Diplomarbeit

Goal Deliberation

von

Tobias Blasche
Universität Hamburg – Fachbereich Informatik
Arbeitsbereich: Verteilte Systeme und Informationssysteme

Betreuer

Prof. Dr. Winfried Lamersdorf und
Dr. Heiko Rölke

22. Juli 2005

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die Zeiten, in denen Computersysteme hauptsächlich isoliert von anderen Systemen benutzt wurde, gehören der Vergangenheit an. Heutzutage und im Laufe der letzten Jahre, ist eine zunehmende Vernetzung dieser vormals isolierten Systeme eingetreten. Dadurch war zu beobachten, dass eine verstärkte Orientierung auf den Verteilungsaskpekt dieser Systeme stattgefunden hat. Dies führte schlussendlich z.B. zu dezentralisierten, großen, offenen und heterogenen Systemen. Ein bekanntes Beispiel für solch ein System stellt das Internet dar.

Durch diese Entwicklungen steigerte sich der Grad der Komplexität. Dies betraf das zugrundeliegende System und im gleichen Atemzug die auf diesem System aufsetzenden Anwendungen. Es wurde argumentiert [Wei2000], dass einer solchen gesteigerten Komplexität nicht mehr durch einen zentralen Modellierungs- und Programmieransatz begegnet werden konnte, dass ein solcher Ansatz sogar kontraintuitiv teilweise auch unmöglich sei. Vielmehr favorisierte man eine Verteilung des Lösungsansatz auf viele kleine spezielle Einheiten. Diese speziellen Einheiten, auch Agenten genannt, sollten der Komplexität entgegenwirken und dem Verteilungsaspekt entsprechen.

Der Begriff Agent[1] hat seinen Ursprung in der künstlichen Intelligenz. Er beschreibt ein autonomes System welches durch Sensoren und Effektoren seine Umwelt wahrnehmen bzw. beeinflussen kann. Häufig wird einem Agenten das Attribut intelligent zugesprochen. Dies soll verdeutlichen, dass ein Agent nicht auf irgendeine Art und Weise handelt, sondern in einem übergeordneten Sinn rational. Das größte Potential entwickeln Agenten in einem Verbund. Solch einen Verbund bezeichnet man allgemein als ein Multiagentensystem. Multiagentensysteme werden als die Schlüsseltechnologie der Zukunft gesehen, die, ähnlich wie die Objektorientierung, die Softwaretechnik grundlegend verändern soll.

Agentenorientierung vertritt ein neuartiges Programmierparadigma, welchem die folgende Annahme zugrundeliegt. Eine Abstraktion eines Softwareentwicklungsprozess bürgt das meiste Potential, wenn der semantische Abstand zwischen den intuitiv verwendeten Konstrukten einer Analyse des Problems und den

[1] Eine Ausführliche Behandlung des Themas Agent ist in Kapitel 2.1 nachzulesen

in einer softwaretechnischen Modellierung dargestellten, minimiert wird [Jen99]. Die Abstraktion die in der Agentenorientierung verwendet wird äussert sich in der Verwendung von mentalistischen Notationen wie „Belief“, „Desire“ und „Intention“ kurz BDI (siehe Kapitel 2.1.2). Der Grund für die Verwendung dieser Abstraktion ist die Möglichkeit damit das Verhalten von komplexen Systemen in einer einfacheren Art und Weise zu beschreiben.

Das BDI-Modell [Bra87] stellt ein vielfach benutztes Modell für intelligente Agenten in dynamischen Umgebungen dar und wurde in zahlreichen Agentensystem (PRS, JACK, JAM, Jadex) implementiert. Es ist theoretisch durch die Standardliteratur [RG91, CL93] fundiert und kann somit als theoretisch abgesichert gelten.

1.1 Motivation

Schon vergleichsweise früh wurde in Standardwerken zur Theorie der BDI-Agenten [RG91, CL93], die Anforderung der Konsistenz an das Konzept Ziel gestellt. Ziele eines Agenten sollten demnach, keine sich gegenseitig ausschließenden Zustände beschreiben, wenn ein Agent diese bearbeiten soll.

Jedoch war das, was man als Ziel bezeichnete, in den implementierten Systemen nur rudimentär als Ereignis (event) vorhanden. Das hatte zu dieser Zeit vor allem den Hintergrund, dass das System berechenbar bleiben sollte. So bot ein Ziel in diesem Sinn, durch seinen „flüchtigen“ Charakter, keine Möglichkeiten einen Ansatz für eine Deliberation zu verfestigen. Dieser Verlust auf der konzeptionellen Ebene, wurde bereits zu Zeiten der Veröffentlichungen von [RG91] diskutiert. So beschreibt Kirsh, wenn auch noch losgelöst von einer BDI-Architektur:

> „Without representation, desires lack the modularity to be reasoned about, or even flexibly assembled. If the representations are not conceptual they will not about enduring states of the world that can be entertained and reasoned over.“ [Kir91]

Viele Jahre mussten noch vergehen, bevor diese Thematik wieder aufgegriffen wurde. So wurde erst im Jahr 2002 in [WPHT2002] untersucht, welche Form die Repräsentation eines Ziels für welche generellen Absichten annehmen sollte. Dabei wurden deklarative und prozedurale Ansätze beschrieben. Für den deklarativen Ansatz wurde festgestellt:

> „However, by omitting the declarative aspect of goals the ability to reason about goals is lost“ [WPHT2002]

Dies stellte einen eindeutigen Hinweis dar, welche Form zukünftige Zielkonzepte haben sollten. In [BPML04] wurde dies aufgegriffen und ein solches verfeinertes Zielkonzept konstruiert. Es ist Bestandteil des Jadex [BP2005] System, welches für die Implementation dieser Diplomarbeit verwendet wird.

In den vorhergehenden Ausführungen ist deutlich geworden, dass für das Entstehen einer Motivation bezüglich der Deliberation über Zielen, das Zielkonzept, mit einer expliziten und deklarativen Repräsentation, eine unabdingbare

Voraussetzung darstellt. Die Frage die sich jetzt anschließt, ist die nach den Ursachen für das Bedürfnis über Ziele zu Deliberieren.

Die Ursachen die eine Deliberation notwendig machen, wurden ebenfalls schon frühzeitigt erkannt, konnten jedoch aufgrund mangelndem Ausdrucksvermögen des Konzeptes Ziel nicht wie in dieser Diplomarbeit angestrebt realisiert werden. So befand M. Pollack 1992:

> "Agents in real, dynamic environments need to be receptive to many potential goals, goals that do not typically arise in a neatly sequential fashion. They need to decide what to respond to, and when. Intelligent behaviour depends not just on being able to decide how to achieve one's goals, but also on being able to decide which goals to pursue in the first place, and when to abandon or suspend the pursuit of an existing goal." [Pol92]

Kirsh leitete aus der Feststellung, dass komplexe Systeme existieren können, die eine Vielzahl an Zielen besitzen, folgendes ab:

> „What this means is that when desire systems get large there must be some type of desire management, such as deliberation, weighing competing benefits and costs, and so on." [Kir91]

Das bedeutet also, dass ab einer gewissen Anzahl an parallel zu bearbeitenden Zielen eine Deliberation notwendig wird. Das liegt zum Einen an den Zielen selbst. So beschreibt Wilensky in [Wil83] eine Vielzahl an Interaktionen die zwischen Zielen auftreten können. Zu erwähnen wären hier neben positiven vor allem die negativen Interaktionen. Es kann aber auch zwischen Plänen von Zielen zu nicht gewünschten Interaktionen kommen die sich indirekt auch auf die Ziele auswirken.

Zusammengefasst kann gesagt werden, dass eine Deliberation auf Zielen notwendig wird, wenn ein Agent mehrere Ziele parallel bearbeiten muss. Denn durch die gesteigerte Anzahl an Zielen, erhöht sich auch die Wahrscheinlichkeit, dass diese Ziele negative interagieren. Da die Agenten, die in dieser Diplomarbeit betrachtet werden rational[2] sind, sollten sie diese Interaktionen berücksichtigen. Sie sollten Ziele so auswählen, dass möglichst alle erfolgreich bearbeitet werden können.

1.2 Zielsetzung

Ziel dieser Diplomarbeit soll es sein, ein Deliberationsverfahren zu entwickeln und zu implementieren. Als Plattform für dieses Verfahren wird das Jadex-System in der Version 0.93beta[3] benutzt. Die Entwicklung und Implementierung des Verfahren wird dabei durch die Analyse bestehender Verfahren unterstützt.

[2] Eigenschaften von Agenten werden in Kapitel 2.1.1 ausgiebig behandelt.

[3] Die Version 0.93beta wurde bis ca. Februar 2005 aktualisiert, enthält also schon Ergänzungen der Folgeversion.

Vorrangiges Ziel einer solchen Entwicklung soll die Möglichkeit sein, dass einem Entwickler eines Agenten erweiterte Spezifikationsmöglichkeiten bereitgestellt werden. Diese kann der Entwickler dann verwenden um, dem Agenten auf diese Weise Informationen zukommen zu lassen. Der Agent kann diese Informationen dann nutzen, um Entscheidungen zwischen Zielen zu treffen. Dabei soll auch die Bedienbarkeit der Spezifikation berücksichtigt werden.

Das Verfahren selbst sollte einen gewissen generischen Charakter aufweisen, also möglichst in einer Vielzahl von Anwendungen einsetzbar sein. Es soll kein spezielles, auf eine spezifische Anwendungsdomäne zugeschnittenes Verfahren sein und somit den Aufwand einer Spezifizierung und den allgemeinen Nutzen welchen man für Jadex aus dem Verfahren ziehen kann in Frage stellen. Es sollte aber auch, besonders im Hinblick auf das bereits vorhandene Deliberationsverfahren[4], nicht „das Rad neu erfinden“ und bestehendes in einer anderen Fassung präsentieren, sondern versuchen ergänzend zu wirken und wenn das nicht möglich ist, als eigenständiges Verfahren konkurrenzfähig zum vorhandenen Verfahren zu werden.

Rahmenbedingungen die die Implementation zu den Seiten hin abgrenzen sollen, sind folgende. Durch die Verwendung des Jadex Systems wird die „Belief“, „Desire“ und „Intentions“ Architektur [Bra87] eines Agenten bevorzugt. Das bedeutet im Besonderen, dass die Pläne die ein Agent verwendet, vorkonfiguriertes prozedurales Wissen sind und nicht zur Laufzeit verändert werden können (replanning). Außerdem soll das Deliberationsverfahren vorerst auf den einzelnen Agenten beschränkt bleiben und somit nur dessen Ziele berücksichtigen.

1.3 Gliederung

Kapitel 2 dieser Diplomarbeit befasst sich mit der Einleitung in die Thematik der intelligenten Agenten. Dort werden ausführlich die Grundlagen dieser Arbeit behandelt. Es werden eine allgemeine Beschreibung von intelligenten Agenten und deren Eigenschaften, das unverzichtbare BDI-Modell und die Beispielarchitekturen IRMA, PRS detailliert vorgestellt. Schlussendlich wird die Agentenplattform Jadex, die Grundlage der Implementation dieser Arbeit ist, betrachtet. Es werden die Konzepte erläutert die Jadex verwendet, wobei genauer auf die Erweiterungen eingegangen werden soll, die relevant für ein Goal Deliberationsverfahren sind.

Im Kapitel 3 wird dann der Fokus der Betrachtungen enger gefasst, indem sich auf das Kernelement eines Goal-Deliberationsverfahren, das Ziel, konzentriert wird. Dabei sollen die Eigenschaften und die Struktur dieses Konzeptes näher beleuchtet werden. Es soll aufgezeigt werden, welche zusätzlichen Informationen in die Struktur eines Ziels integriert werden können, um einen Entscheidungsprozess positiv zu unterstützen. Ebenso sollen die Interaktionen zwischen Zielen betrachtet werden. Hierzu zählen vor allem negative Interaktionen die erheblichen Einfluss auf das Verhalten eines Agenten haben können. Aber auch

[4]Easy Deliberation

positive Interaktionen sind unter den Interaktionen zwischen Zielen zu finden. Sie sind zwar nicht so bedeutend wie negative Interaktionen, jedoch kann sich deren Ausnutzung positiv auf eine Optimierung des Verhaltens eines Agenten auswirken.

Nach der in den Kapiteln 2 und 3 ausreichend beschriebenen Vorarbeit, soll im Anschluss in Kapitel 4 zum Schwerpunkt, der Betrachtung von Goal-Deliberationsverfahren, dieser Diplomarbeit übergegangen werden. Hier werden aktuelle Ansätze der Forschung vorgestellt und analysiert, aber auch bewertet. Eine solche Bewertung soll als Entscheidungshilfe für das Verfahren, dass später in Kapitel 5 implementiert werden soll, dienen.

In Kapitel 5 wird dann schlussendlich das Verfahren vorgestellt welches implementiert werden soll. Für das Verfahren wird zunächst beschrieben wie es konzipiert werden soll um im Anschluss daran die Details der Implementation zu erläutern. Den Schluss dieses Kapitels bilden Vergleiche mit bestehenden Verfahren. Es wird das implementierte Verfahren mit einem ähnlichen Verfahren und mit dem aktuell in Jadex implementierten Verfahren verglichen.

Kapitel 6 schließt diese Diplomarbeit. Dort wird ein Fazit dieser Arbeit gezogen und ein kurzer Ausblick auf weitere Forschungsmöglichkeiten im Bereich der Goal Deliberation präsentiert.

Kapitel 2

Grundlagen

In diesem Kapitel sollen die für das Thema dieser Arbeit wichtigen Grundlagen geliefert werden. Unabdingbar ist dafür ein Grundverständnis über das was einen Agenten ausmacht. Hierzu zählen seinen Eigenschaften (2.1.1) und möglichen Architekturen (2.1.2). Innerhalb der Architekturen wird die „Belief", „Desire" und „Intention" Architektur behandelt werden. Sie stellt das Modell der Agenten dar für die das in der Zielsetzung 1.2 beschriebene Deliberationsverfahren entwickelt werden soll.

Im Anschluss an die doch sehr philosophischen Betrachtungen der BDI Architektur soll das Gesamtverständnis, durch zwei praktische Umsetzungen abgerundet werden. Es werden die „Intelligent Resource Bounded Machine" (IRMA) und das „Procedural Reasoning System" (PRS) vorgestellt. Abschließend widmet sich dieses Kapitel dem Jadex System.

2.1 Intelligente Agenten

Unter dem Begriff eines Agenten ist eine Vielzahl von beliebig komplexen, beliebig spezifischen Systemen vorstellbar. Was einen Agenten dabei genau auszeichnet, darüber gibt es jedoch verschiedene Auffassungen. Ursächlich dafür ist die Vielfältigkeit der Forschungsgebiete die an diesem Thema beteiligt sind. So legt ein Teil z.B. Wert darauf, dass Agenten u.a. die Eigenschaft „lernfähig" besitzen. Andere hingegen sehen diese Eigenschaft als nicht notwendig an. Eine konkrete Einordnung des Begriffes, so wie er hier in dieser Diplomarbeit verwendet werden soll, ist daher nötig.

In Abbildung 2.1 ist ein Agent skizziert. Grundlegend betrachtet ist ein solcher Agent ein Computersystem, welches in eine Umgebung eingebettet ist. In dieser Umgebung führt der Agent autonome Aktionen, zwecks Erreichung seiner Ziele aus [Woo2002]. Autonomie bedeutet dabei, dass der Agent ohne direkte Einwirkung vom Menschen agieren kann und Kontrolle über seine Aktionen und seinen internen Zustand hat.

Die Aktionen, die ein Agent ausführen kann, bilden seine Fähigkeiten. Die-

se haben den Zweck Veränderungen in der den Agent umgebenden Welt hervorzurufen. Diese Veränderungen sind nichtdeterministisch, da ein Agent im Allgemeinen keine vollständige Kontrolle über seine Umwelt besitzt. So kann es passieren, dass der gewünschte Effekt einer Aktion nicht eintritt und diese Aktion damit fehlschlägt.

Aktionen sind nicht in jeder Situation anwendbar. So wird z.B. zur Ausführung einer Aktion A u.U. eine bestimmte Bedingung B benötigt. Nur wenn die Bedingung B gegeben ist, kann der Agent Aktion A ausführen. Solche Bedingungen nennt man Vorbedingungen (preconditions).

Die Aufgabe des Agenten ist zu entscheiden, welche der ihm zur Verfügung stehenden Aktionen zur Bearbeitung seiner Aufgaben angewendet werden. Erschwerend kommt hinzu, dass der Agent die Eigenschaften der Umwelt berücksichtigen muss. In [Woo2002] ist eine Klassifizierung möglicher Eigenschaften dargestellt. Von besonderem Interesse sind z.B. nichtdeterministische, dynamische und nicht vollständig zugreifbare Welten. In einer dynamischen Welt muss der Agent damit rechnen, dass sich die Welt unkontrollierbar für ihn ändern kann. Es kann also passieren das bestimmte Aktionen nicht mehr anwendbar sind bzw. fehlschlagen.

Eine Entscheidung welche Aktionen wann ausgeführt werden sollen ist also nicht trivial, sondern stellt zum Teil erhebliche Ansprüche an den Agenten selbst. Es ist eines der Kernprobleme der Agentthematik und wird unter dem Begriff der Deliberation später in diesem Kapitel aufgegriffen werden.

Der bisher verwendete Begriff des Agenten klassifizierte eine grundlegende Klasse von Agenten. Er trifft auf eine Vielzahl von Systemen zu, auch solche von denen man nicht unbedingt erwartet als Agent bezeichnet zu werden. Deshalb wird der Begriff Agent im weiteren Verlauf dieses Kapitels spezifischer formuliert. Es wird von einem Agenten, so wie er in dieser Diplomarbeit verwendet wird, erwartet, dass er intelligentes Verhalten besitzt. Intelligenz ist ein weitläufiger, schwer zu definierender Begriff. Anhand von Eigenschaften die zusätzlich zu denen bereits beschriebenen vorhanden sein sollten, wird erläutert wann ein Agent als intelligent angesehen werden sollte.

2.1.1 Eigenschaften intelligenter Agenten

Damit einem Agenten die Eigenschaft intelligent zugesprochen werden kann, sollte er folgende weiteren Eigenschaften besitzen [WJ95]:

- Reaktivität,
- Proaktivität,
- Rationalität und
- Sozialität.

Dabei stehen Reaktivität und Proaktivität in engem Verhältnis zueinander. Ein Agent zeigt proaktives Verhalten wenn er zielgerichtet arbeitet. Er wählt seine

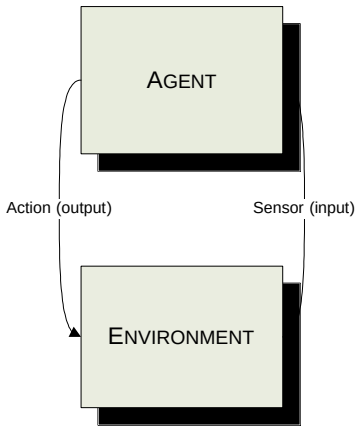

Abbildung 2.1: Ein intelligenter Agent, aus [Woo2002]

Aktionen so, dass sie ihm ermöglichen das aufertragene Ziel zu erreichen. Insbesondere in Realweltdomänen reicht diese vereinfachte Sichtweise jedoch nicht aus. Es können Veränderungen auftreten, die der Agent nicht beeinflussen kann und somit ausgewählte Aktionen nicht mehr ausführbar machen oder Ziele in Frage stellen.

Für einen intelligenten Agenten wäre ein fehlendes Erkennen solcher Situationen ein Manko. Deshalb besitzt er auch reaktives Verhalten. Durch diese Eigenschaft wird es ihm ermöglicht Ereignisse, die die Bearbeitung seiner Ziele betreffen, zu erkennen. Dadurch kann er auf gegebene Veränderungen reagieren.

Entscheidend für die Performanz eines Agenten in einer komplexen Domäne ist die Balance zwischen diesen beiden Eigenschaften. Hier sollte, durch den Entwickler des Agenten eine Maß gefunden werden, das ihm sowohl ermöglicht seinen Aufgaben nachzugehen wie aber auch auf entsprechende Veränderungen der Domäne zu reagieren.

Rationales Verhalten wiederum betrifft die Auswahl von Aktionen. Für einen intelligenten Agenten, insbesondere für die die in diese Arbeit betrachtete werden, stellt diese Eigenschaft einen zentralen Ansatz dar. Ein rationaler Agent handelt so, dass er seine Ziele erreicht und nicht so, dass er sie verfehlt.

Soziales Verhalten wird hier nur am Rande erwähnt, weil es für diese Arbeit nicht so interessant erschient. Soziale Eigenschaften sind die Grundlage für Interaktionen zwischen Agenten und somit der Konstruktion von Multiagentensystemen.

Bis hierher wurde erläutert, was ein Agent ist. Dies betraf vor allem seien Eigenschaften. Im Folgenden soll gezeigt werden wie ein solcher Agent implementiert werden kann.

Für den Entwurf und die Implementation eines intelligenten Agentensystems greifen Forscher im Bereich der künstlichen Intelligenz gerne auf die Idee des „Intentional Stance“ von Daniel Denett zurück. Grundlage des „Intentional Stance“

ist die Erklärung von menschlichem Verhalten durch Alltagspsychologie. Dabei verwendet man mentalistische Begriffe wie „Belief“, „Knowledge“, „Wish“ oder auch „Desire“ um das Verhalten von menschlichen Agenten zu erklären. Ein Beispiel soll dies verdeutlichen. Person A geht spazieren und nimmt einen Regenschirm mit. Das Verhalten dieser Person kann wie folgt erklärt werden. Person A glaubt (Belief) das es regnet und möchte (Wish, Desire) nicht nass werden.

Welche grundlegenden mentalistischen Begriffe einen Agenten beschreiben können ist in [WJ95] aufgeführt. Dort wird zwischen informationsbezogenen, mentalistischen Begriffe wie „Belief“ und „Knowledge“ und proaktiven, mentalistischen Begriffen wie „Desire“, Intention oder „Obligation“ unterschieden. Welche dieser Begriffe für einen Agenten Verwendung finden sollten wird kontrovers diskutiert. So befindet Yoav Shoham:

> „There is no unique “correct” selection of mental categories, nor a correct theory regarding them, as different applications can be expected to call for specific mental properties.“ [Sho92]

Mindestens sollte ein Agent jedoch einen Vertreter aus jedem Bereich wählen. Ein verbreiteter, populärer Ansatz ist der BDI Ansatz. Er basiert auf der Verwendung von dem informationsbezogenen Begriff Belief und dem Begriff Desire aus dem proaktiven Bereich und wird im folgedenen näher betrachtet.

2.1.2 Architekturen für intelligente Agenten

Agenten können auf verschiedenen Architekturen basieren, die ihnen ermöglichen die eine oder andere Eigenschaft besonders stark herauszubilden. So gibt es z.B. Agentensysteme die auf Architekturen basieren die rein reaktiv sind. Solche Architekturen verwenden keine explizite Repräsentation der verwendeten Konzepte und kein abstraktes Schlussfolgern. Einer der wohl bekanntesten Verfechter dieses Typs ist Rodney Brooks. Er demonstrierte mit seiner „subsumption“ Architektur [Bro86] das Agenten, in seinem Fall Roboter, einer solchen Architektur zu beeindruckenden Resultaten in der Lage waren.

