29–31. Fachbereich 1 der Gesellschaft für Informatik e.v., arenDTaP.

[Russell und Norvig 2003] Russell, S. und P. Norvig (2003). *Artificial Intelligence. A Modern Approach*. Prentice Hall, Upper Saddle River, N.J., USA.

[Sacerdoti 1975] Sacerdoti, E. D. (1975). *The nonlinear nature of plans*. In: *Proceedings of the Fourth International Joint Conference on Artificial Intelligence (IJCAI-75)*, S. 206–214, Tbilisi, Georgia.

[Searle 1980] Searle, J. R. (1980). *Minds, Brains, and Programs*. Behavioral and Brain Sciences, 3:417–424.

[Searle 1983] Searle, J. R. (1983). *Intentionality: An Essay in the Philosophy of Mind*. Cambridge University Press, Cambridge.

[Slaney und Thiébaux 2001] Slaney, J. und S. Thiébaux (2001). *Blocks World revisited*. Artificial Intelligence, 125:119–153.

[Soderland und Weld 1991] Soderland, S. und D. S. Weld (1991). *Evaluating nonlinear planning*. Technischer Bericht TR-91-02-03, University of Washington, Department of Computer Science and Engeneering, Seattle, Washington.

[Suchman 1987] Suchman, L. (1987). *Plans and Situated Action: the problem of human-machine communication*. Cambridge University Press.

[Tate 1977] Tate, A. (1977). *Generating project networks*. In: *Proceedings of the Fifth International Joint Conference on Artificial Intelligence (IJCAI-77)*, S. 410–416, Cambridge, Massachusetts.

[Trullier et al. 1997] Trullier, O., S. I. Wiener, A. Berthoz und J.-A. Meyer (1997). *Biologically Based Artificial Navigation Systems: Review and Prospects*. In: *Progress in Neurobiology*, Bd. 51, S. 483–544.

[Tschander et al. 2003] Tschander, L., H. R. Schmidtke, C. Eschenbach, C. Habel und L. Kulik (2003). *A Geometric Agent Following Route Instructions*. In: Freksa, C., W. Bauer, C. Habel und K. F. Wender, Hrsg.: *Spatial Cognition III*, S. 89–111. Springer, Berlin.

[Turing 1950] Turing, A. (1950). *Computing machinery and intelligence*. Mind, 59:433–460.

[Waterman 1989] Waterman, T. H. (1989). *Animal Navigation*. Scientific American Library, New York.

[Wehner und Räber 1979] Wehner, R. und F. Räber (1979). *Visual spatial memory in desert ants cataglyphis bicolor*. In: *Experientia*, Bd. 35, S. 1569–1571.

[Weld 1998] Weld, D. S. (1998). *Recent Advances in AI Planning*. Technischer Bericht, Department of Computer Science & Engineering, University of Washington. UW-CSE-98-10-01.

[Werner et al. 2000] Werner, S., B. Krieg-Brückner und T. Herrmann (2000). *Modelling navigational knowledge by route graphs*. In: Freksa, C., W. Bauer, C. Habel und K. F. Wender, Hrsg.: *Spatial Cognition II*, S. 295–316. Springer, Berlin.

[Wooldridge 1999] Wooldridge, M. (1999). *Intelligen Agents*. In: Weiss, G., Hrsg.: *Multiagent Systems*, S. 1–51. MIT Press.

[Wooldridge 2000] Wooldridge, M. (2000). *Reasoning about Rational Agents*. MIT Press, Cambridge, Massachusetts.

[Wunderlich und Reinelt 1982] Wunderlich, D. und R. Reinelt (1982). *How to get there from here*. In: Jarvella, R. J. und W. Klein, Hrsg.: *Speech, Place, and Action*, S. 183–201. Wiley, Chichester.

[Yang 1997] Yang, Q. (1997). *Intelligent Planning*. Springer, Heidelberg.

[Zhang 1995] Zhang, J. (1995). *Ein integriertes Verfahren zur effizienten Planung und Ausführung von Roboterbewegungen in unscharfen Umgebungen*. Doktorarbeit, Universität Karlsruhe.

Erklärung

Ich versichere, dass ich die vorstehende Arbeit selbstständig und ohne fremde Hilfe angefertigt und mich anderer als der im beigefügten Verzeichnis angegebenen Hilfsmittel nicht bedient habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht.

Hamburg, den 9. Mai 2005

Nils Bittkowski