Dagegen gibt es aber auch deliberative oder planende Agentensysteme die eine reaktive Eigenschaft gänzlich vermissen lassen und lediglich darauf bedacht sind ein vorgegebenes Ziel bzw. Problem durch erfolgreiches Planen zu erreichen. Solche Systeme bauen im Gegensatz zu rein reaktiven Systemen auf eine explizite, symbolische Repräsentation ihrer Anwendungsdomäne. Entscheidungen hinsichtlich welche Aktion ausgeführt werden soll, treffen sie durch Manipulation der Symbole, die die Anwendungsdomäne darstellen. Ein in die Jahre gekommener typischer Vertreter dieser Architektur ist STRIPS [FN71]. Ein bekanntes Problem dieser Architektur ist das Fehlen von Algorithmen die in annehmbarer Zeit vertretbare Resultate lieferten. An einen Einsatz von reinen deliberativen in Realweltanwendungen war daher nicht zu denken.

Agentensysteme die beide der eben besprochenen Eigenschaften entwickeln sind das Ziel einer Agentenarchitektur für Anwendungen in Realwelten. Sie sollen so zum einen auf Veränderungen der Umwelt reagieren können, aber auch

ihnen aufertragene Ziele erfolgreich bearbeiten können. Eine verbreitete Architektur die sich in diesem Bereich durchgesetzt hat ist die BDI-Architektur.

BDI-Architektur

Die BDI-Architektur, BDI für „Belief", „Desire" und „Intention", ist eine verbreitete Architektur für Agenten in dynamischen Umgebungen. Grundlage der BDI-Architektur ist das BDI-Modell des menschlichen „Practical Reasoning" von dem Philosophen Michael Bratman [Bra87]. „Practical Reasoning" ist dabei ein zweiteiliger Prozess, der sich aus Deliberation, der Entscheidung was für eine Option (Ziel, Aufgabe, Plan) verfolgt werden soll und „Means-end Reasoning", dem Planen wie diese Option erreicht werden soll, zusammensetzt. Ziel Bratmans war es, „Practical Reasoning" im Umfeld einer dynamischen Umgebung und eines ressourcen- und wissensbeschränkten Agenten zu platzieren. Zentral dafür war die Einführung von zukunftsgerichteten Intentionen (Future-directed Intentions).

Intentionen wurden bis dato als reduzierbar auf Desires, den Fernzielen eines Agenten und Beliefs, den subjektiven Wissenspeichern eines Agenten definiert. Dagegen argumentierte Bratman. Er sieht Intentionen als ein eigenständiges, primäres Konzept, die das Verhalten des Agenten kontrollieren, statt wie Desires es tun, nur zu beeinflussen.

Intentionen entstehen aus dem Prozess der Deliberation und besitzen Eigenschaften die für ein Handeln im Umfeld einer dynamischen Umgebung vorteilhaft sind. Diese Eigenschaften sollen im Folgenden näher betrachtet werden.

Eigenschaften von Intentionen:

Stabil: Intentionen besitzen die Eigenschaft der Stabilität oder Trägheit (Inertia [Bra87]). Im Verhältnis zu der Umwelt die den Agenten umgibt, bilden Intentionen einen verlässlichen und stabilen Rahmen für das Erreichen von Zielen. Hat der Agent sich entschieden eine Intention zu bilden so entscheidet er sich auch indirekt für eine Abfolge von Aktionen die auf diese Intention hinausläuft. Der Agent muss also nicht in jedem weiteren Deliberationsprozess abwägen ob er generell diese Intention bilden soll, sondern nur wie er diese erreichen will. Dies ermöglicht einen Agenten komplexe Intentionen zu bilden und vor allem in einer dynamischen Umgebung zu erreichen. Komplexe Intentionen werden durch die hierarchische Struktur die eine Intention auszeichnet unterstützt. So kann die Intention *A* zu erreichen, Subintentionen beinhalten *A1* und *A2* zu erreichen. Das ermöglicht dem Agenten im „Means-end Reasoning" Prozess Schritt für Schritt ausgehend von seinem Fokus auf eine allgemeine Intention spezifischere Intentionen zu bilden um so die allgemeine, übergeordnete Intention zu erreichen.

Beschränkend: Intentionen beschränken die Auswahl der Optionen im Deliberationsprozess. Hat der Agent eine Intention gebildet, so wird er im Allgemeinen keine Optionen, das sind Möglichkeiten zur Erfüllung anderer

„Desires“ oder Intentionen, wahrnehmen, die inkonsistent mit der Bearbeitung der betreffenden Intention sind. So beschränkt z.B. die Intention sein Auto einem Freund zu überlassen, die Intention in den Urlaub zu fahren dahingehend, dass die Option mit dem Auto in den Urlaub zu fahren nicht zu denen gehört die der Agent für eine Erfüllung der 2. Intention berücksichtigen sollte.

Beeinflussend: Intentionen beeinflussen außerdem die „Beliefs“ eines Agenten. Besitz der Agent eine Intention am Ort A zu sein, so kann er erwarten, dass er auch glaubt irgendwann einmal an diesem Ort A zu sein. Daraus folgt, dass der Agent aufbauend auf diesem „Belief“ weitere Handlungen planen kann, er also den means-end reasoning Prozess damit beeinflusst.

Im folgenden werden die Eigenschaften von Intentionen noch einmal zusammengefasst [Woo2002].

- *Intentionen fördern den „Means-end Reasoning“ Prozess:*
  Hat ein Agent eine Intention gebildet, so wird er versuchen diese zu erfüllen. Dies beinhaltet, eine vorerst noch allgemeine Intention durch sukzessives Verfeinern zu spezifischeren Intentionen umzubilden, welche somit die Mittel (Means) schaffen um das intendierte „End“ zu erreichen.

- *Intentionen beschränken die Auswahl im Deliberationsprozess:*
  Mit der Bildung einer Intention fokussiert der Agent auf ein ganz bestimmtes Ziel. Dieses Ziel im Auge, wird der Agent nur solche Optionen wahrnehmen bzw. berücksichtigen, die konsistent zu seiner Intention sind und somit zu derer Erfüllung beitragen. Intentionen bietet damit den in [Bra87] beschriebenen Filter der Zuverlässigkeit (Filter of Admissibility).

- *Intentionen sind persistent:*
  Intentionen werden gebildet um ein Ziel zu erreichen. Sie werden in der Regel nicht gleich wieder verworfen. Nur ein triftiger Grund kann sie beenden. Dabei kann es sich z.B. um die Erfüllung dieser Intention handeln.

- *Intentionen beeinflussen Beliefs, auf denen zukünftige Deliberation aufbaut:*
  Bildet der Agent eine Intention, so geht er auch davon aus, dass er diese Intention erreichen wird. Er glaubt daran und kann somit auf diesen Glauben aufbauend, weitere Absichten bilden.

Hinterfragung einer Intention: Intentionen bilden durch ihre Stabilität ein Grundlage dafür, dass der Agent auf seine Ziele fokussiert bleibt. Wie bereits im Kapitel 2.1 über intelligente Agenten erläutert wurde, ist aber ein ausgewogenes Mittelmaß zwischen diesem Fokussiert sein und einem Reagieren auf besondere Umweltveränderungen notwendig. Das bedeutet, dass Situationen eintreten können in denen die weitere Bearbeitung einer Intention hinterfragt werden muss. Diese Hinterfragung ist jedoch aufwendig und es stellt sich somit die Frage wie oft solch eine Hinterfragung sinnvoll ist. Die Beantwortung ist nicht trivial. Das

Dilemma in dem man sich dabei befindet lautet folgendermaßen: Ein Agent der seine Intentionen nicht genügend oft hinterfragt, läuft Gefahr auch dann noch seinen Intentionen nachzugehen, wenn diese bereits unerfüllbar geworden sind. Andererseits, wenn ein Agent seine Intentionen ständig hinterfragt, investiert er mehr Zeit für die Bearbeitung jener und läuft dabei wiederum Gefahr sie möglicherweise nie zu erreichen. Das Fazit lautet [Woo2002]. Für Agenten die ihre Intentionen öfter hinterfragen ist eine Umwelt die eine hohe Änderungsrate besitzt adäquater. Für Agenten hingegen die nicht sooft ihre Intentionen hinterfragen ist eine Umwelt mit einer niedrigeren Änderungsrate besser geeignet.

Das BDI Modell von Bratman hat gezeigt, wie ein philosophische Modell, praktisches Handeln erkären kann. Es hat anhand der verwendeten Begriffe, insbesondere dem der Intention, deutlich gemacht wie sich zielgerichtetes Arbeiten mit einer dynamischen Umwelt vereinbaren lässt. Die Frage die sich daran anschliesst ist nach einer geeigneten Implementierung dieses Modell. Diese Implementierung soll zeigen inwieweit das Modell in technischer „Hardware“ umsetzbar ist.

2.2 Beispielarchitekturen

2.2.1 Intelligent Resource Bounded Machine (IRMA)

Bisher wurde ein abstraktes, philosophisches Modell zur Konstruktion von intelligenten Agenten behandelt. Es wurde vor allem über den Einfluss der Intentionen in diesem Modell gesprochen. Jetzt soll diese abstrakte philosophische Sicht durch eine abstrakte Sicht auf eine Softwarearchitektur ersetzt werden. In dieser Softwarearchitektur sollen die Abläufe während des „Practical Reasoning“ konkretisiert werden. Die abstrakte Softwarearchitektur die hier behandelt wird, ist die Intelligent Resource-Bounded Machine (IRMA) [BIP88, Pol92] die in Abbildung 2.1 dargestellt ist. Sie ist eine Umsetzung des BDI-Modells von Bratman [Bra87] und stellt die Prozesse und Datenstrukturen dar die ein Agent auf Basis einer BDI-Architektur implementieren sollte.

Die IRMA basiert auf der Verwendung von partiellen Plänen. Partielle Pläne sind unvollständig ausformulierte Pläne für ein bestimmtes Ziel. Diese Pläne müssen zu gegebener Zeit vervollständigt werden. Dies geschieht durch die Bearbeitung von Subplänen.

Die in Abbildung 2.2 dargestellte IRMA besteht aus drei Prozessen und einer Datenstruktur.

Means-End Reasoner Dieser Prozess ist für die Verfeinerung der partiellen Pläne verantwortlich. Er generiert für einen partiellen Plan mögliche Subpläne die zur weiteren Bearbeitung ausgeführt werden müssen. Diese Subpläne stellen die möglichen Optionen (options) dar.

Deliberation Process Die Deliberation entscheidet welche Optionen (Pläne) aus der Menge der vorgeschlagenen Optionen (surviving options) zur Be-

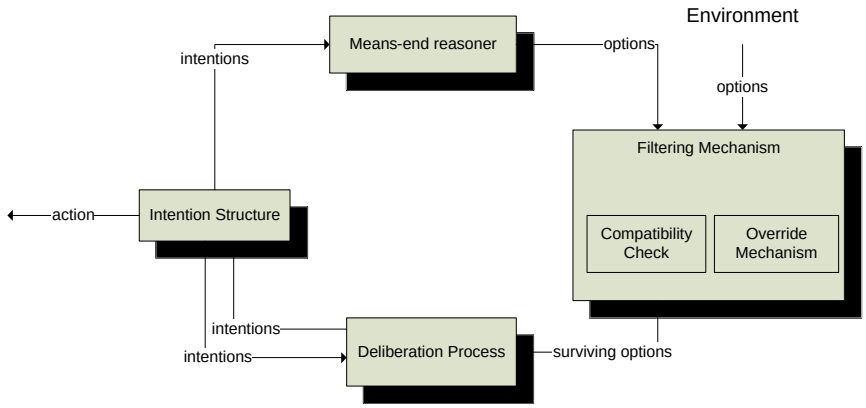

Abbildung 2.2: Abstrakte BDI-Architektur IRMA aus [Pol92]

arbeitung ausgewählt werden. Ausgewählte Optionen bilden Intentionen und werden in der Intentionenstruktur abgelegt.

Filtering Mechanism Um den Umfang der Berechnung des „Practical Reasoning“ zu reduzieren, wurde dieser Prozess in die Architektur der IRMA integriert. Er besteht aus zwei Teilprozessen. Der „Compatibility Check“ prüft ob die Optionen die durch den „Means-End Reasoning“ Prozess generiert wurden oder Optionen die sich durch Veränderungen der Domäne ergaben, kompatibel mit den aktuell zu bearbeitenden Plänen sind. Kompatible Optionen werden der Deliberation zugeführt. Durch diesen Teilprozess wird der „Filter of Admissibility“ (vgl. Eigenschaft „Beschränked“ Kapitel 2.1.2) realisiert.

Der „Override Mechanism“ repräsentiert die Sensitivität des Systems gegenüber bestimmten Zuständen der Domäne. Diese Zustände stellen Situationen dar, in denen der Agent bestehende Intentionen in Anbetracht neuer möglicher Optionen einer erneuten Prüfung (reconsideration) unterzieht. Im Speziellen bedeutet das folgendes: Eine dem „Filtering Mechnanism“ zugeführte Option ist inkompatibel mit bestehenden Intentionen. Löst diese inkompatibele Option einen „Override“ aus, so wird diese und die die Inkompatibilität verursachenden Intentionen dem Deliberationsprozess zugeführt. Der Agent entscheidet so zwischen einer Weiterführung oder einer Aufgabe dieser bestehenden Intentionen (vgl. Hinterfragung einer Intention, Kapitel 2.1.2).

Zu beachten ist, das die Zustände der Domäne die die Situationen beschreiben mit besonderer Aufmerksamkeit spezifiziert werden sollten. Waltet man hier nicht mit besonderer Sorgfalt, so kann es leicht dazu kommen, dass der Agent überempfindlich gegenüber inkompatiblen Optionen wird. Damit besteht eine erhöhte Wahrscheinlichkeit, dass der Agent häufig In-

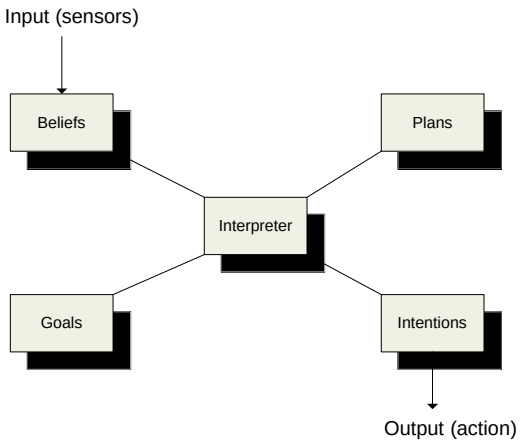

Abbildung 2.3: Systemstruktur des Procedural Reasoning System, aus [Woo2002]

tentionen verwirft um anderen Optionen nachzugehen.

Intention Structure Hier werden die zu bearbeitenden Pläne gespeichert. Besitz ein Plan nur atomare Aktionen, so können diese ohne Wirkung des „Means-End Reasoners“ ausgeführt werden. Für partielle Pläne gilt, dass sie bevor Aktionen ausgeführt werden können durch den „Means-End Reasoner“ verfeinert werden müssen.

Die „Intelligent Resource Bounded Machinery“ wurde anhand eines experimentellen Szenarios (Tileworld) erprobt. Dort versuchte man im Besonderen verschiedene Feinabstimmungen des „Filtering Mechanism“ zu erforschen und erkannte dass verschiedene Strategie beim Filtern zu verschiedenen Typen von Agenten führen.

2.2.2 Procedural Reasoning System (PRS)

Das Procedural Reasoning Systems ist wohl das bekannteste Agentensystem welches als eines der ersten die BDI Architektur in ein lauffähiges System umgesetzt hat. Es ist in den Jahren seines Bestehens mehrfach verbessert wurden was u.a. zu seinen Nachfolgern dMars und JAM [Hub99] führte. Ebenso wurde es in zahlreichen Fällen, wie dem Luftverkehrsraumüberwachungs-System OASIS und einem Reaction Control System (RCS) bei Raumfähren der NASA, praktisch eingesetzt.

In Abbildung 2.3 ist die Systemstruktur des PRS dargestellt. Es besteht aus Beliefs, Zielen (Goals), Plänen (Plans), auch bezeichnet als „Knowledge Areas“ (KA), Intentionen (Intentions) und einem Interpreter.

Beliefs: Beliefs beschreiben Fakten der Anwendungsdomäne. Sie können statisch vom Entwickler vordefiniert werden oder dynamisch zur Laufzeit durch die Bearbeitung von KAs entstehen. Durch die Spezifizierung von Ausdrücken auf der Metaebene (metalevel) können Beliefs Agenteninterna wie Ziele, Pläne oder auch Beliefs selbst referenzieren. Solche Beliefs werden als Meta-Beliefs bezeichnet.

Ziele: Ziele des PRS beschreiben Aufgaben oder gewünschtes Systemverhalten. Das PRS bietet, durch die Verwendung von temporalen Operatoren, die Möglichkeit der Spezifikation der Zieltypen „Achieve", „Maintain" und „Test". Ein Ziel vom Typ „Achieve"[1] kann durch den Ausdruck !g, g für eine Ausdruck der Aufgabe oder des Verhaltens, erzeugt werden. Wie bei Beliefs bereits beschrieben, kann auch bei der Spezifizierung von Zielen ein Ausdruck der Metaebene verwendet werden. Man bezeichnet diese Ziele dann entsprechend als Meta-Ziele.

Pläne: Pläne im PRS implementieren das prozedurale Wissen zur Bearbeitung von Zielen und Situationen. Sie bestehen aus einem Kopfteil (invocation condition) und einem Rumpfteil[2] (body). Der Kopfteil ist weiterhin unterteilt in einen Bereich der den Auslöser (triggering part) spezifiziert und in einen Bereich der den Kontext (context part) einer Bearbeitung nennt. Für den erfolgreichen Aufruf eines Plans müssen Auslöser und Kontext erfüllt sein.

Der Rumpfteil beschreibt die Schritte die ausgeführt werden wenn der Plan aufgerufen werden kann. Er wird als ein Graph mit einem ausgezeichneten Startknoten und mehreren möglichen Endknoten repräsentiert. Die Kanten zwischen den Knoten stellen Subziele dar. Für eine erfolgreiche Traversierung einer mit einem Subzielausdruck beschriebener Kanten muss zuerst das Subziel bearbeitet werden. Dies kann zur Bearbeitung weiterer Pläne führen. Eine erfolgreiche Ausführung des Planes besteht in dem Erreichen aller Subziele auf einem Pfad vom Startknoten zu einem Endknoten. Auch in Plänen bietet sich die Verwendung von Ausdrücken der Metaebene an. Pläne die diese Ausdrücke verwenden werden dann entsprechend Meta Pläne genannt. Solche Pläne werden im „Procedural Reasoning System" häufig zur Auswahl aus ausführbaren Plänen verwendet und implementieren damit ein sogenanntes „Meta-level Reasoning"

Intentionen: Die Intentionen des PRS bilden zusammen eine Intentionenstruktur. Diese Struktur definiert eine partielle Ordnung auf den Intentionen. Die die früher in der Ordnung liegen als andere, müssen auch früher realisiert werden. Das PRS ermöglicht somit eine priorisierte Ausführung von Intentionen. Die Intention an sich besteht aus Plänen. Diese Pläne sind in einem Stack zusammengefasst. Initial besteht eine Intention also aus einem Stack mit einem initialen Plan , der im weiteren Verlauf des „Means-end Reasoning", durch weitere Subpläne ergänzt werden kann.

[1] Genaue Ausführungen zu Zielen und Zieltypen werden in Kapitel 3 behandelt.
[2] KAs ohne einen body sind elementar ausführbare Aktionen

```
options := options-generator(event-queue);
selected-options := deliberate(options);
update-intentions(selected-options);
execute();
get-new-external-events();
drop-successful-attitudes();
drop-impossible-attitudes();
```

Abbildung 2.4: PRS - Interpreter aus [SRG99]

Interpreter: Das Herzstück des Procedural Reasoning System ist sein Interpreter. Dieser arbeitet nach einer Vorlage aus [SRG99], die in Abbildung 2.4 dargestellt ist und auf die im folgenden näher eingegangen werden soll.

Interpreterzyklus

Die Erzeugung von Zielen oder die Veränderung eines Beliefs erzeugen Ereignisse. Diese Ereignisse werden in einer entsprechenden Warteschlange (event-queue) abgelegt. Für diese Ereignisse generiert der Interpreter anwendbare Pläne. Dabei geht der Interpreter folgendermaßen vor [SRG99]. Er vergleicht in einem ersten Durchlauf den Auslöser eines jeden vorhandenen Plans mit den Ereignissen in der Warteschlange. Für jede Übereinstimmung wird in einem zweiten Durchlauf der Kontext dieser Pläne überprüft. Pläne, für die auch der Kontext positiv evaluiert werden konnte, stellen schlussendlich die generierten Optionen (options) dar. In der Funktion `deliberate` wird der Plan ausgewählt der als nächstes bearbeitet werden soll. Dies geschieht entweder nach dem Zufallsprinzip oder nach einem in einem Meta-Plan definierten Verfahren. Der ausgewählte Plan (selected-option) stellt eine Intention dar. In der Funktion `update-intentions` wird die Intentionenstruktur aktualisiert. Bei Vorliegen einer Intention zu einem neuen Ziel, wird ein vollständig neuer Stack erzeugt und diese Intention als initiale Intention auf diesen Stack geschoben. Ist sie jedoch aus einem Subziel entstanden so wird diese Intention auf denjenigen Stack geschoben, der für die Erzeugung des Subziels verantwortlich war.

In der Funktion `execute` wird ein Schritt einer Intention ausgeführt. Dies kann zum Einen in die Ausführung einer Aktion münden oder in die Generierung von neuen Zielen oder Beliefs welche wiederum für neue Ereignisse in der Event Queue sorgen. Abschließend werden erfolgreich bearbeitete und unbearbeitbare Intentionen aus der Menge der Intentionen, der Intentionenstruktur, entfernt und der Zyklus beginnt von vorn.

Das PRS erntete mit seiner Interpretation des BDI Modells Erfolge. Es erwies sich in ersten Experimenten als fähig, in Echtzeitumgebungen eingesetzt zu werden. Erheblichen Anteil an diesen Erfolgen hatten die durch Pläne implementierte partielle Planung, die Reaktivität und die Fähigkeiten auf der Metaebene zu agieren.

2.3 Jadex

Das BDI-Agentensystem Jadex ist Grundlage der Implementierung dieser Diplomarbeit. Im Folgenden wird daher eine ausführliche Beschreibung der Architektur von Jadex und der verwendeten Konzepte gegeben, die für das Verständnis der in Kapitel 5 vorgestellten Implementation von Wichtigkeit sind.

Jadex ist eine BDI Erweiterung der Agentenplattform JADE [Til2005] und wurde mit der Motivation entwickelt, Middlewareaspekte wie sie in JADE behandelt werden mit kognitiven Aspekten des BDI Ansatzes zu verbinden. Im Zuge dessen wurde die Weiterentwicklung der BDI Architektur durch die Entwickler Lars Braubach und Alexander Pokahr in entscheidenden Punkten vorangetrieben. So besitzt Jadex im Unterschied zu anderen BDI Systemen ein explizites und deklaratives Modell eines Ziels. Dieses erweiterte Zielkonzept bot zusammen mit dem variablen Interpreterzyklus die Voraussetzung für die Implementation eines Goal Deliberationsverfahren. Desweiteren wurde bei der Entwicklung von Jadex besonderer Wert auf die Verwendung bewährter Techniken aus der Objektorientierten Softwareentwicklung gelegt. Dies äussert sich durch die Integration von XML und der Programmiersprache Java in den Entwicklungsprozess eines Agenten.

Dieses Kapitel ist folgendermassen gegliedert. Im Anschluss wird der Aufbau eines Jadex Agenten näher betrachtet. Es wird gezeigt, aus welchen Bestandteilen der Jadex Agent aufgebaut ist und wie diese spezifiziert werden. Danach werden die Konzepte besprochen die Jadex implementiert. Hierbei wird auf die Konzepte aus dem BDI Modell eingegangen. Weiterhin wird beschrieben, welche grundlegenden Abläufe innerhalb eines Agenten zur Laufzeit stattfinden. Abschließend zeigt dieses Kapitel den Aufbau des variablen Interpreterzykluses.

Aufbau eines Jadex Agenten

Abbildung 2.5 zeigt, dass ein Jadex Agent aus den Bestandteilen „Agent Definition File“ (ADF) und einer Menge von Plänen besteht. Dabei spezifiziert der Entwickler eines Jadex Agenten, durch Angaben zu u.a. Beliefs, Zielen und Plänen in der ADF die statische Struktur des Agenten, während durch die Implementation der Pläne das dynamische Verhalten des Agenten festgelegt wird. Die Syntax der ADF folgt einer XML Sprache die durch XML Schema definiert wird. Diese Sprache setzt das BDI Metamodell von Jadex um. Ergänzt wird die XML Sprache durch eine sogenannte „Embedded Expression Language“, welche zur Spezifizierung von Ausdrücken verwendet wird die in einer XML basierenden Sprache nur schwer darstellbar wären. Bei diesen Ausdrücken kann es sich z.B. um den Wert (Fact) eines Beliefs handeln. Die Syntax dieser Erweiterungssprache geht dabei konform mit der Syntax der Programmiersprache Java und wird ergänzt durch eine Untermenge von OQL, der „Object Query Language“.

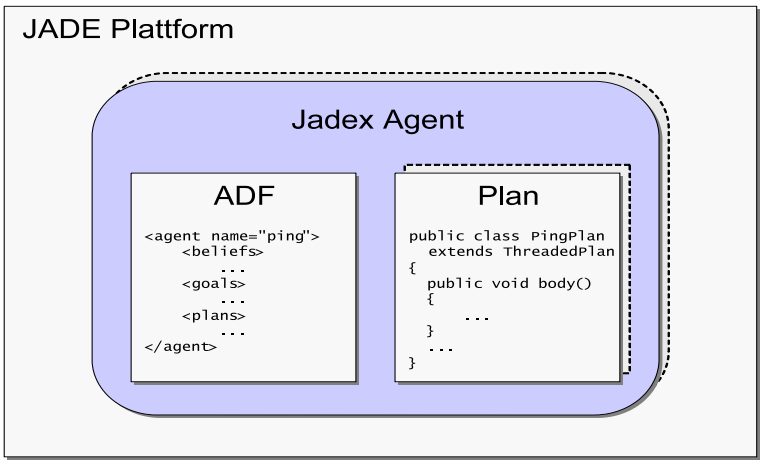

Abbildung 2.5: Jadex Agent aus [PBL05b]

Jadex-Konzepte

Die in der ADF verwendete XML Sprache ermöglicht dem Entwickler die Konzepte des BDI-Modells umzusetzen. Welche Zielsetzungen Jadex bei den einzelnen Konzepten verfolgt, soll hier beschrieben werden.

Beliefs: Das Konzept Belief wird in Jadex im Gegensatz zu anderen Agentensystemen, objektorientiert repräsentiert. Beliefs können einzelnes oder aber auch Mengen von Faktenwissen unter den Begriffen `belief` bzw. `beliefset` aufnehmen. Beliefs werden in einer Beliefbase gespeichert. Die Hauptaufgabe der Beliefbase besteht in der Verwaltung der einzelnen Beliefs. Zusätzlich dazu überwacht die Beliefbase Bedingungen die auf Beliefs spezifiziert wurden. Solche Bedingungen können relevant für die Ausführung von Plänen oder Zielen sein und beschreiben somit besondere zu berücksichtigende Situationen. Näheres dazu wird im Kapitel 3 erläutert.

Goals: Ziele (Goals) stellen in Jadex ein zentrales Konzept dar. Sie werden ebenso wie Beliefs in einer entsprechenden Base, der Goalbase verwaltet. Im Gegensatz zu anderen gängigen BDI Systemen werden Ziele explizit als eigenständiges Objekt implementiert. Sie unterscheiden sich damit von der üblichen in gängigen Systemen verwendeten Repräsentation durch Ereignisse. Dadurch treten Ziele in Jadex nicht implizit als Ursache eines ausgeführten Plans auf, sondern besitzen den Status eines eigenständigen Objekts mit spezifischen Zustand. Jadex besitzt somit auch die Möglichkeit Ziele unabhängig zu Plänen zu verwalten und die Möglichkeit der Artikulation einer Attitüde gegenüber Zielen. Unter der Vielzahl der vorhanden Zieltypen bietet Jadex die Spezifizierung der vier Zieltypen „Achieve“, „Maintain“, „Perform“ und „Query“. Was genau diese Zieltypen bedeuten wird ebenfalls im Kapitel 3, dem Kapitel über das Konzept Ziel, erläutert.

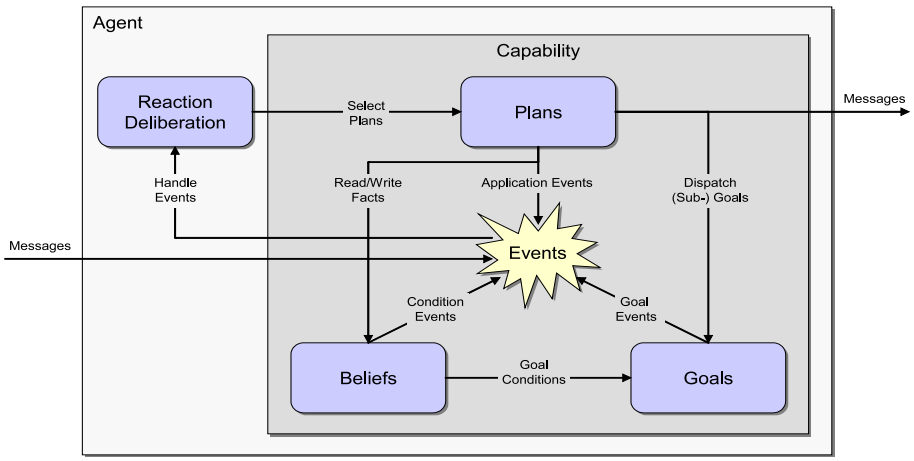

Abbildung 2.6: Jadex abstrakte Architektur, aus [PBL05b]

Plans: Pläne (Plans) werden in Jadex, im Gegensatz zu den beiden vorhergehenden Konzepten, in zwei Dateien spezifiziert. Der Kopf (head) eines Plans wird im ADF spezifiziert. Er enthält Angaben zu dem den Plan auslösenden Ereignis (trigger), zum verwendeten Rumpf (body), aber auch zu Bedingungen für die Anwendbarkeit des Plans (Precondition) und den Kontext (Contextcondition) in dem der Plan ausgeführt werden kann. Auslösende Ereignisse eines Plans können z.B. ein Ziel (goal) oder auch eine Nachricht (messagevent) sein. In der zweiten Datei, dem Rumpf des Plans, werden die Aktionen beschrieben die der Plan ausführen soll. Diese können Aktionen wie das Senden einer Nachricht oder das Erzeugen eines Subziels (Subgoal) sein. Der Rumpf ist vollständig in Java zu implementieren.

Capabilities Als weiteres zusätzliches Konzept integriert Jadex Capabilities. Capabilities kapseln, wie der Name vermuten lässt, Fähigkeiten die auf einer gemeinsamen Untermenge von Beliefs, Desires und Plans arbeiten. Die so spezifizierten Capabilities dienen den in ihr befindlichen Plänen, Zielen oder Beliefs als Begrenzung ihrer Gültigkeit (scope). Capabilities sind vergleichbar mit dem Modulkonzept.

Jadex Architektur

In Abbildung 2.6 ist die abstrakte Architektur eines Jadex Agenten dargestellt. Ein Jadex Agent arbeitet nach dem folgenden Prinzip. Die internen Konzepte „Belief“, „Plans“ und „Goals“, wie aber auch eintreffende Nachrichten, erzeugen im inneren eines Jadex Agenten spezifische Ereignisse. Diese Ereignisse werden in einer Liste (Event-list), die nicht dargestellt ist, verwaltet. Jedes Ereignis aus dieser Liste wird durch die Reaktions-/Deliberationskomponente möglichen

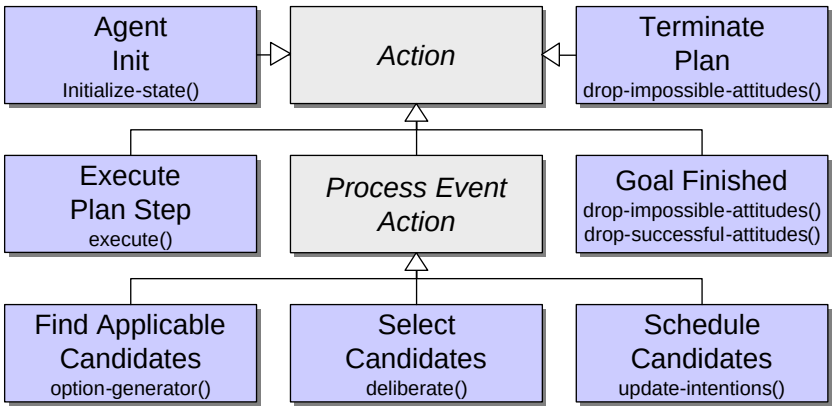

Abbildung 2.7: Menge an Meta-Aktionen, aus [PBL05a]

Plänen zugeordnet. Dieser Zuordnungsprozess läuft in zwei Teilen ab. Im ersten Schritt werden alle möglichen Pläne berechnet (matching) und im zweiten Schritt wird eine Auswahl aus der möglichen Menge getroffen (meta-level reasoning) die zur Ausführung gebracht werden soll. Die definitiv zur Ausführung ausgewählten Pläne werden in der Liste für bereitstehende Pläne (ready list), ebenfalls nicht dargestellt, gespeichert. Aus dieser Liste werden die Pläne nacheinander entfernt und begonnen zu bearbeiten. Ein Plan wird dabei solange bearbeitet bis er explizit eine Warteanweisung ausführt oder den internen Zustand des Jadex Agenten beeinflusst.

Jadex-Interpreter

Der Interpreterzyklus der in 2.2.2 vorgestellt wurde, arbeitet in jeder Abfolge des Zykluses dieselben Operationen in der gleichen Reihenfolge ab. Jadex beschreitet hier einen anderen Weg. Um die architekturelle Grundlage für ein Goal Deliberationsverfahren zu ebenen, wurde aus dem statischen, fest vorgegebenen Zyklus ein flexibler, wahlweise veränderbarer entwickelt. Dies geschah, indem die starre Abfolge der Aktionen aufgebrochen wurde und die einzelnen Aktionen in sogenannte Meta-Aktionen überführt wurden. So entwickelte sich z.B. aus der Operation `deliberate(options)` die Meta-Aktion `SelectCandidates`. In Abbildung 2.7 ist der komplette Satz an Meta-Aktionen mit Vererbungsbeziehungen dargestellt. Dieser Satz an Meta-Aktionen ist ausreichend um den originalen Interpreterzyklus nachzubilden.

Durch die Aufspaltung der Prozesskette des originalen Zyklus, ging die Information über die Reihenfolge der abzuarbeitenden Operationen verloren. Die neue Implementation benötigt deshalb eine Möglichkeit festzulegen bzw. festzustellen welche Meta-Aktion die nächste in der Aktionsabfolge sein soll. Dieses wird durch eine Agenda, die in Abbildung 2.8 dargestellt ist erreicht.

In der Agenda werden alle Meta-Aktionen aufgeführt die zur Ausführung

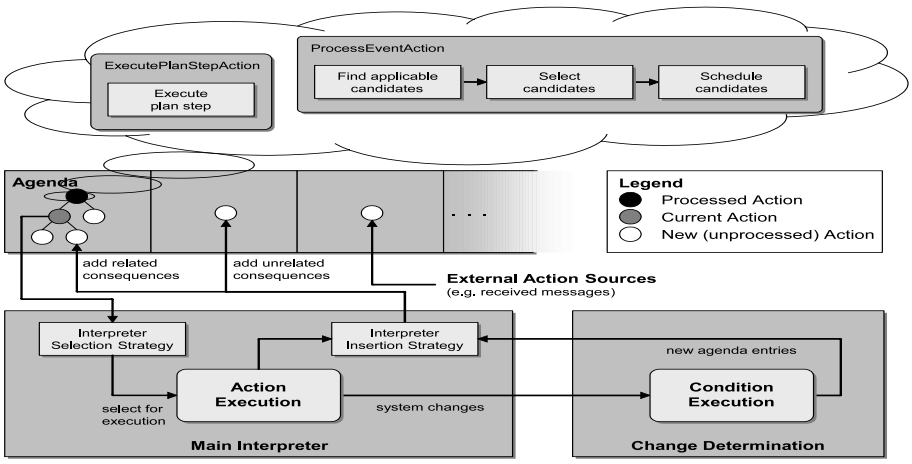

Abbildung 2.8: Jadex Interpreter aus [PBL05a]

bereitstehen. Dabei werden in Beziehung stehende Aktionen durch eine Baumstruktur, wie im ersten Eintrag der Agenda ersichtlich, dargestellt. Nicht in Beziehung stehende Aktionen erhalten jeweils einen eigenen separaten Eintrag. Meta-Aktionen können mit einer Vorbedingung versehen werden, die festlegt unter welchen Umständen die Meta-Aktion ausführbar ist. Sollte die Vorbedingung nicht mehr zutreffen, so ist die betreffende Meta-Aktion nicht mehr ausführbar und wird aus der Agenda entfernt. Entscheidungslogik in den Komponenten „Auswahl Strategie“ (Interpreter Selection Strategy) und „Einschub Strategy“ (Insertion Strategy) bestimmt die Auswahl bzw. den Ort des Einfügens von Aktionen. So entstehen z.B. die oben erwähnten Baumstrukturen, beim Einfügen in Beziehung stehender Aktionen. Nach der Auswahl einer geeigneten Aktion wird diese in der Komponente „Action Execution“ ausgeführt. Dadurch können neue Meta-Aktionen entstehen, die in die Agenda integriert werden müssen. Eine Ursache für das Entstehen neuer Aktionen ist das Auftreten von Seiteneffekten bei der Ausführung einer Meta-Aktion. Seiteneffekte, so z.B. die Änderung des Wertes eines Beliefs, können u.U. das Auslösen einer Bedingung nachsichziehen. Wenn es sich bei der betreffenden Bedingung um eine handelt, die angiebt in welcher Situation ein Ziel beendet werden soll, so resultiert das in der Eintragung eines neuen Agendaeintrags mit der Bezeichnung DropGoalAction. Verantwortlich für das Bestimmen der möglichen neuen Meta-Aktionen sind die Komponenten „Change Computation“ und „Consequence Determination“ im rechten unteren Bereich der Abbildung 2.8. Dabei kommt der Komponente „Change Computation“ die Berechnung aller möglichen Systemveränderung zu und der Komponente „Consequence Determination“ die Feststellung ausgelöster Bedingungen und Erzeugung daraus resultierender Meta-Aktionen.

2.4 Zusammenfassung

In diesem Kapitel wurden die Grundlagen für diese Diplomarbeit dargestellt. Hierbei ging es vorallem darum das sehr spezielle Thema dieser Arbeit in den Kontext der Multiagententhematik einzuordnen. Begonnen wurde mit einer Erläuterung von intelligenten Agenten und deren Eigenschaften, um anschließend thematisch auf Architekturen für Agenten überzugehen. Dort wurde speziell die BDI Architektur vorgestellt und ihre Konzepte erläutert. Dabei wurde deutlich, dass die BDI Architektur sich stark auf Intentionen konzentriert. Im Anschluss daran beschrieb dieses Kapitel zwei sehr bekannte Implementation der BDI Architektur. Dies war zum einen die Intelligent Resource Bounded Machinery (IRMA) und zum anderen das Procedural Reasoning System (PRS). Beide Implementationen sollten zeigen, wie eine BDI Architektur umgesetzt werden konnte. Im Schlussteil dieses Kapitels über Grundlagen der Arbeit wurde das Jadex System vorgestellt. Das Jadex System ist die Basis für Implementationen in dieser Arbeit und daher unabdingbar. Es wurde dargestellt, was Jadex ist, welche BDI Konzepte es umsetzt, wie es funktioniert und welche Ergänzungen Jadex zusätzlich zu den Standard BDI Konzepten besitzt.

Kapitel 3

Das Konzept Ziel

Philosophie	Belief	**Desire**	Intention
Theorie	Belief	**Ziel**	Intention
Implementation	Relationale DB (oder Objekt)	**Ereignis**	Plan

Abbildung 3.1: Unterschiede in der Umsetzung der BDI Begriffe, aus [WPH2001]

Bei der Betrachtung von proaktiven, intelligenten Agenten spielt das Konzept des Ziels eine zentrale Rolle. Ziele spiegeln die Motivatoren wieder die Ursache für das Handeln des Agenten sind. Jedoch führt ein Ziel im Vergleich zu einem Plan ein Schattendasein in aktuellen Agentenplattformen wie JACK [BRHL99], JAM [Hub99] oder PRS. So kritisieren die Autoren in [TPH2002a], dass Ziele zwar ein integraler Bestandteil theoretischer Arbeiten wie [RG91, CL93] sind, aber in praktischen Belangen wie der Repräsentation wenig Publikationsmaterial vorhanden ist. Abbildung 3.1 verdeutlicht dieses Manko. Sie zeigt auf, dass Ziele in gängigen Implementationen lediglich durch den flüchtigen Charakter eines Ereignisses dargestellt werden und somit eine konzeptionelle Lücke zwischen Theorie und Praxis vorhanden ist.

Ziel dieses Kapitels ist es, diese Lücke zu verkleiner. Dies geschieht indem die bis dato vorhanden Arbeiten zur Repräsentation und Strukturierung des Konzeptes Ziel vorzutragen. Insbesondere sollen dabei die Strukturen, die in Jadex umgesetzt sind und im Hinblick auf eine Goal Deliberationsverfahren von Wichtigkeit sind, vorgestellt werden. Dabei wird ausgehend von verschiedenen Sichtweisen auf ein Ziel und den durch BDI Theorien geforderten Eigenschaften ein Konstrukt erörtert, welches als eine gute Grundlage für ein Goal Deliberationsverfahren gelten kann.

3.1 Eigenschaften

Die Eigenschaften die einem Ziel zugesprochen werden können, sind vorallem durch die entsprechenden BDI-Theorien [RG91, CL93] geprägt. Ein Ziel besitzt

demnach folgenden Eigenschaften. Es ist

1. Konsistent
2. Persistent
3. Durchführbar
4. Bekannt
5. Unerreicht[1]

Konsistent: Als die grundlegende Eigenschaft von Zielen, betrachtet man die Konsistenz. Zwei Ziele eines Agenten sollen demnach keine Zustände beschreiben die sich gegenseitig logisch ausschließen. So ist z.B. denkbar, dass ein Agent keine zwei aktiven Ziele bearbeitet die die Systemzustände $\neg A$ und A beschreiben.

Persistent: Ziele eines Agenten sind persistent. Bearbeitet ein Agent ein Ziel so versucht er alle Möglichkeiten auszuschöpfen um dieses Ziel zu erreichen. Der Agent wird nicht ohne triftigen Grund das Ziel verwerfen, sondern nur wenn er keine Möglichkeit mehr sieht das Ziel erfolgreich zu bearbeiten oder wenn er bereits das Ziel erreicht hat.

Durchführbar: Es soll dem Agenten theoretisch möglich sein, die Ziele die er annimmt, auch zu erreichen. Dies bedeutet, dass der Agent glaubt, das ein Zustand der Domäne eingenommen werden kann in dem der durch das Ziel beschriebene Zustand gilt. Ist es dem Agenten nicht möglich an solch einen Zustand jemals glauben zu können, so soll das betreffende Ziel kein Ziel eines Agenten sein. Ziele unterschieden sich so von reinen Wünschen (Desires) [RG91].

Bekannt: Unverzichtbar für die Konstruktion eines Deliberationsverfahren ist, dass die Ziele einem Agenten bekannt sein sollten. Nur wenn der Agent weiß welche Ziele er besitzt, kann er folglich Interaktionen zwischen diesen Zielen erkennen [BPML04].

Unerreicht: Für den Standardtyp eines Ziels, den Typ „Achieve“ gilt, dass der Zustand den das Ziel beschreibt noch nicht erreicht ist. Nur wenn diese Situation zutrifft soll ein Agent das Ziel aktiv bearbeiten. Ist das Ziel bereits erreicht, so ist es kein Ziel mehr für den Agenten.

3.2 Prozeduraler Aspekt

Die prozedural Sichtweise auf Ziele, bei der auch von „To-Do Goals“ [vRvdHM2003] gesprochen wird, unterscheidet sich kaum von der Verwendung des Begriffs Prozedur oder Funktion. Annahmen die für die Ausführung einer Prozedur getroffen

[1] Diese Eigenschaft gilt nur für den Typ „Achieve“ eines Ziels

wurden entsprechen dabei den Vorbedingungen für die Bearbeitung des Ziels. Der Bezeichner einer Prozedur kann im Allgemeinen als Beschreibung für das Ziel betrachtet werden. Der Effekt den die Prozedur nach ihrer Ausführung erwirkt, ist mit dem Zustand den das Ziel beschreibt gleich zu setzen. Die Operationen der Prozedur sind in ihrer Gesamtheit als Plan für die Bearbeitung des Ziels zu sehen.

Ein Vorteil der prozeduralen Sichtweise ist, die immer vorhandene Implementation eines Plans zur Bearbeitung des Ziels. Somit kann sichergestellt werden, das der Agent nur solche Ziele besitzt, für die er auch stets eine Realisierung parat hat.

Nachteilig an der prozeduralen Sichtweise ist die enge Verknüpfung von erfolgreicher bzw. nicht erfolgreicher Zielbearbeitung mit erfolgreicher bzw. nicht erfolgreicher Planbearbeitung. Schlägt eine Prozedur fehl, so schlägt auch immer unweigerlich das durch diese Prozedur beschriebene Ziel fehl. Das bedeutet also, dass das Ergebnis der Bearbeitung des Ziels immer von der Bearbeitung des Plans abhängt.

3.3 Deklarativer Aspekt

Die deklarative Sichtweise eines Ziels ist die Standardsichtweise auf Ziele in der künstlichen Intelligenz. Sie wird auch als „To-Be Goal“ bezeichnet [vRvdHM2003] und kennzeichnet sich durch die Spezifizierung eines Zustandes. Bei diesem spezifizierten Zustand handelt es sich um den Zustand, den der Agent durch die Bearbeitung des Ziels erreichen will. Die Zustände werden häufig in einer temporalen Logik spezifiziert.

Die deklarative Sichtweise wird als Voraussetzung für die Möglichkeit der Deliberation auf Zielen gesehen. So stellen die Autoren in [WPHT2002] fest, dass das Fehlen einer deklarativen Sichtweise die Möglichkeit der Deliberation auf einem Ziel ad absurdum führen würde. Durch das Vorhandensein einer deklarativen Sichtweise, so argumentieren Winikoff et al. wird einem Agenten u.a folgendes ermöglicht.

- Der Agent kann erkennen, ob ein Ziel erreicht ist. Da das Ziel explizit einen Zustand beschreibt, kann nachgeprüft werden, ob dieser geforderte Zustand bereits durch die „Beliefs“ des Agenten erfüllt wird. Ist dem so, so kann das betreffende Ziel als erreicht angesehen werden. (vgl. Eigenschaft „Unerreicht“ Kapitel 3.1)
- Ziele können auf Inkonsistenzen untereinander überprüft werden. Für zwei Ziele kann festgestellt werden, ob die Zustände die sie beschreiben, sich logisch ausschließen. (vgl. Eigenschaft „Konsistent“)
- Für ein Ziel ist feststellbar, ob es möglich ist dieses auszuführen. Durch die Verwendung von Zusatzinformation in deklarativer Form, z.B. in Form einer „Failurecondition“ [WPHT2002] kann festgestellt werden, ob ein Zustand der Domäne vorliegt, der die erfolgreiche Bearbeitung des betreffenden Ziels unmöglich werden lässt.

Eine deklarative Sichtweise unterstützt also die Umsetzung der in den BDI Theorien vorgeschlagenen Eigenschaften eines Ziels. Das trägt dazu bei, dass die in der Einführung dieses Kapitels erwähnte Lücke zwischen Theorie und Implementation verringert werden kann.

Eine deklarative Sichtweise sollte, insbesondere wenn man Schlussfolgerungen über Ziele anstellen möchte, die Wahl sein. Inwieweit dazu eine temporale, oder einfachere Logik verwendet werden muss, hängt im einzelnen von der Mächtigkeit ab die man mit dieser Spezifizierung erreichen möchte und von dem Anwendungsbereich. Zu beachten ist dabei jedoch folgender Gesichtspunkt, den van Lamsweerde beschreibt.

> „More formal specification yield more powerful reasoning schemes at the price of higher specification effort and lower usability by non-experts“ [vLam2001]

Es ist also auch zu berücksichtigen wer die betreffenden Informationen für Ziele spezifizieren soll.

Eine Alternative zu der rein formalen Spezifikation stellt eine semi-formale Spezifikation dar [vLam2001]. Der Zweck der semi-formalen Spezifikationen ist, eine einfacherer Art der Spezifizierung von deklarativer Information zu ermöglichen, aber dennoch einer Analyse zugänglich zu sein. Solche semi-formalen Spezifikationen deklarieren Ziele in Form ihres Typs, ihrer Attribute und ihrer Beziehungen. Sie verwenden häufig Schlüsselworte für die Darstellung von temporalen Aspekten des Zustandes eines Ziel. Zum Beispiel wird in [DvLF93] für den temporalen Ausdruck $\Diamond Q$ das Schlüsselwort bzw. der Zieltyp „Achieve“ festgelegt. Dadurch wird durch das Schlüsselwort implizit ein Zustand Q beschrieben der irgendwann (somtimes) in der Zukunft eintrifft.

Eine semi-formale deklarative Sichtweise wird auch in Jadex verwendet. Sie verwendet die eben beschriebenen Schlüsselwörter und ermöglicht durch die Spezifikation von Zuständen in der „Embedded Reasoning Language“ eine vereinfachte Darstellung der Situationen die durch die Ziele erreicht werden soll. Welche zusätzlichen Strukturen man mit einem Ziel in Verbindung bringen kann und welche insbesondere im Zielkonzept in Jadex verwirklicht sind soll im folgenden im Abschnitt über Attribute erläutert werden.

3.4 Attribute

Attribute dienen der Umsetzung der Merkmale eines Objektes und somit auch einer Unterscheidung zwischen verschiedenen ausgestalteten Instanzen. In diesem Abschnitt werden daher gebräuchliche Attribute eines Ziels behandelt. Dabei handelt es sich um das zentrale Attribut Kern (core), die Attribute Wichtigkeit, Nutzen und Dringlichkeit, aber auch um die Attribute Attitüde und dynamischer Zustand.

3.4.1 Der Kern eines Ziels

Der grundlegendste Bestandteil der ein Ziel auszeichnet, ist der Kern (core) eines Ziels [Bea94]. Ein Kern stellt einen Zustand der Welt dar, zu dem der Agent eine bestimmte Motivation entwickeln kann. Diese Motivation kann sich in verschiedensten Weisen artikulieren. So bestehen folgende Möglichkeiten eine Motivation gegenüber einem Zielkern herauszubilden. Der Agent

- hat die Motivation den durch den Kern beschriebenen Zustand zu erfüllen,
- hat die Motivation den Zustand nicht zuerfüllen,
- hat die Motivation den beschriebenen Zustand aufrecht zu erhalten und
- hat die Motivation das Erreichen des Zustands zu vermeiden.

Anhand der Unterscheidung zwischen diesen Motivationen, kann ein Ziel in verschieden Typen klassifiziert werden [Hub99, vLam2001, DvLF93, BPML04]. Es werden daher die Typen „Achieve“, „Maintain“, „Cease“ und „Avoid“ unterschieden. Der Typ „Achieve“ stellt dabei die im ersten Punkt erwähnte Motivation dar, während der Typ „Cease“ die Motivation des zweiten Punktes, also genau das Gegenteil zum Typ „Achieve“, darstellt. Der Typ „Maintain“ stellt die Motivation des dritten Punktes dar und der Typ „Avoid“ die des vierten. Auch diese beiden Typen sind jeweils das Gegenteil voneinander.

Solch eine Klassifizierung in die Typen „Achieve“, „Maintain“, „Cease“ und „Avoid“ bezeichnet man als eine Klassifizierung nach zeitlichen Aspekten oder auch als temporale Klassifizierung. Diese Klassifizierung ist die am weitesten verbreiteteste in Agentensystemen. Insbesondere die Verwendung des Typs „Achieve“ ist dabei von grosser Bedeutung [BPML04], da er den Typus Ziel vertritt, der dem menschlichen Verständnis eines Ziels am nächsten kommt. Es wird daher im Folgenden eine Konzentrierung auf diesen Typ und den Typ „Maintain“ erfolgen.

Im Zusammenhang mit dieser Typisierung werden auch die Kerne entsprechend bezeichnet. So umschreibt eine „Targetcondition“, einen Zustand der Welt, der den Kern für ein als Ziel vom Typ „Achieve“ darstellt und eine „Maintaincondition“, einen Zustand der Welt der dem Kern für ein als Maintain klassifiziertem Ziel entspricht. Für die restlichen Ziele der temporalen Klassifizierung ist eine solche Einteilung ebenso denkbar.

Zusätzliche Zustandsbeschreibungen für einen Kern

Für einen Kern und somit auch indirekt für ein Ziel lassen sich weitere Zustandsbeschreibungen spezifizieren, die die Rahmenbedingungen der Bearbeitung des Ziels konkretisieren. Diese Zustände sind teilweise hervorgerufen wurden, um die Eigenschaft „persistent“ und „durchführbar“ umzusetzen. So ist z.B. eine „Contextcondition“, „Dropcondition“ und „Failurecondition“ vorstellbar. Ein Zustand der durch eine „Contextcondition“ beschrieben wird, soll den Zustand der Welt beschreiben der gelten muss, damit die Ausführung des betreffenden Ziels gesichert ist. Die „Dropcondition“ andererseits soll die Situation angeben, die wenn

sie entritt, einen Grund dafür bietet das Ziel nicht mehr zu bearbeiten. Diese Zustandsbeschreibung ist also als eine Umsetzung der Eigenschaft persistent zu sehen. Und schliesslich die „Failurecondition“, die eine Situation beschreibt in der es nicht mehr möglich ist den betreffenden Kern noch jemals zu erreichen. Das bedeutet, dass mit dieser Zustandsbeschreibung die Eigenschaft der Durchführbarkeit eines Ziels spezifiziert werden kann. Die „Failurecondition“ gilt im besonderen für den Zieltyp „Achieve“.

Unabhängig von einer temporalen Klassifizierung gibt es andere Möglichkeiten Ziele zu klassifizieren. So wird in [vLam2001, BPML04] weiterhin zwischen sogenannten Hard- und Softgoals unterschieden. Hardgoals stellen dort Ziele dar, die die Dienste, die ein System anbietet, codieren. Softgoals hingegen repräsentieren nicht funktionale Eigenschaften wie die Qualität eines Systems. Vorstellbar sind hier z.B. Anforderungen hinsichtlich der Sicherheit oder Performanz. Andere Typisierungen als die temporale Typisierung, werden in dieser Diplomarbeit keine Berücksichtigung finden. Zusätzlich zu dem für ein Ziel grundlegenden Kern werden weitere Attribute vorgeschlagen.

3.4.2 Wichtigkeit

Das wohl entscheidendste und am häufigsten verwandte Attribut, neben dem Kern, ist das Attribut Wichtigkeit. Das Attribut Wichtigkeit soll eine Darstellung für die Abwägung zwischen den Kosten und Nutzen des Erfüllens bzw. des Nichterfüllens eines Ziels sein [Bea94]. Im Allgemeinen kann die Berechnung der Wichtigkeit eines Ziels ein beliebig komplexer geistiger Vorgang sein, der verschiedene Parameter wie Zeit und Nutzen integriert. Die Bedeutung der Spezifikation einer Wichtigkeit, liegt in der Ordnung die sie auf Zielen ermöglicht. Somit hat sie einen erheblichen Einfluss auf die Entscheidung des Agenten, ob ein Ziel zur Ausführung angenommen wird. Konkrete Beispielwerte die diesem Attribut zugeordnet werden können sind in [SBW94, BS93] angegeben. Dort sind Abstufungen wie neutral, niedrig, mittel, hoch, oder unbekannt vorgeschlagen. Es gibt aber auch die Möglichkeit der Spezifizierung durch numerische Werte (Null bis eins) wie in [DvLF93] erwähnt wird. Vielfach lässt sich die Verwendung des Attributs Wichtigkeit in konkreten Systemen wie Hap [Loy97] und InterRap [MP93] oder in Vorschlägen zu Systemen wie [TPH2002a] in Form anderer Bezeichner wie Priorität oder Präferenz ausmachen. Diese Differenzierung ist aber lediglich syntaktischer Form, semantisch entsprechen sie dem Begriff der Wichtigkeit.

3.4.3 Nutzen

Ein Attribut Nutzen kann als eigenständiges Attribut oder als Bestandteil der Berechnung einer Wichtigkeit angesehen werden. Ein Nutzen eines Ziel wird nicht immer klar von der Wichtigkeit eines Ziels abgetrennt und so teilweise austauschbar verwendet [Hub99]. Die Spezifikation eines Nutzen für ein Ziel beruht vermutlich auf Einflüssen aus der Ökonomie, die vorallem im Bereich

der „Decision Theory“ mit einem Nutzen von Aktionen arbeitet. Ein Attribut Nutzen wird praktisch im System JAM [Hub99] verwendet. Weitere Erwähnung findet dieses Attribut in zahlreichen Publikationen die mögliche Attribute für Ziele beschreiben [vLam2001]. Im Vergleich zur Wichtigkeit fristet dieses Attribut eher ein Schattendasein.

3.4.4 Dringlichkeit

Dringlichkeit soll den vielfach im alltäglichen Leben gebrauchten temporalen Aspekt eines Ziels wiedergeben. Dringlichkeit ist dabei im einfachen Fall ein Indikator für die Zeit die noch verbleibt um das Ziel, speziell bei Zielen vom Typ Achieve, zu erreichen und indiziert somit einen gewissen Grad an Wichtigkeit der monoton mit der Zeit wächst [Bea94]. Solch eine Dringlichkeit nennt man deadline oder auch terminale Dringlichkeit [Bea94]. Eine Umsetzung dieses Attributs ist gut in [NL96] zu erkennen. Dort wird ein Ziel als eine Struktur definiert die neben bekannten Bestandteilen wie dem Kern, dort als Proposition p bezeichnet und einer Wichtigkeit, bezeichnet als i_{max}, eine Spezifizierung eines temporalen Kontext enthält. Dieser temporale Kontext wird durch Angabe von zwei Zeitpunkten, Beginn t_{dt} und Ende t_{dl} und der Angabe der Dauer der Ausführung des Ziels $\Delta_a t$ festgelegt. Durch die Berechnung eines letzmöglichen Zeitpunktes t_{max} für den Start der Ausführung des betreffenden Ziels bei dem die Dringlichkeit der Bearbeitung am höchsten ist wird ein monoton ansteigende Kurve bis zu diesem Zeitpunkt berechnet. Mit fortschreitender Zeit gibt der Verlauf dieser Kurve die Dringlichkeit für das spezielle Ziel an. Anhand dieses Beispiels lässt sich gut die Beziehung zwischen Dringlichkeit und Wichtigkeit erläutern. Man kann sagen, dass die Wichtigkeit der Dringlichkeit übergeordnet ist. Dringlichkeit ist ein Aspekt eines Ziels der zur Bestimmung der Wichtigkeit eines Ziel herangezogen werden kann. Insbesondere für die Bestimmung einer dynamischen Komponente der Wichtigkeit spielt Dringlichkeit eine entscheidende Rolle. Eine genauere Betrachtung dieses Attributs und dem Umgang damit wird in 4.5.1 gegeben.

3.4.5 Attitüde (Commitment Status)

Um es einem Agenten zu ermöglichen eine Attitüde gegenüber seinen Zielen zu artikulieren, ist es notwendig einem Ziel verschiedene Zustände zuzusprechen. Diese Zustände werden in dem Attribut Verpflichtungzustand (Commitment Status) zusammengefasst. Die Notwendigkeit der Darstellung einer Attitüde gegenüber Zielen, ist durch das Ziel einen Deliberationsmechanismus zu entwickeln begründet. Der Agent soll entscheiden können, ob er ein Ziel aktiv bearbeitet, von ihm abkehren will oder es zurückstellen möchte. Das bedeutet, dass ein Ziel, aus Sicht eines Agenten, sich in eben diesen Zuständen befinden kann. In Abbildung 3.2 kann man die Darstellung von Attitüden eines Agenten gegenüber einem Ziel erkennen. Die hier verwendete Darstellung einer Attitüde stammt aus der Agentenplattform Jadex. Diese implementiert eine Attitüde gegenüber Zielen durch folgende Zustände:

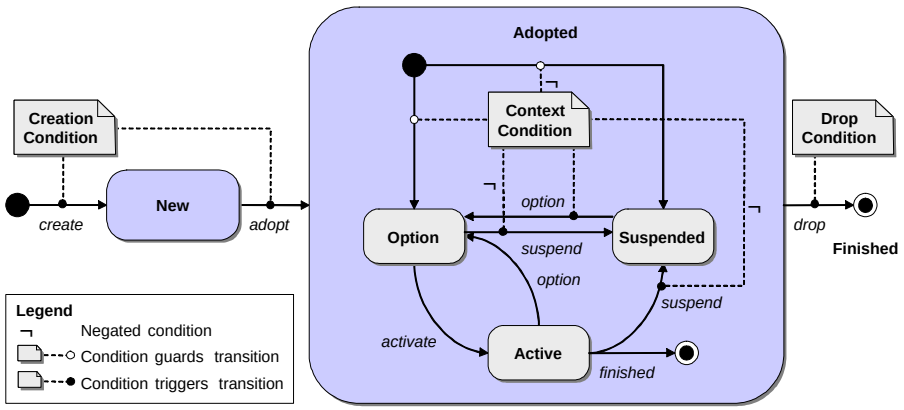

Abbildung 3.2: Modell eines Ziels, aus [BPML04]

New Ein unmittelbar erzeugtes Ziel wird durch diesen Zustand charakterisiert. Dieses Ziel liegt momentan nur als eine Idee vor und befindet sich noch nicht im Besitz des Agenten.

Adopted Der Agent hat dieses Ziel aufgenommen und entscheidet innerhalb dieses komplexen Zustandes wie mit diesem Ziel weiter verfahren werden soll.

Option Befindet sich das Ziel in diesem Zustand, so wird es vom Agenten als zu bearbeitende Möglichkeit (Desire) angesehen. Der Agent hat eine noch unentschiedene Haltung gegenüber diesem Ziel und muss durch geeignete Kriterien entschieden können, ob das Ziel bearbeitet werden soll.

Active Ziele im Zustand „Active“ sind Ziele für die der Agent entschieden hat, dass sie bearbeitet werden sollen. Der Zustand „Active“ unterteilt sich in spezifischere Subzustände die in 3.4.6 näher erläutert werden.

Suspended Ist der Agent zu der Entscheidung gelangt, dass im Augenblick das Ziel in seiner Bearbeitung unterbrochen werden sollte, so drückt er dies mit der durch diesen Zustand repräsentierten Haltung aus.

Dropped Dieser Zustand symbolisiert einen Endzustand eines Ziels. Mit dieser Haltung drückt der Agent aus, dass er als aussichtslos erachtet dieses Ziel noch jemals zu erreichen. Das kann aus dem Grund erfolgen, dass das Ziel bereits erreicht ist oder dass das Ziel nicht mehr erreichbar erschient.

3.4.6 Dynamischer Zustand

Das Attribut dynamischer Zustand beschreibt die Zustände eines Ziels die es während seiner aktiven Phase einnehmen kann. Mit diesen zusätzlichen Zuständen können Bearbeitungsmuster unterschiedlicher Zieltypen dargestellt werden.

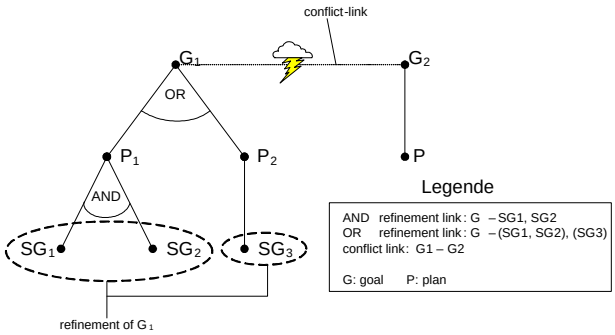

Abbildung 3.3: „Goal-links“ für Ziele

Die Zuständen dieses Attributs sind Subzustände des Zustandes „Active“ des Attributs Attitüde. In [BPML04] sind u.a. für die Zieltypen Achieve und Maintain konkrete Angaben gemacht wurden, wie solche Zustände aussehen könnten. In Abbildung 3.4 und 3.5 sind diese Zustände dargestellt. Der Zieltyp Achieve besitzt die dynamischen Zustände „In Process“, „Succeeded“, „Unknown“ und „Failed“. Dabei bedeutet „In Process“ das das Ziel momentan bearbeitet wird. Succeeded bedeutet, dass das Ziel erfolgreich bearbeitet wurde und es somit in einen Endzustand übergeht. Der Zustand „Unknown“ bedeutet, dass der Bearbeitungsstatus nicht ermittelt werden konnte und der Zustand „Failed“ dass das Ziel erfolglos bearbeitet wurde und in einen Endzustand übergeht.

Für den Zieltyp Maintain wurden in [BPML04] die dynamischen Zustände „Idle“, „In Process“, „Unmaintainable“ und „Unknown“ vorgeschlagen. Der Zieltyp Maintain besitzt im Gegensatz zum Typ Achieve keinen ausgezeichneten Endzustand. Der Zustand „Idle“ stellt den Zustand dar den das Ziel einnimmt wenn der zu überwachende Zustand mit dem aktuellen Zustand übereinstimmt. Der Zustand „In Process“ entspricht semantisch dem des Achieve Ziels. Der Zustand „Unmaintainable“ wird eingenommen wenn der Agent keine Pläne mehr besitzt um den Wunschzustand herzustellen und der Zustand „Unknown“ ist wie der Zustand „In Process“ ,vergleichbar mit den entsprechenden des Zieltyps Achieve.

3.5 Hierarchien zwischen Zielen

Bei der Bearbeitung eines Ziels durch einen Plan kann es vorkommen, dass Subziele erzeugt werden. Diese Subziele stellen eine Verfeinerung (refinement) des Ursprungsziels dar und stehen in gewisser Beziehung zu diesem. Diese Beziehung muss bei einer Betrachtung der Struktur eines Ziele berücksichtigt werden. In [DvLF93, vLam2001] werden solche Beziehungen durch den Begriff eines „Goal-Link“ überschrieben. Im speziellen Fall der Subziele spricht man auch von einem „Refinement-link“. Nach [DvLF93, vLam2001] gestalten sich die Be-

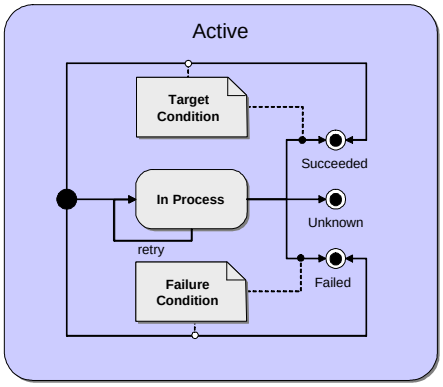

Abbildung 3.4: Active Zustand des Zieltyps Achieve [BPML04]

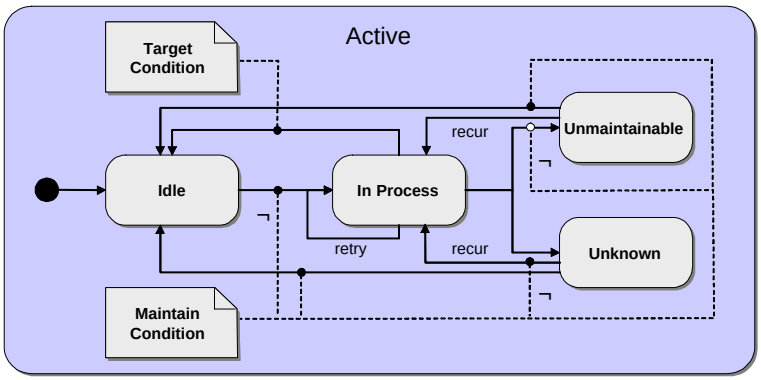

Abbildung 3.5: Active Zustand des Zieltyps Maintain [BPML04]

ziehungen im Allgemeinen in einer hierarchischen Art und Weise und können durch „AND“ oder „OR“ Graphen dargestellt werden. Eine „AND“ Verfeinerung (AND-refinement link) von Subzielen beschreibt dabei den Fall in dem alle so verfeinerten Subziele erfolgreich bearbeitet werden müssen um eine erfolgreiche Bearbeitung des Ursprungsziels zu gewährleisten. Ein „OR“ Verfeinerung hingegen beschreibt alternative Verfeinerungen eines Ziels. Für die erfolgreiche Bearbeitung des Ursprungsziels ist nur die erfolgreiche Bearbeitung von mindestens einer Verfeinerung notwendig.

Als Vorschlag für die Umsetzung einer solchen Hierarchie ist in [DvLF93] die Deklaration eines `ReducedTo` Attributs vorgesehen. `ReducedTo` führt alle Subziele des betreffenden Ziels auf die eine „AND“ Verfeinerung darstellen. Eine „OR“ Verfeinerung wird duch die Verknüpfung von zwei oder mehr `ReducedTo` Attributen mit `or` erreicht.

Subziele eines Ziels tragen immer zur Bearbeitung des Ursprungsziels bei. Jedoch gibt es auch Ziele die nicht in einem Ziel – Subziel Verhältnis stehen und somit keine Verfeinerung darstellen. Solche Ziele können sich in ihrer Bearbeitung stören oder sogar hemmen. Beziehungen dieser Art werden als Konflikte bezeichnet (conflict-link) und im anschließenden Anschnitt 3.6 genauer untersucht.

Abbildung 3.3 fasst alle Beziehungen zwischen Zielen zusammen.

3.6 Interaktionen zwischen Zielen

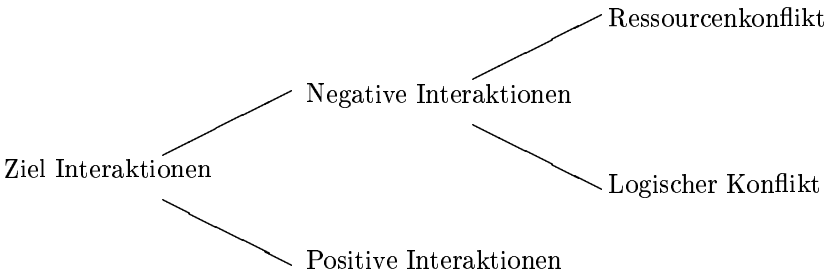

Abbildung 3.6: Interaktionsmöglichkeiten zwischen Zielen

Intelligente Agenten, wie sie in dieser Diplomarbeit betrachtet werden, arbeiten nicht nur an einem Ziel und auch nicht nur an Zielen die sequentiell abgearbeitet werden, sondern sehen sich häufig mit der Anforderung konfrontiert viele komplexe Ziele nebenläufig zueinander zu bearbeiten. Insbesondere

beim Einsatz dieser Agenten in Realweltanwendungen ist diese Anforderung besonders stark ausgeprägt. Durch diese Tatsache können Ziele und deren Abarbeitung nicht mehr isoliert voneinander betrachtet werden, sondern müssen in Verbindung mit anderen Zielen und deren möglichen Ausführungen gesehen werden. Dabei können verschiedenste Interaktionen zwischen ihnen auftreten die die Erfüllung aller dem Agenten auferlegten Ziele negativ oder auch positiv beeinflussen. Diese Interaktionen werden in diesem Kapitel untersucht.

Interaktionen zwischen Zielen sind eine Thematik mit der sich insbesondere auf der Macroebene, also der Ebene zwischen mehreren Agenten ausgiebig beschäftigt wurde. Auf der Microebene, also der Ebene eines einzelnen Agenten ist dagegen nicht so nachhaltig geforscht wurden. Hier müssen Erkenntnisse aus der Macroebene transferiert werden. Ein weitere wichtiger Ansatz für das Verständnis von Interaktionen zwischen Zielen eines einzelnen Agenten bietet das klassische Planen [Wil83]. Dort wurden bereits Erkenntnisse erlangt wie Ziele eines Agenten interagieren können. Dies war dort notwendig, um eben jene Interaktion bei der Konstruktion der Pläne zu berücksichtigen. In diesem Kapitel wird auf die Erkenntnisse aus dem Bereich des Planens zurückgegriffen insbesondere auf Erkenntnisse aus [Wil83].

Die Breite der Interaktionsmöglichkeiten reicht von Interaktionen zwischen Zielen innerhalb eines Agenten bis zu Interaktionen zwischen Zielen unterschiedlicher Agenten. Im ersten Fall werden diese Interaktionen als interne Interaktionen bezeichnet, im zweiten Fall als externe Interaktionen [Wil83]. In beiden Fällen können diese Interaktionen wiederum positive oder negativ ausfallen.

In dieser Arbeit werden nur positive und negative interne Interaktion zwischen Zielen eines einzelnen Agenten betrachtet und deshalb im folgenden einfach von den positiven und negativen Interaktionen gesprochen.

3.6.1 Negative Interaktionen (Konflikte)

Negative Interaktionen, bezeichnet als Zielkonflikte, beeinflussen die rationale Bearbeitung von Zielen negativ. So kann es passieren, dass Ziele die negativ interagieren, nicht erfolgreich bearbeitet werden können, und somit in einem Abbruch eines Ziels oder mehrerer Ziele resultieren. Daher ist es notwendig, dass der Agent solche Situationen erkennen kann, um entsprechend darauf zu reagieren. Negative Interaktionen werden deshalb der primäre Ansatzpunkt der Betrachtungen der Interaktionen zwischen Zielen sein.

Konflikte zwischen Zielen kommen immer dann zustande, wenn die betreffenden Ziele Zustände beschreiben, die sich gegenseitig ausschließen. Das bedeutet für Ziele, dass das Erreichen des einen Ziels das Erreichen des anderen Ziel unmöglich macht. Zielkonflikte lassen sich in primäre Zielkonflikte (intrinsischer Konflikt) und sekundäre Zielkonflikte einteilen.

Primäre Zielkonflikte bezeichnen Konflikte zwischen Zielen in ihrem eigentlichen Sinne. Hierbei konzentriert sich die Analyse auf die Kerne der beteiligten Ziele. Die sekundären Zielkonflikte hingegen sind eigentlich Plankonflikte.

Primärer Zielkonflikt

Der primäre Zielkonflikt zeichnet sich dadurch aus, dass die Kerne der betroffenen Ziele Zustände beschreiben die sich unmittelbar gegenseitig ausschließen. So kann für ein Ziel A die Aussage getroffen werden, dass es mit einem Ziel B in primärem Zielkonflikt steht, wenn Ziel A einen Zustand p beschreibt und Ziel B einen Zustand q beschreibt für die gilt:

$$p \Rightarrow \neg q$$

Das bedeutet, dass Ziel A einen Zustand erreichen will, der genau das logische Komplement zu dem Zustand ist, der durch Ziel B beschrieben wird. Es ist leicht verständlich, dass beide Zustände nicht parallel erreicht werden können.

Für einen Agenten kann die Erkennung eines solchen Konflikts unterschiedlich diffizil sein. Ganz in Abhängigkeit wie der Zustand spezifiziert wird, formal oder objektorientiert wie in Jadex, variiert der Aufwand von gering bis extrem hoch bzw. unentscheidbar. Zusätzlich dazu kann für die Beurteilung, ob ein Zustand einen anderen ausschließt, Hintergrundwissen benötigt werden. Das soll an einem Beispiel verdeutlicht werden. Es wird angenommen, dass ein Agent das Ziel A besitzt an dem Ort Hamburg zu sein und gleichzeitig das Ziel B an dem Ort Berlin zu sein. Beide Ziele sollen parallel ausgeführt werden. Für einen Agenten ohne Hintergrundwissen, ist das Erkennen des Konflikts schwierig. Er benötigt die Zusatzinformation, dass es für ein untrennbares Objekt unmöglich ist an zwei verschiedenen Orten zur gleichen Zeit zu sein. Zustände die sich gegenseitig ausschließende und in keinem Fall gleichzeitig erreicht werden können, werden sich logisch ausschließende Zustände genannt.

Sekundärer Zielkonflikt

Auch beim sekundären Zielkonflikt besteht die Möglichkeit des Konflikts durch sich gegenseitig ausschließende Zustände. Jedoch sind im Konflikt stehenden Zustände hier nicht als Zielkerne erkenntlich, sondern treten erst durch die Abarbeitungen der betroffenen Ziele auf. Es können so Vorbedingungen für die Ausführung eines Plans wie auch Effekte der Ausführung eines Plans sich gegenseitig wie auch die Zielkerne ausschließen. Ein ganz besonderes Augenmerk bei der Behandlung der sekundären Zielkonflikte, wird auf die Ressourcenkonflikte gelegt.

Ressourcenkonflikt Zur Abarbeitung der Ziele eines Agenten werden Pläne benutzt. Diese Pläne benötigen unter Umständen Zugriff auf Ressourcen um die Aktionen in diesen Plänen ausführen zu können. Dabei kann man zwischen zwei verschiedenen Ressourcentypen unterscheiden. Es handelt sich hierbei um Ressourcen die verbraucht werden und Ressourcen die wiederverwendet werden können.

Verbrauchbare Ressourcen: Ressourcen die verbraucht werden, stehen einem Agenten in ihrer Funktionalität nur einmal zur Verfügung. Zur Verdeutlichung was dies für die Ziele eines Agenten bedeuten kann, wird ein

kleines Beispiel herangezogen. Es wird angenommen, dass ein Agent zwei Ziele besitzt die aufgrund ihrer Bearbeitung die Ressource X jeweils in Stückzahl fünf und zehn benötigen. Daraus folgt, dass der Agent für die erfolgreiche Bearbeitung beider Ziele eine Anzahl von 15 Stück der Ressource X benötigt. Diese Anzahl ist unabhängig von der Art der Bearbeitung der beiden Ziele. Egal ob die Ziele parallel oder sequentiell bearbeitet werden, die Anzahl ändert sich nicht, da verbrauchte Ressourcen nicht mehr benutzt werden können. Besitzt der Agent nun aber nur eine Menge von zehn Stück der Ressource X so entsteht ein Ressourcenkonflikt zwischen diesen beiden Zielen. Das bedeutet für den Agenten, dass er nur maximal eins der beiden Ziele erreichen kann.

Wiederbenutzbare Ressourcen: Ressourcen die wiederbenutzt werden können, stehen einem Agenten in ihrer Funktionalität mehr als einmal zur Verfügung. Jedoch ist dieses Plus an Funktionalität zeitlich restringiert. Das bedeutet, dass während der Benutzung einer wiederverwendbaren Ressourcen diese für andere Aktionen nicht zur Verfügung steht, nach Beendigung der Benutzung jedoch wieder in den Pool von Ressourcen zurückgelegt wird und somit einer erneuten Benutzung zur Verfügung steht. Zur Verdeutlichung wie dieses sich auf die Ziele eines Agenten auswirken kann folgt ein kleines Beispiel. Es wird angenommen, dass eine Ressource Y in einer Stückzahl von eins vorhanden ist. Zwei Ziele eines Agenten benötigen zu ihrer erfolgreichen Bearbeitung, diese Ressource jeweils in einer Stückzahl von eins. Daraus folgt, dass der Agent für eine erfolgreiche Bearbeitung beider Ziele eine Anzahl von zwei Stück der Ressource Y bei paralleler Bearbeitung und eine Anzahl von einem Stück der Ressource Y bei sequentieller Bearbeitung benötigt. Es ist also beim Typ der wiederverwendbaren Ressourcen festzustellen, dass ein Konflikt zwischen zwei Plänen unter Umständen vermieden werden kann, wenn die Ausführung der Pläne entsprechend sequentiell erfolgt.

Pläne benötigen also zur Bearbeitung der ihnen zugeteilten Ziele möglicherweise Ressourcen, die nicht in unmittelbar erforderlicher Stückzahl vorhanden sind oder sie benötigen Ressourcen die nicht gleichzeitig benutzbar sind. In beiden Fällen kann man im Allgemeinen von einer Form des Ressourcenkonflikts sprechen.

Zusätzlich zur Betrachtung des Ressourcentyp, kann man die Ausführung eines Ziels durch verschiedene Pläne in Betracht ziehen. Dadurch ergeben sich Ressourcen die notwendigerweise bei der Bearbeitung eines Ziels durch die zur Verfügung stehenden Pläne benötigt werden und Ressourcen die möglicherweise bei der Bearbeitung spezieller Pläne benötigt werden. Welche Auswirkungen das auf die Interaktionen zwischen Zielen haben kann wird in Kapitel 4.3.2 näher erläutert werden.

Zeit: Ein weiterer Typ Ressource ist die Zeit. Sie besitzt im Vergleich zu den beiden oben genannten Ressourcentypen einen Sonderstatus unter den Ressourcen. Zeit ist eine spezielle Ressource, die allen Agenten im glei-

chen Umfang zur Verfügung steht. Ein Agent besitzt keinen direkten Zugriff auf sie und Zeit wird auch ohne die direkte Nutzung durch Agenten verbraucht. Wie kann es dann zu Konflikten bezüglich der Zeit kommen? Dazu wird angenommen, dass ein Agent in einer Domäne arbeitet, in der Zeit eine wichtige Rolle, bei der Bearbeitung der Ziele, spielt. Zunächst einmal benötigt ein Agent zur Ausführung seiner Pläne, oder abstrakter zur Umsetzung seines Ist-Zustandes in seinen Soll-Zustand, eine bestimmte Zeit. Nun kann es passieren, dass das Erreichen des Soll-Zustandes eine bestimmte Soll-Zeit voraussetzt. Daraus folgt für die Umsetzung des Ist-Zustandes, also für die Ausführung des betreffenden Plans ein bestimmter Startzeitraum, sodass bei Beginn einer Ausführung innerhalb dieses Zeitraumes eine Beendigung zur Soll-Zeit garantiert werden kann. Hier tritt jetzt ein zweites Ziel und damit eine zweiter Plan ins Spiel. Dieses Ziel fordert ebenso wie das vorher erwähnte eine Umsetzung des Soll-Zustandes zu einer Soll-Zeit. Dabei erfordert dieses Ziel ebenfalls eine gewisse Zeit für die Umsetzung, so dass eine sequentiell Ausführung beiden Pläne nicht möglich ist. Die beiden Pläne müssen teilweise parallel bearbeitet werden. Ein Zeit-Konflikt tritt nun auf, wenn eben diese parallele Ausführung aus Gründen von Ressourcenmangel oder durch Auftreten von sich gegenseitig ausschließenden Zuständen unterbunden wird. Das hat zur Folge, dass eines der beiden Ziele nicht erfolgreich beendet werden kann.

3.6.2 Positive Interaktionen

In diesem Abschnitt werden positive Interaktionen zwischen Zielen betrachtet. Positive Interaktionen haben eine nicht so bedeutende Wirkung wie negative Interaktionen, da sie lediglich optimierenden Charakter besitzen. Positive Interaktionen haben auch nicht zur Folge dass möglicherweise bestimmte Ziele eines Agenten nicht erfolgreich bearbeitet werden können. So gesehen ist das Erkennen dieser Interaktionen eine zu begrüßende Fähigkeit des Agenten, notwendig ist sie jedoch nicht zwangsläufig. Ebenso wie im vorhergehenden Abschnitt über negative Interaktionen, werden auch hier nur Interaktionen zwischen Zielen eines einzelnen Agenten untersucht. Deshalb wird ebenso vereinfacht nur von den positiven Interaktionen gesprochen werden.

Positive Interaktionen können wie im vorhergehenden Abschnitt auf zwei Ebenen betrachtet werden. Zum einen können positive Interaktionen durch die Zielkerne selbst beschrieben werden, zum Anderen aber auch erst während der Bearbeitung der Ziele durch Pläne entstehen. So unterscheidet man zwischen gegenseitiger Inklusion [Wil83] und Planüberschneidung [TPW2003b].

Gegenseitige Inklusion

Bei der gegenseitigen Inklusion zwischen Zielen eines Agenten kann man zwischen folgenden Typen unterscheiden. Der triviale Typ (Identität), bei dem zwei Ziele identisch sind und somit beide durch einen Plan aus der Menge der Pläne für beide Ziele bearbeitet werden können und der implizierende Typ, bei

dem das eine Ziel einen Zustand beschreibt der in der Zustandsbeschreibung des anderen Ziels bereits enthalten ist, beschreiben diese Typen. Beide Typen sind sehr abstrakte Beschreibungen von möglichen Interaktionen auf Ebene der Ziele und sind daher hier nur als Hinweise zu verstehen. Sie werden nicht weiter konkretisiert.

Planüberschneidung

Abbildung 3.7: Positive Überschneidung von Plänen

Durch die Bearbeitung von Zielen durch Pläne können Situationen entstehen, die postitiv ausgenutzt werden können. So können z.B. folgende Fälle vorliegen.

1. **Fall:** Der Plan zur Bearbeitung eines Ziels A ist identisch mit einem Plan zur Bearbeitung eines Ziels B. Hierbei wird jedoch nicht davon ausgegangen, dass Pläne u.U. parametrisiert werden können. Somit folgt, dass zur Bearbeitung beider Ziele der Plan lediglich einmal ausgeführt werden muss.

2. **Fall:** Für diesen Fall wird angenommen, das ein Plan zur Bearbeitung eines Ziels G_1 in dem Effekt e resultiert und ein anderer Plan zu Bearbeitung des Ziels G_2 zu den Effekten e,f führt. Daraus folgt, dass für das erfolgreiche bearbeiten beider Ziele lediglich der Plan des Ziels G_2 ausgeführt werden müsste, da er sowohl die Effekte die für die Bearbeitung von G_2 wie aber auch die Effekte für G_1 bereitstellt.

3.7 Zusammenfassung

Ziele sind ein wichtiger Ansatzpunkt für die Kontrolle des Verhaltens eines Agenten. Insbesondere für ein Deliberationsverfahren auf Zielen mit ausreichender Komplexität bedarf es eines im Verhältnis zu bestehenden Zielkonzepten erweiterten Konzeptes.

In diesem Kapitel wurden daher verschieden Bereiche eines Ziels beleuchtet. Einen großen Anteil an Ausführungen dazu, hatte das Zielkonzept in Jadex. Hier wurde vor allem auf die Attribute Attitüde und dynamischer Zustand eingegangen, um deren Wichtigkeit für ein Deliberationsverfahren aufzuzeigen. Die Attitüde eines Agenten zu seinen Zielen ermöglichte ihm zusätzliche Standpunkte zu artikulieren. So konnte z.B. der Standpunkt dargestellt werden, dass ein betreffendes Ziel momentan nur als Option (Desire) gesehen wird und somit noch

nicht zur Bearbeitung ausgewählt wurde. Der dynamische Zustand hingegen beschrieb Situationen die je nach Zieltyp während einer Bearbeitung eingenommen werden konnten. Für den Zieltyp „Achieve“ bedeutete dies die Zustände „In Process“ für die Situation, dass das Ziel gerade bearbeitet wird, „Suceeded“, dass das Ziel erfolgreich bearbeitet wurde und „Failed“ eben entsprechend nicht erfolgreich. Ergänzt wurden die Ausführungen über Ziele durch Eigenschaften für Ziele die in den gängigen BDI Theorien [CL93, RG91] gefordert werden, verschiedene Aspekte der Betrachtungen von Zielen und schlussendlich dem nicht minder wichtigen Abschnitt über Interaktionen zwischen Zielen.

Das Kapitel über das Konzept Ziel sollte damit eine Grundlage schaffen, die für die Abschätzung der Verfahren im nun folgenden Kapitel notwendig ist.

Kapitel 4

Deliberation auf Zielen

In Kapitel 2 wurde der Vorgang des „Practical Reasoning“ erläutert. Dabei wurde festgestellt, dass zwei entscheidende Prozesse das „Practical Reasoning“ gestalten. Einer dieser Prozesse ist die Deliberation. Die Deliberation hat die Aufgabe aus einer Menge von Optionen, diejenige oder diejenigen auszuwählen, die der Agent sich verpflichten sollte zu erreichen. Dabei sind Optionen als die Möglichkeiten beschrieben, die zum Erreichen der durch den Agenten aufgestellten Probleme (Goals, Desires, Subgoals, Tasks) führen.

Der Begriff der Goal-Deliberation kann in diesem Kontext nun wie folgt definiert werden. Wie die Bezeichnung vermuten lässt, bezieht sie sich auf ganz bestimmte Optionen - die Ziele. Ebenso wie die Deliberation sucht sie eine Entscheidung herbeizuführen, welche der Ziele aus der Menge der dem Agenten zur Verfügung stehenden Ziele in Betracht gezogen werden sollen. Dabei können unterschiedlichste Strategien gebildet werden, die im folgenden beschrieben werden sollen.

Begonnen werden soll mit dem Deliberationsverfahren, welches bereits in Jadex implementiert ist. Dies soll einen Hintergrund schaffen, der darstellt was bereits mit Jadex möglich ist. Dieses in Jadex implementierte Verfahren wird daher nur als ein Vergleich herangezogen und logischerweise nicht als ein Verfahren das zur Auswahl steht gesehen. Im Anschluss werden Verfahren die auf sogenannten „Summary“ Informationen beruhen vorgestellt. Diese Verfahren sind eine vielversprechende Möglichkeit den Agenten über seine Ziele deliberieren zu lassen und machen einen bedeutenden Anteil dieser Arbeit aus. Danach werden regelbasierte Mechansimen besprochen. Dabei wird auf die Programmiersprache 3APL für Agenten und auf Konstruktionsregeln für konsisitente Ziele eingegangen. Darauf folgend wird ein Verfahren vorgestellt, welches auf die Verwendung einer Intensität von Zielen basiert. Die Intensitiät wird durch die Spezifikation eines zeitlichen Rahmens definiert und betrachtet somit speziell die zeitlichen Anforderungen von Zielen. Im letzten Teil dieses Kapitel werden dann Ansätze vorgestellt die auf der Verwendung eines Nutzen und einer Priorität basieren. Diese Ansätze entstammen agentenbasierenden Systemen wie JAM [Hub99], Interrap [MP93] und HAP [Loy97].

Die Einteilung der Verfahren, die in diesem Kapitel gewählt wurde, orientiert sich an dem Hauptbestandteil eines jeden Verfahrens. Dabei kann es u.U. vorkommen, dass in einem Verfahren auch Bestandteile einer anderen Einteilung auftreten. Dies ist so beabsichtigt und die auftretenden Frembestandteiel sollen deshalb den Leser nicht irritieren.

4.1 Kriterien für eine Untersuchung

Für die Bewertung der folgenden Deliberationsverfahren ist es notwendig Kriterien aufzustellen. Dabei wird folgendermassen vorgegangen. Zuerst ist von Interesse welche Information man dem Verfahren zugänglich machen muss damit es funktioniert. Dabei ist wichtig, zu Erfahren wie man die Information gewinnen kann und ob diese speziell oder eher allgemein gehalten ist. Anschliessend wird das Resultat betrachtet, welches das Verfahren mit dieser Information erreichen kann. Hierbei soll untersucht werden, welchen Nutzen das Ergebnis der Deliberation hat. Ist es möglich nur einfache Entscheidungen zu treffen oder ist eine komplexere Entscheidung möglich? Abschliessend soll entschieden werden, welche Vorteile und welche Nachteile das betreffende Verfahren mitsichbringt.

4.2 Goal Deliberation in Jadex – Easy Deliberation

„Easy Deliberation“ ist die Bezeichnung für das bereits vorhandene Deliberationsverfahren in Jadex. Grundlage für dieses Deliberationsverfahren sind der in Kapitel 2.3 beschriebene variable Interpreterzyklus und das in Kapitel 3 eingeführte erweiterte Zielkonzept. Der variable Interpreterzyklus ermöglicht dem Entwickler eines Agenten immer dann einen Deliberationsprozess zu starten, wenn er dafür die Notwendigkeit sieht. Er fügt dafür lediglich an den gewünschten Stellen Meta Aktionen in die Agenda ein. Diese Meta Aktionen sind spezifische Aktionen, die das betreffende Deliberationsverfahren umsetzen.

Für die Spezifikation der Meta Aktionen orientiert man sich am Lebenszyklus (lifecycle) eines Ziels. Im Fall der „Easy Deliberation“ Strategie wurden die Übergänge zwischen den Zuständen „Option“ und „Active“ als Ansatzpunkte für eine Deliberation festgelegt. So wurde abgeleitet, dass die „ Easy Deliberation“ Strategie die Basismenge an Meta-Aktionen um Aktionen für das Aktivieren eines Ziels im Zustand „Option“ und für das Deaktivieren eines aktiven Ziels ergänzen sollte.

Jadex verwendet zur Deliberation auf Zielen die Konzepte Kardinalität und Hemmungskante (inhibition arc). Diese Konzepte stammen aus Erfahrungen, die durch den Einsatz von Jadex in Beispielanwendungen gewonnen wurden und Ideen die den Softwareentwicklunsmethodiken wie Tropos [GMP2002] und KAOS [DvLF93], im speziellen der Modellierung von Zielen, entstammen. Beide Konzepte werden im ADF auf Typen von Zielen spezifiziert. Die Kardinalität eines Ziel beschreibt die maximale Anzahl an Instanzen die zur Laufzeit

```
<achievegoal name="AchieveCleanupWaste ">
  [omitted parameter and condition specs . for brevity]
  <deliberation cardinality="1">
    <inhibits ref="PerformLookForWaste "/>
    <inhibits ref="AchieveCleanupWaste ">
      $beliefbase.my_location.getDistance($ref.waste.location) >
      beliefbase.my_location.getDistance($goal.waste.location)
    </inhibits>
  </deliberation>
</achievegoal>
```

Abbildung 4.1: Spezifizierung der Deliberation am Beispiel, aus [PBL05a]

gleichzeitig aktiv sein dürfen. Eine Hemmungskante beschreibt eine negative Beziehung zwischen Zieltypen und deren Instanzen. Ein Ziel welches durch ein anderes Ziel gehemmt wird, darf nicht gleichzeitigt mit dem hemmenden Ziel ausgeführt werden. Zur Vermeidung einer Schleifenbildung bei der Spezifizierung von Hemungskanten, müssen diese einen azyklischen gerichteten Graphen bilden. Für eine Verfeinerung der Spezifizierungsmöglichkeiten der Hemmungskanten, kann für Instanzen eines Zieltyps durch die Angabe einer Bedingung in der „ embedded reasoning language“ diejenigen Instanzen ausgewählt werden für die diese spezifizierte Bedingung zutrifft. In Abbildung 4.1 ist ein Ausschnitt aus einem ADF dargestellt, indem ersichtlich wird, wie die Deliberationskonzepte der „Easy Deliberation“ Strategie verwendet werden können.

4.2.1 Spezifizierung

Angaben zur Deliberation werden innerhalb von `<deliberation>` gemacht. Dazu spezifiziert man dort das Tag `<inhibits>` und gibt durch das `ref` Attribut an welches Ziel gehemmt werden soll. Optional lässt sich mit dem Attribut `inhibit` der Zustand des Ziels festlegen, in dem die Hemmungskante aktiv werden soll. So ist z.B. denkbar, dass für Ziele vom Typ Maintain Hemmungskanten nur aktiv sein sollen, wenn das Ziel im Bearbeitungszustand „in process“ vorliegt. Ausdrückbar ist dieser Sachverhalt durch `inhibit="when in process"`. Als Standardwert gilt, dass ein Ziel ein anderes hemmt, wenn es sich im Zustand „Active“ befindet.

Der Ausdruck zwischen dem öffnenden und dem schließenden zweiten `<inhibits>` Tag ist in der „embedded reasoning language“ spezifiziert. Er ermöglicht die bereits oben beschriebene Verfeinerung für Instanzen des Zieltyps *AchieveCleanupWaste* festzulegen. Dadurch lässt sich die Hemmung von Instanzen des Ziels *AchieveCleanupWaste* auf diejenigen beschränken, die einen weiteren Weg zu ihrer Aufgabe haben als das betreffende Ziel selbst.

Mit der Angabe des Attributs `cardinality` im Tag `deliberation`, lässt sich das Konzept der Kardinalität beschreiben. In dem hier vorliegenden Beispiel,

kann höchstens eine Instanz des Zieltyps AchieveCleanupWaste aktiv sein.

4.2.2 Realisierung

Die „Easy Deliberation“ Strategie setzt an neuralgischen Punkten des Modells eines Ziels an. In Abbildung 4.2 ist das Modell des Ziel mit diesen Punkten dargestellt. Zwei entscheidende Situationen sind dabei auszumachen, die die Aktivierung der Strategie bedürfen.

- Ein Ziel kann sich in dem Zustand „Option“ befinden und soll zur weiteren Bearbeitung in den Zustand „Active“ überführt werden.
- Ein Ziel kann durch das Eintreten bestimmter Situationen in die Zustände „Suspended“ oder „Finished“ bzw. „Dropped“ übergehen.

Für die Aktivierung eines Ziels, bestimmt die Strategie in der Meta-Aktion `Deliberate goal activation` ob die Kardinalität des Zieltyps nicht verletzt wird und ob das Ziel nicht durch ein aktives Ziel in seiner Aktivierung gehemmt wird. Sind beide Überprüfungen positiv aus Sicht des zu aktivierenden Ziels verlaufen, wird das Ziel aktiviert.

An die Ausführung der Meta-Aktion `Deliberate goal activation` schließt sich jetzt die Ausführung der Meta-Aktion `Deactivate inhibited goals` an. Diese Meta-Aktion ist dafür vorgesehen alle aktiven Ziele die durch das gerade aktivierte Ziel gehemmt werden zu deaktivieren. Betreffende Ziele werden in den Zustand „option“ zurückversetzt. Ändert ein Ziel seinen Zustand von „Active“ nach „Suspended“ oder von „Active“ nach „Dropped“, so muss die „Easy Deliberation“ Strategie feststellen, welche Ziele mit dem Zustand „Option“ einer Deliberation über ihre Aktivierung unterzogen werden können. In Frage kommende Ziele sind all diejenigen, die Instanzen vom gleich Typ wie das betreffende Ziel sind. Denn es ist möglich, das Ziele aufgrund einer negativ ausgefallenen Prüfung der Kardinalität nicht aktiviert werden konnten. In Frage kommen aber auch alle Ziele, die durch das betreffende Ziel gehemmt wurden. Da das hemmende Ziel jetzt nicht mehr aktiv ist, haben diese Ziele eine berechtigte Chance auf eine Aktivierung.

Für jedes dieser Ziele wird die Meta-Aktion `Deliberate goal activation` in die Agenda geschrieben. Es tritt jetzt wieder die erste beschrieben Situation ein.

4.2.3 Beurteilung der „Easy Deliberation“ Strategie

Die „Easy Deliberation“ Strategie stellt ein einfaches und schnelles Verfahren zur Deliberation auf Zielen dar. Insbesondere die Verwendung der Hemmungskanten ermöglicht dabei, durch die Spezifikation einer Beziehungsstruktur zwischen Zielen, Konflikte und Prioritäten abstrakt darzustellen, ohne diese konkret ausformulieren zu müssen. Dieses reduziert den Aufwand einer Spezifikation erheblich. So wäre in dem Fall der expliziten Spezifikation von Prioritäten, die Festlegung einer globalen Ordnung zwischen Zielen notwendig. In einer globalen

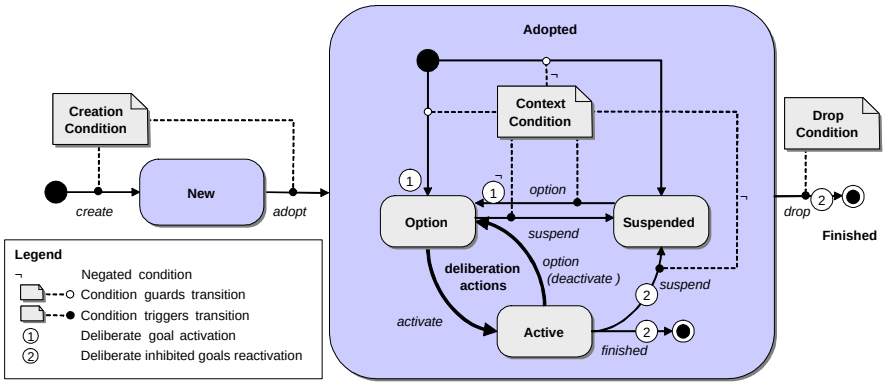

Abbildung 4.2: Ansatzpunkte der „Easy Deliberation“ Strategie, aus [PBL05b]

Ordnung müssen alle Ziele untereinander berücksichtigt werden, was bedeutet, dass man Prioritäten nicht unabhängig voneinander spezifizieren kann. Speziell wenn das Konzept der Capability[1] verwendet wird, können so Schwierigkeiten entstehen. Da Capabilities nicht an einen speziellen Agenten gebunden sind und somit nicht feststeht in welchem Umfeld von Zielen sich die Capability möglicherweise bewegt, ist im Fall der Notwendigkeit der Deliberation auf Ziele einer Capbility nicht einfach festlegbar, wie diese Ziele geordnet werden sollten.

Das Konzept der Hemmungskanten hat aber auch einige Einschränkungen zu verzeichnen. Wie in [PBL05a] erwähnt wird, können Hemmungskanten wie sie in der „Easy Deliberation“ Strategie verwendet werden nur bilaterale Beziehungen zwischen Zielen darstellen. Das bedeutet, dass jeweils nur eine Beziehung zwischen zwei Zielen möglich ist. „N:1“ Beziehungen sind mit diesem Konzept nicht darstellbar. So kann man z.B. nicht ausdrücken, dass zwei oder mehrere Ziele die in Konflikt mit einem anderen Ziel stehen, zusammen wichtiger als dieses Ziel sind. Ebenso berücksichtigt die „Easy Deliberation“ Strategie keine Interaktionen zwischen Plänen und bietet demnach keine Spezifikationsmöglichkeiten dafür. So können wichtige Interaktion auf der Ebene der Pläne, wie z.B. Ressourcenkonflikte, entweder gar nicht oder nur unzureichend abstrakt auf der Ebene der Ziele spezifiziert werden. Dadurch geht mögliches zusätzliches Entscheidungspotential verloren.

Zusammenfassend kann gesagt werden, dass das einfache Konzept der Hemmungskanten relativ ausdrucksstark ist. Insbesondere wenn man die Entwicklung eines Agentensystems aus Richtung des „Requirements Engineering“ betrachtet, hilft der nahtlose Übergang von Systemzielen, des „Requirements Engineering“, zu Zielen eines Agenten, den Entwurfsprozess zu verbessern.

[1] Capabilities sind wie der Name schon sagt Fähigkeiten eines Agenten. Diese sogenannten Fahigkeiten entsprechen Modulen und sind Zusammenfassungen von Beliefs, Zielen und Plänen eines Agenten. Durch dieses modulartige Konzept können somit häufig benötigte Fähigkeiten, bei der Entwicklung weiterer Agenten wiederverwendet werden. In [PL2000] findet sich näherers zu dem Konzept der Capability.

4.3 Summary Information

Summary Informationen, wie sie im Folgenden in verschiedenartiger Ausgestaltung auftreten werden, sind ursprünglich von Clement und Durfee [CD99] für die Koordination von Plänen auf einer abstrakten Ebene verwendet worden. Mit Hilfe dieser Informationen konnte z.B. ein Algorithmus den Ablauf der nebenläufigen Ausführung von Plänen steuern. Im Folgenden werden diese „Summary" Informationen dazu verwendet, um verschiedene Interaktionen zwischen Zielen eines Agenten zu erkennen und in die Abarbeitung dieser Ziele durch eine geeignete Ablaufsteuerung einzugreifen.

4.3.1 Interferenzen – Interaction Summary

Bei der Bearbeitung eines Ziels können verschiedene Pläne benutzt werden. Diese Pläne beschreiben Schritte, die zur erfolgreichen Bearbeitung des betreffenden Ziels auszuführen sind. Schritte können atomare Aktionen, aber auch Subziele darstellen, wobei Subziele wiederum durch Pläne bearbeitet werden.

Bei der Bearbeitung eines Ziels durch einen Plan, der Subziele spezifiziert, kann es vorkommen, dass bestimmte Pläne von diesen Subzielen Zustände erzeugen, die notwendig für die Bearbeitung der anderen Subziele sind. Solche Zustände bezeichnet man als Vorbedingungen (preconditions) oder als Kontextbedingung (incondition), je nachdem ob diese Zustände notwendig für Anwendbarkeit (Vorbedingung) oder Ausführbarkeit (Kontextbedingung) sind.

Ein rationaler Agent wird diese Zustände solange sichern, bis der auf diese Zustände aufbauende Plan bearbeitet worden ist. Er wird es nicht zulassen, dass die erreichten Zustände durch Effekte anderer Ziele bzw. Pläne zunichte gemacht werden.

Zustände die durch Pläne erzeugt werden und die notwendig für die Ausführung eines anderen Plans sind, werden in [TPW2003] als „Preparatory" Effekte (P-Effekte) bezeichnet. Abhängigkeiten zwischen dieser Art von Effekten und Plänen bzw. Zielen werden als „Dependency-link" tituliert. Zur Überwachung von möglich auftretenden Interferenzen werden sogenannten „Interaction-Summaries" eingesetzt. Näheres zu diesen Begriffen wird im Folgendenen erläutert. Vorab ist es aber notwendig, genauer auf die Repräsentation von Plänen und Zielen einzugehen. Denn für dieses und die folgenden Verfahren in diesem Kapitel werden bestimmte Anforderungen gestellt.

Repräsentation von Zielen und Plänen

Dem hier beschriebenen Verfahren zur Erkennung von Interferenzen, liegt eine gewisse Repräsentation von Zielen und Plänen zugrunde. Ein Ziel besitzt demnach einen Bezeichner, eine Angabe der direkten Effekte, eine Angabe der Kontextbedingung und eine Menge an möglichen Plänen die zur Erfüllung dieses Ziels beitragen können. Zusätzlich dazu besitzt ein jedes Ziel einen sogenannten „Goal-Plan Tree", das ist eine Baumstruktur die die Beziehungen zwischen dem Ziel und seinen Plänen wiedergibt. Dabei bilden das Ziel und seine Pläne eine

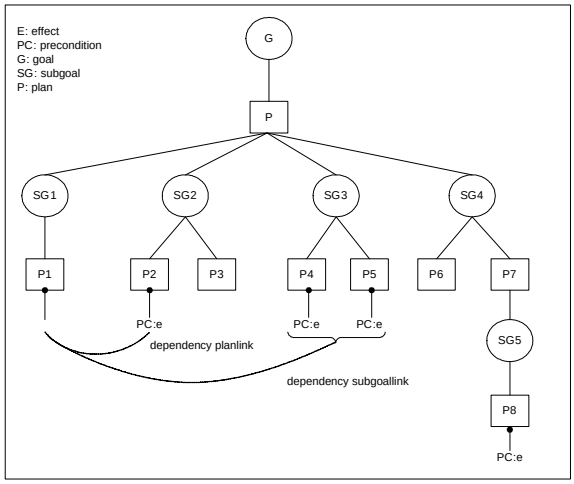

Abbildung 4.3: Goal-plan tree aus [TPW2003]

OR-Verfeinerung[2] (siehe 3.5) und Subziele eines Plans eine AND-Verfeinerung für das zu bearbeitende Ziel des Plans. Ein „Goal-plan Tree“ ist in Abbildung 4.3 dargestellt.

Ein Plan besitzt ebenso wie ein Ziel einen Bezeichner, eine Angabe der direkten Effekte und Kontextbedingungen. Zusätzlich zu den Anforderungen an ein Ziel besitzt ein Plan hier noch Angaben über die Vorbedingungen die für eine Ausführung des Plans gelten müssen und einen Rumpf (body), der spezifiziert welche Aktionen der Plan ausführt.

Dependency-Link

Abbildung 4.3 zeigt einen „Goal-plan Tree“. Dort ist zu erkennen, das Plan P_1 den Effekt e erzeugt. Dieser Effekt wird vom Plan P_2 benötigt, damit dieser anwendbar ist. Der Effekt e ist ein P-Effekt und der Plan P_2 der abhängige Plan. Die Abhängigkeit zwischen e und P_2 bezeichnet man als „Dependency-link“. In diesem speziellen Fall spricht man von einem „Dependency-planlink“. Im Fall der Abhängigkeit zwischen e und den Plänen P_4, P_5 des Subziels SG_3 spricht man von einem „Dependency-subgoallink“, da das gesamte Subziel von der Vorbedingung abhängt.

„Dependency-Links“ sind weiterhin in potentielle und definitive unterteilbar. Bei potentiellen „Dependency-Links“ handelt es sich um eine Abhängigkeit, die nicht zwangsläufig notwendig ist. So ist z.B. der „Dependency-Link“ zwischen P-Effekt e und dem Plan P_2 potentiell. Es besteht nämlich noch die Möglichkeit

[2] Eigentlich müsste genau gesagt von einer OR-Verfeinerung zwischen den Subzielen der jeweiligen für das betreffende Ziel vorhandenen Pläne gesprochen werden. Da aber Pläne indirekt für diese Verfeinerung durch Subziele stehen wird hier nur von einer OR-Verfeinerung zwischen dem Ziel und seinen Plänen gesprochen.

das Subziel durch den Plan P_3 zu bearbeiten. Generell sind alle „Dependency-planlinks“ potentiell.

„Dependency-subgoallinks“ können potentiell oder definitiv sein. Potentielle „Dependency-subgoallinks“ beschreiben eine Abhängigkeit zwischen einem Effekt und einem Subziel, welche nicht notwendigerweise zur Bearbeitung des betreffenden Urpsprungsziels notwendig sind. Dies ist der Fall zwischen dem P-Effekt e und dem Subziel SG_5 in Abbildung 4.3. Definitive „Dependency-subgoallinks“ hingegen sind in jedem Fall notwendig um eine erfolgreiche Bearbeitung des Ursprungsziels zu gewährleisten.

Durch die Spezifizierung von potentiellen und definitiven „Dependency Links“ ist es möglich eine Ordnung auf diesen Klassen von „Dependency Links“ zu definieren. Diese Ordnung gibt an, welche „Dependency Links“ Priorität in ihrer Sicherung haben. Die wichtigsten „Dependency Links“ sind die definitiven „Dependency Subgoallinks“. Anschliessend folgend die potentiellen „Dependency Subgoallinks“ und die „Dependency Planlinks“.

„Dependency Links“ werden in „Dependency Entries“ zusammengefasst. Dependency-Entries sind Einträge die ausschließlich an Planknoten eines „Goal-plan Tree“ befestigt werden.

Interaction Summary

„Interaction-summaries“ dienen der Erkennung, ob die Effekte eines Ziels mit den Kontextbedingungen und „Dependency-links“ eines anderen Ziels interferieren. Dazu umfasst die „Interaction-summary“ die Effekte ($S_e(n)$) , Kontextbedingungen ($S_i(n)$) und P-Effekte($S_{pe}(n)$) eines Ziels und ist folgendermaßen definiert:

$$I_{summary}(n) = \langle S_i(n), S_e(n), S_{pe}(n) \rangle.$$

Im Gegensatz zu den „Dependency Links“, ist die „Interaction-summary“ für einen jeden Knoten n eines „Goal-plan Tree“ definiert. Einen „Goal-plan Tree“ der solche Informationen besitzt, bezeichnet man dann als „Interaction Tree“. Solch ein „Interaction Tree“ ist in Abbildung 4.4 dargestellt.

Beispielhaft für den Aufbau der in der „Interaction Summary“ enthaltenen Informationen wird die „In-Conditions Summary“ betrachtet. Die „Effect-Summaries“ und die „P-Effekt Summaries“ sind entsprechend aufgebaut.

Die „In-condition Summaries“, $S_i = \langle D_i, P_i \rangle$, sind ein 2 Tupel bestehend aus definitiven Bedingungen (D) und aus potentiellen Bedingungen (P) die der Knoten i benötigt. Definitive wie auch potentielle Bedingungen sind dabei Mengen von Kontextbedingungen und stellen zum einen Bedingungen dar, die definitiv benötigt werden und zum anderen Bedingungen die möglicherweise benötigt werden. Definitive Bedingungen sind Bedingungen die auf jedem möglichen Pfad zur Erfüllung des betreffenden Ziels erreicht (Effekte) bzw. benötigt (Kontextbedingungen) werden. Potentielle Bedingungen sind dagegen Bedingungen die auf mindestens einem Pfad, aber nicht allen Pfaden zur Bearbeitung eines Ziels benötigt bzw. erreicht werden.

Berechnung der Interaction Summary eine Knoten

Die Berechnung einer „Interaction-Summary“ eines Knoten des „Goal-plan Tree“ läuft folgendermaßen ab. Die „Interaction Summary“, also die Informationen der Effekte, P-Effekte und der Kontextbedingungen eines Knoten, leitet sich aus der Kombination der lokalen „Interaction Summary“ mit den „Interaction-Summaries“ der Folgeknoten (Kinder) ab. Dabei gibt es in der Art der Ableitung Unterschiede zwischen Zielknoten und Planknoten. Zunächst wird die Berechnung im Fall von Planknoten betrachtet.

Die Effekte (effects) eines Plans berechnen sich aus den lokalen Effekten des Plans selbst und aus den Effekten der Subziele des Plans. Es gilt für die „Effect Summary“ eines Plans:

$$S_e(P) = \langle \text{effects-of}(P), \rangle \oplus \bigoplus_{g \in G(P)} S_e(g)$$

Die Teilmengen D und P werden dabei jeweils durch den Vereinigungsoperator kombiniert. Es gilt:

$$(D_1, P_1) \oplus (D_2, P_2) = ((D_1 \cup D_2), (P_1 \cup P_2))$$

Die Herleitung der „P-Effekt Summary“ und der „Incondition Summary“ verläuft analog zu der Herleitung der „Effekt Summary“.

Die Effekte eines Ziels hingegen berechnen sich nur aus den Effekten der für dieses Ziel vorhandenen Pläne. Es gilt somit für die „Effect Summary“ eines Ziels:

$$S_e(G) = \bigotimes_{p \in plansOf(G)} S_e(p)$$

Bei der Berechnung der Effekte für die Teilmengen D und P muss beachtet werden, dass Pläne für ein Ziel nur Alternativen also OR-Verfeinerungen darstellen. Sie können also nicht uneingeschränkt, wie im Fall eines Plans, einfach vereinigt werden. Das bedeutet, dass die definitiven Bedingungen eines Ziels definitiv in allen Plänen sein müssen. Potentielle Bedingungen hingegen sind nur in einigen Plänen vorhanden, nicht aber in allen. Sie setzen sich aus definitiven Bedingungen zusammen, die nicht definitiv in allen Plänen sind und aus den potentiellen Bedingungen der Pläne. Es gilt:

$$(D_1, P_1) \otimes (D_2, P_2) = \langle (D_1 \cap D_2), ((D_1 \cup D_2 - D_1 \cap D_2) \cup P_1 \cup P_2) \rangle$$

Die Herleitung der P-Effekt Summary verläuft analog zur Herleitung der „Effekt Summary“. Die Berechnung der „In-conditions Summary“ hingegen weicht davon ab. Da ein Ziel eigene Kontextbedingungen besitzen kann, fließen diese mit in den Berechnungsprozess ein. Die „In-conditions Summary“ setzt sich also aus den lokalen Kontextbedingungen des Ziels und aus den Kontextbedingungen der für dieses Ziel vorhandenen Pläne zusammen. Für die Kombination der Kontexbedingungen der Pläne gelten die Aussagen die für die „Effect Summary“ getroffen wurden. Die Kombination der lokalen Kontextbedingungen mit

den Kontextbedingungen der Pläne geschieht, wie im Fall der Bestimmung der Effekte von Subzielen eines Plans. Es gilt demnach:

$$S_i(P) = \langle inCond(G), \rangle \oplus \bigotimes_{g \in G(P)} S_i(g)$$

Um die „Interaction-summary" einer Menge an Zielen zu spezifizieren berechnet man wie folgt:

$$\begin{aligned} IS(GoalSet) &= \uplus_{g \in GoalSet} IS(g) \text{ mit} \\ IS(g_1) \uplus IS(g_2) &= \langle S_i(g_1 \oplus S_i(g_2), S_e(g_1) \oplus S_e(g_2), S_{pe}(g1) \oplus S_{pe}(g_2) \rangle \end{aligned}$$

Ablaufsteuerung für die parallele Bearbeitung von Zielen

Zunächst stellt man für ein neues Ziel G_{new} fest, ob dieses Ziel kompatibel zu der vorhandenen Gesamtmenge an Zielen ist. Ein neues Ziel ist dann kompatibel zur vorhandenen Menge an Zielen, wenn die Effekte des neuen Ziels kompatibel mit den Kontextbedingungen und den P-Effekten der Menge von Zielen sind. Dabei bedeutet die Kompatibilität zweier Bedingungen, dass sie gleichzeitig wahr sein können.

Wenn dies nicht der Fall ist, und somit Interferenzen zwischen den Zielen zu erwarten sind, muss eine geeignete Ablaufsteuerung implementiert werden. Für den Ablauf der Abarbeitung von Zielen ist es von Wichtigkeit die Kontextbedingungen und die „Dependency Links" gegen Interferenzen zu sichern. Dies geschieht durch die Einführung einer weiteren Datenstruktur, dem sogenannten *Guarded Set*. Das *Guarded Set* beinhaltet alle aktuell zu schützenden Bedingungen. Dies sind Kontextbedingungen von aktuell bearbeiteten Zielen und Plänen und „Dependency-links" von aktuell bearbeiteten Plänen. Zu beachten ist weiterhin die Möglichkeit des Auftretens von Deadlocks. So wird vor der Ausführung eines Planes oder Ziels überprüft, ob die Effekte die durch vorhandene Ziele erreicht werden, nicht mit den Kontextbedingungen der neu zu bearbeitenden Ziele oder Pläne interferieren.

Um den Ablauf der parallelen Bearbeitung von Zielen hinsichtlich auftretender Interferenzen zu überwachen, sollten laut [TPW2003] folgende Schritte in den Bearbeitungszyklus eines intelligenten Agenten aufgenommen werden.
Für Ziele gilt:

1. Ab dem Beginn der Bearbeitung eines Ziels wird dessen Kontextbedingung dem *Guarded Set* hinzugefügt.

2. Ist das Ziel erfolgreich bearbeitet worden oder fehlgeschlagen, so wird die Kontextbedingung des betreffenden Ziels aus dem *Guarded Set* entfernt. Sollte das betreffende Ziel Teil eines „Dependency-link" sein, so wird auch dieser entfernt.

3. Steht ein neues Ziel bereit ausgeführt zu werden, so wird vor dem Beginn einer Ausführung geprüft, ob die Kontextbedingung des neuen Ziels kompatibel zu den definitiven und potentiellen Effekten der im Guarded Set

aufgeführten Ziele sind. Sind sie kompatibel, so kann das neue Ziel parallel zu den bereits vorhandenen Zielen ausgeführt werden. Sollte aber das neue Ziel nicht kompatibel sein, so wird zunächst geprüft ob es kompatibel zu den definiten Effekten der bestehenden Ziele ist. Ist das Ziel nicht kompatibel zu den potentiellen Effekten der aktiven Ziele, so kann der Agent eine zweigleisige Strategie fahren. Erstens kann der Agent die Ausführung des neuen Ziels verhindern, solange die inkompatiblen Bedingungen noch nicht erreicht sind und anschließend die Ausführung des neuen Ziels beginnen. Damit befindet er sich auf der sicheren Seite. Oder aber zweitens, er beginnt die Ausführung des Ziels unmittelbar und hofft darauf, dass es sich bei den inkompatiblen Bedingungen um potentielle Bedingungen handelt, dass solche Pläne ausgewählt werden in denen nicht die inkompatiblen Bedingungen auftreten.

Für Pläne gilt:

1. Wird ein Plan begonnen auszuführen, so werden die Kontextbedingungen und in dem Fall in dem der betreffende Plan P-Effekte besitzt auch die für den P-Effekt relevanten „Dependency-entries“ in das *Guarded Set* übernommen.

2. Wird ein Plan erfolgreich beendet oder schlägt ein Plan fehl, so werden die relevanten Kontextbedingungen und „Dependency-Entries“ aus dem *Guarded Set* entfernt.

3. Vor der Ausführung eines neuen Plans, werden folgende Dinge geprüft. Sind die Effekte der neuen Pläne kompatibel mit den gesicherten Bedingungen im *Guarded Set*? Wenn nicht muss mit der Ausführung des Planes solange gewartet werden, bis die Kompatibilität eintritt. Um mögliche „Deadlocks“ zu vermeiden muss ebenso geprüft werden, ob die abgeleiteten Effekte der Ziele die einen Eintrag im *Guarded Set* haben mit den Kontextbedingungen und Effekten des jeweiligen Plans kompatibel sind.

4.3.2 Ressourcenkonflikte – Resource Summary

Ein rationaler intelligenter Agent sollte sich bewusst sein, dass er keine Ziele verfolgt die sich gegenseitig beeinflussen. Eine allgemeine Art der Beeinflussung wurde im vorhergehendem Kapitel über Interferenzen beschrieben. Eine speziellere Art der Beeinflussung kann sich in der Verwendung von Ressourcen manifestieren. Ressourcen sind Gegenstände der Umwelt, die der Agent benötigt um seine Ziele zu erreichen. Grob betrachtet existieren zwei Arten von Ressourcen. Zum einen sind das solche Ressourcen, die verbrauchbar (consumable) und somit nur in einem bestimmten quantitativen Umfang vorhanden sind und zum anderen Ressourcen die wiederverwendbar (reuseable) sind, also lediglich in einem bestimmten zeitlichen Intervall nicht zur Verfügung stehen. Da hier, wie im vorigen Kapitel, davon ausgegangen wird, dass Ziele eines Agenten durch die Ausführung von Plänen erreicht werden und es potentiell mehrere Pläne

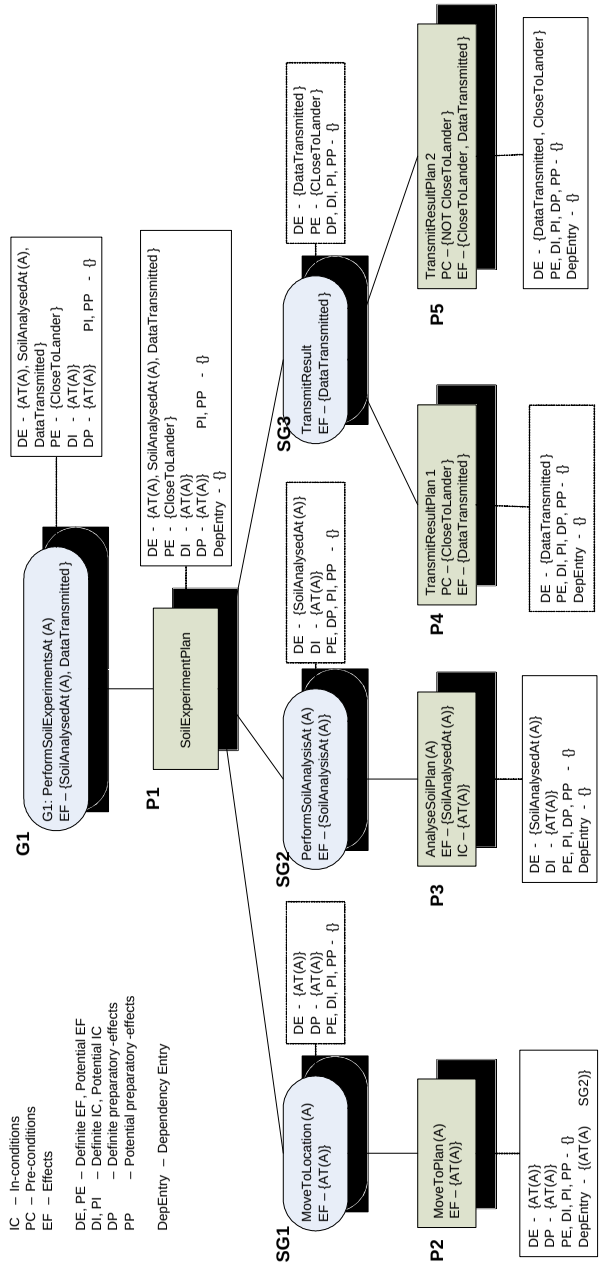

Abbildung 4.4: Interaction Tree aus [TPW2003]

zur Ausführung eines Ziels geben kann, unterscheiden wir weiterhin folgende Ressourcen.

Ressourcen die bei der Ausführung aller möglichen Pläne benötigt werden bezeichnet man, ähnlich wie definitiven Bedingungen im vorigen Abschnitt, als notwendige (necessary) Ressourcen. Ressourcen die hingegeben nur in bestimmten Plänen benötigt werden, werden als mögliche (possible) Ressourcen bezeichnet. Betrachtet man zusätzlich den Bedarf eines Plans an einer Ressource, so geben notwendige Ressourcen ein Minimum an und potentielle Ressourcen ein Maximum. Zu beachten ist jedoch, dass notwendige Resourcen nicht gleich den Ausreichenden sind.

In [TWPF2002b] wird ein Mechanismus angegeben, der genau solche Beeinflussung durch Ressourcen überwachen soll. Er wird im Folgenden näher erläutert.

Spezifizierung von Ressourcen

Zunächst wird, um geeignete Schlussfolgerungen über Ressourcen anstellen zu können, der Begriff der Ressource näher spezifiziert. Ressourcen werden unterteilt in eine Menge von Ressourcentypen $\mathcal{T} = \{t_1, \cdots, t_n\}$. Für jeden Ressourcentyp $t_i, i \in 1, \ldots, n$ wird weiterhin ein Bedarf spezifiziert. Ein Ressourcenbedarf ist definiert als 2 Tupel (t, n) mit $t \in \mathcal{T}$ und $n \geq 0$. So bedeutet z.B. das Tupel $(energy, 20)$, dass von dem Ressourcentyp energy, 20 Einheiten benötigt werden. Die Tupel des Ressourcenbedarf werden in einer normierten Menge $\mathcal{R}$ von Ressourcen zusammengefasst.

Die Menge der wiederverwendbaren Ressourcen wird mit R^r und die Menge der Ressourcen die verbraucht werden können mit R^c bezeichnet. Es gilt: $R = R^r \cup R^c$ und $R^r \cap R^c = \emptyset$. Auf Mengen von Ressourcen kann eine partielle Ordnung $\sqsubseteq$ definiert werden. So gilt:

$$R_1 \sqsubseteq R_2 \text{ iff } \forall t. R_1(t) \leq R_2(t)$$

Das bedeutet, dass Menge R_1 kleiner ist als Menge R_2 wenn jeder Ressourcentyp der in R_1 ist auch in R_2 ist und dabei der Wert dieses Ressourcentyps kleiner ist als der entsprechende Wert des Ressourcentyp aus R_2.

Die Mengen der notwendigen und die der möglichen Ressourcen werden jeweils mit N bzw. P bezeichnet. Zusammengefasst in einem 2 Tupel bilden sie die „Resource Summary“. Formal ist eine „Resource-Summary also: $S = \langle N, P \rangle$. Zu beachten gilt es, dass eine notwendige Ressource auch zugleich eine mögliche Ressource ist und damit die Menge N eine untere Schranke und die Menge P eine obere Schranke an Ressourcen angibt.

Berechnung der Resource Summary

Im Folgenden wird angegeben wie die „Resource Summary“ für Pläne und Ziele berechnet wird. Für Ziele werden die „Resource Summaries“ der für dieses Ziel relevanten Pläne kombiniert und für Pläne werden die „Resource Summaries“ der für diese Pläne bestehenden Subziele mit der „Resource Summary“ der Aktionen

eines jeweiligen Planes kombiniert. Dabei werden nicht für einzelne Aktionen eines Planes die „Resource Summaries“ angegeben, sondern es wird durch die Angabe einer „Resource Summary“ für die Aktionen eines Planes festgelegt, welche Ressourcen die Aktionen eines Plans benötigen. P^R bezeichnet dabei die Ressourcen die durch die Ausführung von Aktionen eines Plans P benötigt werden.

Für die Bestimmung der „Resource Summaries von Zielen und Plänen benötigt man in diesem Verfahren verschiedene Operatoren auf den Ressourcenmengen. Diese werden jetzt vorgestellt.

Es wird der Operator $\sqcup$ definiert, der das Maximum zweier Ressourcenmengen, speziell von Ressourcenmengen aus R^r berechnet. Es gilt:

$$R_1 \sqcup R_2 = \{(t, max(R_1(t), R_2(t)))|t \in \mathcal{T}\}$$

Desweiteren wird der Operator $\sqcap$ definiert. Dieser berechnet das Minimum zweier Ressourcenmengen und es gilt für diesen Operator formal:

$$R_1 \sqcap R_2 = \{(t, min(R_1(t), R_2(t)))|t \in \mathcal{T}\}$$

Bei der Bestimmung der „Resource Summary“ eines Planes muss die Abfolge der Subziele berücksichtigt werden. Werden die Subziele eines Planes parallel ausgeführt, so werden die Ressourcenmengen bei verbrauchbaren Ressourcen wie auch bei wiederverwendbaren Ressourcen einfach addiert. Bei der sequentiellen Abarbeitung hingegen wird die Kombination der wiederverwendbaren Ressourcen durch den Operator $\sqcup$ bestimmt. Für die verbrauchbaren Ressourcen die sequentiell verarbeitet werden ändert sich nichts. Sie werden wie gewohnt addiert. Für die Unterscheidung dieser beiden Fällen werden die Operatoren $\oplus$ und $\otimes$ verwendet. Formal definiert bedeuten sie:

$$\begin{aligned} R_1 \oplus R_2 &= \{(t, R_1(t) + R_2(t))|t \in \mathcal{T}\} \\ R_1 \otimes R_2 &= (R_1^c \oplus R_2^c) \cup (R_1^r \sqcup R_2^r) \end{aligned}$$

Für die Berechnung der „Resource Summary“ eines Ziels muss man folgendes beachten. Ziele besitzen Pläne zu ihrer Bearbeitung. Diese Pläne werden nicht alle ausgeführt, sondern im besten Fall nur einer dieser. Es kann aber auch passieren das bei der Ausführung eines solchen Plans ein Fehler passiert und dieser Plan somit fehlschlägt. Die bei dieser Bearbeitung des betreffenden Plans benutzten Ressourcen sind im Fall von verbrauchbaren Ressourcen verloren. Diese Besonderheit muss bei der Bestimmung der potentiellen Ressourcen bedacht werden. Bei der Bestimmung des Maximum, also der potentiellen Ressourcen eines Ziels muss also berücksichtigt werden, dass sich die potentiellen Ressourcen der einzelnen Pläne addieren können, da sie sequentiell ausgeführt werden können. Für die Berechnung der notwendigen Ressourcen, also des Minimums an Ressourcen, verfährt man wie bisher gehabt durch die Anwendung des Operators für die Berechnung eines Minimum zwischen zwei Ressourcen. Für die Berechnung der „Resource Summary“ eines Ziel gilt demnach:

$$\langle N_1, P_1 \rangle \uplus \langle N_2, P_2 \rangle = \langle N_1 \sqcap N_2, P_1 \otimes P_2 \rangle$$

Jetzt ist es möglich die komplette Berechnung der resource summary eines Ziels formal zu spezifizieren. Dabei bezeichnet $S(G)$ und $S(P)$ die resource summary eines Ziels bzw. Plans

$$\begin{aligned} S(P) &= \langle (P^R, P^R \rangle \oplus S(body(P)) \\ S(A) &= \langle \emptyset, \emptyset \rangle \\ S(G) &= \biguplus_{p \in P(G)} S(p) \\ S(SG_1 \| SG_2) &= S(SG_1) \oplus S(SG_2) \\ S(SG_1; SG_2) &= S(SG_1) \otimes S(SG_2) \end{aligned}$$

Ressourcen-Konflikte

Diese im vorigen Unterkapitel besprochenen „Resource Summaries“ können jetzt dazu benutzt werden, um ausgehend von einer Menge an Zielen $G = \{G_1, \ldots, G_n\}$ mit $S_{G_i} = \langle N_i, P_i \rangle$ festzustellen, ob diese in Anbetracht der vorhanden Menge an Ressourcen, nebenläufig zueinander ausgeführt werden können. Dazu wird eine Klassifizierung vorgenommen. Die Menge G wird in folgende Klassen eingeteilt:

1. sequentielle obere Schranke $\bigotimes P_i$
2. parallele obere Schranke $\bigoplus P_i$
3. sequentielle untere Schranke $\bigotimes N_i$
4. parallele untere Schranke $\bigoplus N_i$

Diese Klassen stehen in folgenden Verhältnissen zueinander. $\bigotimes N_i \sqsubseteq \bigoplus N_i \sqsubseteq \bigoplus P_i$ und $\bigotimes N_i \sqsubseteq \bigotimes P_i \sqsubseteq \bigoplus P_i$. Daraus ergeben sich drei interessante Fälle.

$R \sqsubset \bigotimes N_i$: In diesem Fall sind die vorhandenen Ressourcen R kleiner als die sequentielle untere Schranke. Das bedeutet, dass für die sequentielle Bearbeitung aller Ziele nicht genügend Ressourcen zur Verfügung stehen und deshalb einige Ziele nicht bearbeitet werden können.

$\bigoplus P_i \sqsubseteq R$: Hier sind ausreichend Ressourcen für die parallele Bearbeitung aller Ziele vorhanden. Sogar in dem Fall, dass einige Pläne die zur Bearbeitung ausgewählt wurden fehlschlagen, kann die parallele Bearbeitung fortgesetzt werden.

$\bigotimes N_i \sqsubseteq \mathcal{R} \sqsubset \bigoplus P_i$: Dieser Fall stellt eine Unsicherheit im Umgang mit den auszuführenden Zielen bezüglich ihres Ressourcenbedarfs dar. Er lässt sich weiter in zwei Unterfälle gliedern.

$\bigotimes P_i \sqsubseteq R \sqsubset \bigoplus P_i$:
In diesem Fall müssen unter Umständen Ziele sequentiell ausgeführt werden. Dies kommt ganz darauf an, welche Pläne zur Bearbeitung der Ziele ausgewählt wurden. Nicht jede Kombination von Plänen und somit der

Bedarf an Ressourcen ist verträglich mit den vorhandenen Ressourcen. Generell kann jedoch jedes Ziel bearbeitet werden.

$\bigotimes N_i \sqsubseteq R \sqsubset \bigoplus N_i$:
Hier ist es notwendig Ressourcen aus R^r wiederzuverwenden, sonst droht in allen Fällen eine nicht erfolgreiche Bearbeitung aller Ziele.

Für den weiteren Verlauf, speziell zur Bestimmung der Beziehung zwischen der Menge an Zielen und den zur Verfügung stehenden Ressourcen, wird ein Status einer Menge an Zielen definiert. Dieser Status wird durch die Funktion $status(G, R)$ mit G für die Menge der Ziele und R als der Menge der vorhandenen Ressourcen. Für die Ergebnisse der Berechnung eines Status gilt:

- status(G,R) = **Safe** wenn: $\bigoplus P_i \sqsubseteq R$
- status(G,R) = **Schuleable** wenn: $\bigotimes P_i \sqsubseteq R \sqsubset \bigoplus P_i$
- status(G,R) = **Schedule-dependent** wenn: $\bigotimes N_i \sqsubseteq R \sqsubset \bigoplus N_i$
- status(G,R) = **Conflicting** wenn: $R \sqsubset \bigotimes N_i$
- Sonst status(G,R) = **Uncertain**

Mit diesen Regeln kann man bestimmen, ob eine Menge an Zielen in einer gewissen Beziehung zu den zur Verfügung stehenden Ressourcen steht.

Für die Berechnung des Status einer Aktivierung eines Ziels oder einer Menge von Zielen geht man folgendermassen vor. Man berücksichtigt lediglich die Ressourcen die das neue Ziel oder die Menge an neuen Zielen verwendet und ignoriert die Ressourcen, die nicht von dem zu aktivierenden Ziel verwendet werden aber von den bereits aktiven Zielen. Das bedeutet, dass die Menge von Ressourcen der aktiven Zielen durch die Menge von Ressourcen des zu aktivierenden Ziels beschränkt wird.

Formal bedeutet das, dass man die vorhandene Menge an Ressourcen durch die zu übernehmende Menge an Ressourcen beschränkt. Es gilt für Ressourcenmengen $R_1, R_2 : R_1 \downarrow R_2 = \{(t, \text{if } R_2(t) > 0 \text{ then } R_1(t) \text{ else } 0) | t \in \mathcal{T}\}$ Mit Hilfe dieser Beschränkung kann man dann die Definitionen von status(G,R) verwenden um das Risiko einer Aufnahme zu bestimmen.

In dem Fall, dass das neue Ziel oder die Menge an Zielen keinerlei Ressourcen benötigt, kann man ohne eine weitere Berechnung anstellen zu müssen davon ausgehen, dass es sicher ist dieses Ziel zur weiteren Bearbeitung aufzunehmen. In dem Fall, dass das Ziel oder die Menge an Zielen keine notwendigen Ressourcen benötigt, braucht man sich nur auf die Fälle mit den möglichen Ressourcen zu beschränken. In allen anderen Fällen, also den Fällen in denen $N_G(t) = x \wedge P_G(t) = y | x \leq y$ gilt, berechnet man den Status nach den Angaben des Algorithmuses in Abbildung 4.5. Dieser Algorithmus aus [TWPF2002b] berechnet das Risiko der Aufnahme einer neuen Menge an Zielen (G) bei bestehender Menge an aktiven Zielen (E) mit vorhandenen Ressourcen (R).

```
function status(G,E,R)
⟨N_G, P_G⟩ := S(G)
for each resource type t ∈ T
      if N_G(t) = 0 ∧ P_G(t) = 0 then statusG[t]:= safe
        else if N_G(t) = 0 then
            status_t(E ∪ G, R) = Schedulable then statusG[t]:= Schedulable
            else statusG[t] = Uncertain
      else if status_t(E ∪ G, R) = Conflicting then statusG[t]:= Uncertain
          else if status_t(E ∪ G, R) = Schedulable&Schedule-dependent then
              statusG[t]:= Uncertain
              else if status_t(E ∪ G, R) = Schedule − dependent then
                  statusG[t] := Uncertain
                  else statusG[t]:= status_t(E ∪ G, R)
endfor
if ∃t : statusG[t] = Conflicting then return Conflicting
if ∀t : statusG[t] = Safe then return Safe
if ∀t : statusG[t] ∈ {Safe, Schedulable&Schedule-dependent,Schedulable} then
      if ∃t : statusG[t] = Schedulable&Schedule-dependent then
        return Schedulable&Schedule-dependent
      else return Schedulable
if ∃t : statusG[t] = Schedule-dependent then return Schedule-dependent
else return Uncertain
```

Abbildung 4.5: Algorithmus zur Bestimmung des Risikos, aus [TWPF2002b]

Reaktion auf Konflikte

Nachdem das Risiko der Aktivierung einer Menge von Zielen berechnet worde, kann der Agent in Abhängigkeit von dem Ergebnis dieser Berechnung entscheiden, welche weiteren Schritte er unternehmen will. Ein rationaler Agent wird dabei ein Ziel, welches im Konflikt (`status(G,E,R) = Conflicting`) mit den aktiven Zielen steht, nicht aktivieren. Es ist aber auch möglich, ein aktives Ziel oder mehrere aktive Ziele zugunsten eines zu aktivierenden Ziels zu opfern. Dies implementiert dann einen Prioritätsgedanken.

Was soll aber mit den Fällen geschehen die nicht so eindeutig sind wie die Typen Konflikt und Sicher? Hierbei kommt es ganz darauf an, welche Strategie der betreffende Agent verfolgt.

Das kann zum einen sein, dass der Agent vorsichtig ist und auch bei unsicherer Lage eine Annahme verweigert. Oder der Agent ist optimistisch und nimmt Ziele solange an wie sie definitiv nicht in Konflikt mit der aktiven Menge an Zielen stehen. Wenn der Agent solch eine optimistische Strategie verfolgt, bieten sich weitere Verfahren an. So sollte der Agent dynamisch nicht sichere

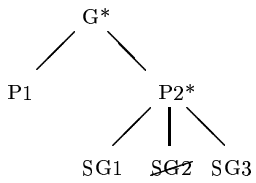

Abbildung 4.6: Aktualisierung der Resource Summaries anhand eines Goal-plan Tree

Ziele überwachen, neuen Ressourcenbedarf von schon teilweise erfüllten Ziels berechnen (dies kann sich auf die Aufnahme von weiteren Zielen auswirken, da Ressourcen von erfüllten Subziele nicht mehr relevant sind und sich somit eine neue Konstellation an Ressourcen ergibt). Wie die „Resource Summaries" dynamisch aktualisiert werden, wird im Folgenden erläutert.

Dynamische Aktualisierung von Resource Summaries

Wie schon aus 4.3.1 bekannt, werden auch hier wieder „Goal-plan Trees" benutzt. Jedes Ziel besitzt einen solchen „Goal-plan Tree", der aus Plan- und Zielknoten für ein bestimmtes Ziel, dem sogenannte Wurzelziel besteht. Jedem Knoten werden die besprochenen „Resource Summaries" zugeteilt. Sollte nun eine Subziel oder ein Plan eines Ziels erfüllt werden, so werden die „Resource Summaries" der Elternknoten neu berechnet. Diese Neuberechnung schlägt sich bis zum Wurzelziel durch. Ein Beispiel ist in Abbildung 4.6 zu sehen. Das Subziel SG_2 wurde erfolgreich beendet. Somit kann die „Resource Summary" des Plans P_2 neu berechnet werden, da die Ressourcen die das SG_2 in die Berechnung der resource summary von P_2 eingebracht hat nun nicht mehr für die aktuelle Bestimmung zählen. Ebenso muss aber auch der Ressourcenvorrat, den der Agent zur Verfügung hat, aktualisiert werden, um so die tatsächlich verbrauchten Ressourcen des SG_2 zu ermitteln und in die Bestimmung der Risikoanalyse mit einfließen zu lassen. Wann genau solch eine Aktualisierung stattfinden sollte ist eine Frage der Optimierung des Laufzeitverhalten der Berechnung. Diese Frage wird in der Analyse dieses Verfahren aufgegriffen.

4.3.3 Positive Interaktionen – Effect Summary

Ziele eines Agenten können nicht nur, wie in den oben beschriebenen Kapiteln, negativ, sondern auch positiv interagieren. Eine Situation in der eine positive Interaktion zwischen Zielen eines Agenten auftritt, zeichnet sich durch das Vorhandensein eines gemeinsamen Subzustandes aus. Dieser gemeinsame Subzustand wird von den Zielen während ihrer Bearbeitung eingenommen. Die Idee die einer Ausnutzung einer solchen positiven Interaktion zugrunde liegt, ist in einer ökonomischen Betrachtungsweise der Bearbeitung der betreffenden Ziele begründet. Das bedeutet, dass der Subzustand nicht für ein jedes Ziel separat erarbeitet wird, sondern nur einmal für alle.

Um positive Interaktionen zu erkennen, werden in dem hier vorgestellten Verfahren aus [TPW2003b], „Effect Summaries“ verwendet. Diese „Effect Summaries entsprechen denen die bereits in Kapitel 4.3.1 beschrieben wurden und repräsentieren somit die Effekte die ein Ziel während seiner Bearbeitung erwirkt. Aufbauend auf dieser Grundlage werden in zwei weiteren Datenstrukturen, der *„Definitly Mergeable Plans“ (DMP)* und der *„Possibly Mergeable Plans“ (PMP)* die Effekte gespeichert für die definitiv (*DMP*) die beteiligten Pläne zusammengefasst werden können bzw. für die möglicherweise (*PMP*) die beteiligten Pläne zusammengefasst werden können. Durch dieses Zusammenfassen von Plänen wird also versucht positive Interaktionen auszunutzen.

Ebenso wie die vorhergehenden, auf ‚Summary“ Information basierende Verfahren, setzt auch dieses Verfahren eine geeignete Repräsentation von Zielen und Plänen voraus. Diese Repräsentation entspricht der in [TPW2003] vorgestellten und wurde bereits in 4.3.1 ausführlicher beschrieben, so dass hier darauf verzichtet wird.

Effect Summaries

Ein „Effect Summary“ stellt Informationen über die Effekte eines Plans oder Ziels dar und setzt sich aus den Teilmengen der definitiven Effekte (D) und der möglichen Effekte (P) zusammen.
Formal gilt für ein „Effect Summary“ eines Ziels $\mathcal{S}_E(G)$:

$$\mathcal{S}_E(G) = \langle \mathcal{D}_E, \mathcal{P}_E \rangle$$

Definitive Effekte $\mathcal{D}_E$ sind solche Effekte die definitiv erreicht werden. Das bedeutet, dass diese Effekte auf allen möglichen Pfaden, die zur Erreichung eines Ziels beschritten werden können, auftreten werden. Das bedeutet aber nicht, dass diese Effekte von einem jedem auf solch einem Pfad vorhandenem Plan erzeugt werden müssen. Vielmehr bedeutet dies, dass diese Effekte von mindestens einem Plan auf jedem möglichen Weg zur Bearbeitung eines Ziels erreicht werden.

Potentielle Effekte $\mathcal{P}_E$ hingegen, sind Effekte die möglicherweise bei der Bearbeitung eines Ziels auftreten können. Das bedeutet, dass diese Effekte von mindestens einem Plan auf mindestens einem Pfad zur Bearbeitung eines Ziels verursacht werden, höchstens aber auf $n-1$, bei n = Anzahl aller möglicher Pfade, auftreten dürfen. Es gilt weiterhin, dass definitive Effekte keine Untermenge potentieller Effekte sind:

$$D_e \cap P_e = \emptyset \tag{4.1}$$

$\mathcal{D}_E, \mathcal{P}_E$ beinhalten jeweils Effekte und die dazugehörigen Pläne die diese Effekte hervorbringen. Als Beispiel sei hier der Aufbau der Menge der definitiven Effekte angegeben.

$$\mathcal{D}_E = \{(\text{effect}, \{\text{plan}, \text{plan}, \ldots), (\text{effect}, \{\text{plan}, \ldots\}) \ldots\}$$

Berechnung der Effect Summaries

Zur Berechnung der „Effect Summaries“ dient hier wie in 4.3.1 eine Baumstruktur, die als „Goal-plan Tree“ bezeichnet wird. Ebenso wie in 4.3.1 spezifiziert man diesen „Goal-plan Tree“ als „Interaction Tree“. Die Knoten dieses „Interaction Tree“ werden mit „Effect Summaries“ versehen. Bevor genauer auf die Berechnung der „Effect Summary“ für Pläne und Ziele eingegangen wird, werden die Hilfsoperatoren Schnittmenge, Vereinigung und Subtraktion definiert. Diese Hilfsoperatoren erleichtern die spätere formale Spezifikation der Berechnungen von „Effect Summaries“ für Pläne und Ziele.

Der Schnittmengenoperator $\cap^*$ bildet aus zwei Mengen von Effekten die Menge der Effekte die sowohl in der einen, wie auch in der anderen Menge vorhanden sind. Dabei werden die Pläne die zu diesen Effekten gehören ebenfalls vereinigt. Es gilt:

$$E_1 \cap^* E_2 = \{(e_1, p_1 \cup p_2) | (e_1, p_1) \in E_1 \wedge (e_2, p_2) \in E_2 \wedge e_1 = e_2\}$$

Der Vereinigungsoperator $\cup^*$ kombiniert aus zwei Mengen von Effekten eine neue Menge mit folgenden Randbedingungen. Die Effekte in der neuen Menge sind Effekte die:

1. in E_1 sind aber nicht in E_2
2. in E_2 sind aber nicht in E_1 und
3. die in $E_1 \cap^* E_2$ sind.

Es gilt somit:

$$E_1 \cup^* E_2 = \{(e,p) | ((e,p) \in E_1 \wedge (e,p) \notin E_2) \vee ((e,p) \in E_2 \wedge (e,p) \notin E_1) \\ \vee ((e,p) \in E_1 \cap^* E_2)\}$$

Als letzter Operator wird der Subtraktionsoperator definiert. Die Subtraktion einer Menge von Effekten von einer anderen Menge von Effekten ergibt eine Menge von Effekten, die nur solche Effekte enthält, die durch Pläne aus dem Minuend hervorgebracht werden. Formal gilt:

$$E_1 - E_2 \quad = \quad \{(e,p) | ((e,p) \in E_1) \wedge \neg(\exists p^{'}.(e,p^{'}) \in E_2)\}$$

Jetzt kann die Berechnung der „Effect Summary“ für Pläne und Ziele betrachtet werden. Wie gewohnt setzt sich die Berechnung von „Summary“ Informationen aus den lokalen „Summary“ Informationen und den „Summary“ Informationen der Folgeknoten zusammen. Dabei unterscheidet man zwischen Zielknoten und Planknoten.

Effect Summary eines Ziels: Der „Effect Summary“ eines Zielknoten kombiniert sich aus den „Effect Summaries“ der für dieses Ziel relevanten Planknoten. Ein lokaler „Effect Summary“ wird hier nicht berücksichtigt, da ein Zielknoten

selbst keine eigenständigen Effekte außer die seiner relevante Pläne besitzt. Es gilt:

$$S_E(G) = \bigotimes_{p \in G.plans} S_E(p)$$

Zu beachten ist, dass bei der Bearbeitung eines Ziels nicht festgelegt ist, welcher der vorhandenen Pläne ausgeführt wird. Pläne stellen lediglich eine Möglichkeit dar um das für diese Pläne spezifizierte Ziel zu erreichen. Somit gilt für die Festlegung der definitiven und potentiellen Effekte folgendes. Definitive Effekte eines Ziels ergeben sich aus den Effekten die definitive für **alle** zur Verfügung stehenden Pläne sind. Potentielle Effekte eines Ziels ergeben sich aus den definitiven Effekten eines jeden Plans die **nicht** definitiv für alle Pläne sind und aus den potentiellen Effekten eines jeden Plans. Formal gilt dann für die Bestimmung der definitiven bzw. potentiellen Effekte zweier Effektmengen:

$$\langle D_{E1}, P_{E1} \rangle \otimes \langle D_{E2}, P_{E2} \rangle = \langle (D_{E1} \cap^* D_{E2}), (P_{E1} \cup^* P_{E2} \cup^* ((D_{E1} \cup^* D_{E2}) - (D_{E1} \cap^* D_{E2}))) \rangle$$

Effekt Summary eines Plans: Für die Bestimmung des „Effect Summary“ eines Plans kombiniert man die lokale „Effect Summary“ mit den „Effect Summary“ der Subziele des Plans. Es gilt:

$$S_E(p) = \bigoplus_{g \in p.subgoals} S_E(g) \oplus \langle \{(e, \{p.name\}) | e \in p.EF\}, \{\} \rangle$$

Dabei berechnen sich die Teilmengen D und P folgendermaßen. Die definitiven Effekte zweier Subziele sind die Vereinigung der definitiven Effekte der beiden Ziele. Identisches gilt für die Bestimmung der potentiellen Effekte beider Ziele. Die potentiellen Effekte sind die potentiellen Effekte beider Subziele. Für den Fall, dass ein Effekt sowohl in der Menge der potentiellen wie auch der definitiven Effekte vorhanden ist, wird dieser Effekt der Menge der definitiven Effekte zugerechnet. Dies lässt sich wie folgt erklären. Subziele eines Plans werden definitiv für eine erfolgreiche Bearbeitung des Planes abgearbeitet. Dies würde bedeuten, dass ein und derselbe Effekt einmal in der Menge der definitiven Effekte des Plans und zugleich aber auch in der Menge der potentiellen Effekte des Plans vorhanden ist. Nach Gleichung 4.1 ist dies aber nicht möglich. Somit wird der Effekt nur einer Menge zugeordnet und das ist aufgrund der Tatsache, dass der Effekt in einem Subziel bereits definitiv ist, die definitive Menge. Es gilt für die Kombination von definitiven und potentiellen Effekten:

$$\langle D_{E1}, P_{E1} \rangle \oplus \langle D_{E2}, P_{E2} \rangle = \langle (D_{E1} \cup^* D_{E2}), ((P_{E1} \cup^* P_{E2}) - (D_{E1} \cup^* D_{E2})) \rangle$$

Zur Bestimmung der „Effect Summary“ einer Menge an Zielen geht man folgendermaßen vor. Man bestimmt wie gewöhnlich die „Effect Summaries“ der einzelnen Ziele und kombiniert diese dann mit dem Operator $\oplus$. Es gilt formal: $S_E(GoalSet) = \bigoplus_{g \in GoalSet} S_E(g)$

Dynamische Aktualisierung von Effect Summaries

Der „Interaction Tree“ bietet Informationen über die Effekte die bei der Bearbeitung eines Ziels entstehen können. Da diese Effekte aber während der Laufzeit variieren können, bedarf es geeigneter Aktualisierungsstrategien um die Informationen des „Interaction Tree“ aktuell zu halten. Die Strategie die in [TPW2003b] verfolgt wird ist Folgende:

Vollständig abgearbeitete Planknoten und Zielknoten werden aus der Baumstruktur „Goal-plan Tree“ entfernt. Je nach Ergebnis der Abarbeitung, fehlerhaft oder erfolgreich, werden weitere Aktionen an der Baumstruktur durchgeführt. Anschließend werden die entsprechenden „Effect Summaries“ neu berechnet. Es werden die Aktualisierungsschritte nach dem Typ des Knoten unterschieden. Für vollständig bearbeitete Pläne gilt:

- Der entsprechende Planknoten wird aus dem „Interaction Tree“ entfernt.
- Wurde der Plan erfolgreich vollständig bearbeitet, so ist auch das Ziel für welches dieser Plan galt erfolgreich und vollständig bearbeitet und wird ebenso aus dem „Interaction Tree“ entfernt. Veränderungen der „Effect Summary“ werden an die Väterknoten propagiert.
- Ist der Plan fehlgeschlagen, so kommt es darauf an ob das betreffende Ziel weitere Pläne ausführen kann. Kann das Ziel keine weiteren Pläne ausführen, so schlägt es fehl. Das bedeutet wiederum, für den Fall das dieses Ziel eine Subziel ist, dass der betreffende Plan fehlschlägt. Alle drei Knoten werden entfernt und die Veränderung der „Effect Summary“ an die Väterknoten propagiert.

Für vollständig bearbeitete Ziele gilt:

- Der entsprechende Zielknoten wird aus dem „Interaction Tree“ entfernt.
- Die „Effect Summary“ des Vaterknoten wird neu berechnet.

Ausnutzung von positiven Interaktionen

Erzeugen Pläne von zwei verschiedenen Zielen denselben Effekt, so besteht die Möglichkeit diese beiden Pläne derart zu kombinieren, dass nicht unbedingt beide Pläne zur Erreichung des besagten Effekts ausgeführt werden müssen. Dieses wird im Folgenden mit dem Begriff „merging“ bezeichnet. Identifiziert werden positive Interaktionen wie gewohnt durch die „Effect Summary“ der beteiligten Ziele. Für das „merging“ der betroffenen Pläne werden weitere Datenstrukturen benötigt. Diese sind zum einen die *„Definitely Mergeable Plans“ (DMP)* und zum anderen die *„Possibly Mergeable Plans“ (PMP)*. Diese Datenstrukturen beinhalten die Pläne die definitiv bzw. potentiell zusammengeführt (merging) werden können. Um Deadlocks zu vermeiden wird eine weiter Datenstruktur verwendet. Dies ist die *„Waiting Goal List“(WGL)*. Diese Datenstruktur beinhaltet eine Liste von zeitweilig suspendierten Zielen.

Die Datenstruktur *DMP* besteht aus Effekten, den jeweils dazugehörigen Zielen und die jeweils für den betreffenden Effekt verantwortlichen Pläne. Alle Effekte in der *DMP* sind definitive Effekte in allen spezifizierten Ziele. Die Datenstruktur *DMP* ist folgendermassen definiert.

$$DMP = [(e_1, (G_1\{P_1, P_2, \ldots\}), (G_2\{P_3, \ldots\}), \ldots), (e_2, \ldots), \ldots]$$

Die *PMP* ist ähnlich strukturiert wie die *DMP*. Im Unterschied zur *DMP* enthält die *PMP* Effekte die potentiell in allen Zielen sind oder definitive in einigen und potentiell in anderen der aufgeführten Ziele sein können. Um was für einen Effekt es sich jeweils handelt, wird durch ein „Flag" kenntlich gemacht. Eine Datenstruktur *PMP* ist wie folgt definiert:

$$PMP = [(e_3, (G_3, p/d^3, \{P_4, P_5, \ldots\}), (G_4, p/d, \{P_6, \ldots\}), \ldots), (e_4, \ldots), \ldots]$$

In einem kleinen Fallbeispiel soll nun gezeigt werden wie diese Datenstrukturen verwendet werden können. Nehmen wir an, ein Agent besitzt zwei Ziele G_1 und G_2. Die zu diesen Zielen gehörenden Pläne sind für $G_1 : P_1, P_2$ und für $G_2 : P_3, P_4$. Beide Ziele haben einen gemeinsamen Effekt der durch diese Pläne hervorgerufen wird.

Es bestehen drei Möglichkeiten wie dieser Effekt den beiden Zielen gemein sein kann.

1. Der Effekt ist ein definitiver Effekt von G_1 und G_2
2. Der Effekt ist ein definitiver Effekt von G_1 aber ein potentieller Effekt von G_2
3. Der Effekt ist ein potentieller Effekt von G_1 und G_2

1. Situation: Hierbei handelt es sich um einen definitiven Effekt beider Ziele. Das bedeutet, dass dieser Effekt definitiv durch beide hervorgebracht wird, unbeachtet der Tatsache welche der entsprechenden Pläne zur Bearbeitung der Ziele ausgewählt werden. Daraus folgt, dass man diesen Effekt in der Datenstruktur *DMP* vermerkt. Es gilt somit:

$$DMP = [(e, (G_1\{P_1, P_2, \}), (G_2\{P_3, P_4\})), \ldots]$$

Diese Zuordnung des Effektes e zur *DMP* zeigt dem Agenten an, das eine definitiv positive Interaktion zwischen diesen beiden Zielen besteht. Der Agent sollte dies wahrnehmen und eine geeignete Ablaufsteuerung festlegen. Die grundlegende Idee hierbei ist, die Pläne die ausgeführt werden sollen zu synchronisieren, indem sie als ausführbar gekennzeichnet werden und anschließend auf andere Pläne die ebenfalls diesen Effekt hervorbringen warten. Wird dann solch ein weiterer Plan als ausführbar gekennzeichnet, so können diese beiden Pläne zusammengeführt werden.

Folgende Schritte setzen diese Idee um. Sie sollten in den Ausführungszyklus eines Agenten integriert und vor der Ausführung eines Planes abgearbeitet werden.

```
00| P := next Plan to execute
01| G := goal of P
02| If P is in DMP then
03|   e is associated effect of P in DMP
04|   if all goals associated with e (other than G) have an associated plan that has
05|   been flagged ready then
06|     perform merge between plans
07|   else if any of the goals (other than G) associated with e, that does not have a
08|   plan flagged ready is already in the WGL then
09|     execute P (in order to avoid any potential deadlocks)
10|   else
11|     flag P as ready; suspend G; place G in WGL
```

Abbildung 4.7: Aktionen für Pläne in DMP, aus [TPW2003b].

Das bedeutet für einen zur Ausführung stehenden Plan folgendes: Gibt es Ziele, ausser demjenigen welches zu dem auszuführenden Plan gehört, die ebenfalls den Effekt hervorbringen, so prüfe ob diese Ziele als „bereit" gekennzeichnete Pläne besitzen. Für Pläne auf die das zutrifft, kann eine Zusammenführung vollzogen werden. Gibt es aber unter besagten Zielen irgendein Ziel welches keinen bereit gestellten Plan hat und zudem noch in der *„Waiting Goal List"* vermerkt ist, so wird zur Vermeidung von Deadlocks der zur Ausführung stehenden Plan einfach ausgeführt. Besteht die Situation in der die besagten Ziele keine bereiten Pläne besitzen aber auch nicht in der *„Waiting Goal List"* vermerkt sind, so wird der betreffende Plan als bereit gekennzeichnet, die Bearbeitung des dazugehörenden Ziels suspendiert und das suspendierte Ziel in der *„Waiting Goal List"* vermerkt.

2. Situation: Im zweiten Fall ist der Effekt für G_1 ein definitiver Effekt, aber für G_2 nur noch ein potentieller. Das bedeutet für diesen Effekt im Fall von G_2: es kann nicht definitiv sichergestellt werden, dass dieser Effekt eintritt. Dieser Effekt wird nicht in die *DMP* übernommen, sondern in der *PMP* gespeichert. Denn durch das potentielle Vorhandensein des Effektes e in G_2 kann man nicht mehr von "definitiv mergeable plans" sprechen. Die Informationen in der *PMP* wird folgendermaßen überwacht.

Das bedeutet in diesem Fall für ein zu bearbeitenden Plan folgendes: Gibt es für den Effekt den dieser betreffende Plan hervorbringt, ein weiteres Ziel, welches diesen Effekt ebenfalls hervorbringt und dies definitiv, so wird der Eintrag aus der *PMP* in die *DMP* transferiert. Gibt es aber kein anderes Ziel, welches diesen Effekt definitiv hervorbringt, so bieten sich zwei Strategien an. Erstens kann der Agent vorsichtig sein und den betreffenden Plan einfach ausführen, weil ja nicht sichergestellt ist ob ein anderes Ziel diesen Effekt jemals hervorbringen wird. Oder er ist mutig und stellt diesen Effekt ebenfalls in die *DMP* und hofft darauf, dass die anderen Ziele so bearbeitet werden, dass der Effekt dennoch hervorgebracht wird.

Für das korrekte Funktionieren des Verfahrens, muss die Information in der „Effect Summary" aktuell sein. Deshalb gilt folgendes: Wird ein po-

```
00| If P3 is next Plan to execute then
01|   store entry of effect e in DMP
02|   flag P3 ready
03| If P1 is next Plan to execute then
04|   choose between strategies: 1. cautious or 2. optimistic
05|   if strategy = cautious then
06|     execute P1; remove entry from PMP
07|   if strategy = optimistic then
08|     move entry to DMP; flag P1 ready; suspend G1; put G1 in WGL
```

Abbildung 4.8: Aktionen für Pläne in PMP, aus [TPW2003b].

tentieller Effekt durch eine Aktualisierung der „Effect Summary“ eine definitiver Effekt und gilt für jedes Ziel, welches ebenfalls diesen Effekt hervorbringt, dass dieser definitiv ist, so wird der Effekt und seine Ziele in die *DMP* übernommen. Wenn der Effekt für ein bestimmtes Ziel nicht mehr hervorgebracht wird, dann wird das Ziel aus der *PMP* entfernt. Ebenso wird der entsprechende Eintrag entfernt, wenn nicht mindestens zwei Ziele für einen bestimmten Effekt vorhanden sind. Die entsprechenden Aktionen um diese Situationen zu behandeln, können alle während der Aktualisierung der „Effect Summaries“ getätigt werden.

3. Situation: Hier handelt es sich ausschließlich um potentielle Effekte beider Ziele. Somit würde es sich anbieten den Effekt in die *PMP* zu übernehmen und anschließend beim Erreichen einer der Pläne entweder eine vorsichtige oder optimistische Strategie zu wählen. In einer vorsichtigen Strategie warten die Pläne nicht aufeinander und werden somit unabhängig voneinander ausgeführt. In der optimistischen wird man den Eintrag von der *PMP* in die *DMP* übertragen und davon ausgehen, dass ein gewisse Möglichkeit besteht das die Pläne der Ziele einem „merging“ unterzogen werden können.

In [TPW2003b] wird auch der Fall berücksichtigt, dass Pläne nicht zusammengeführt werden sollen. So wird für einen jeden Plan ein Attribut „mergeable“ spezifiziert. Im Standardfall hat dieses Attribut den Wert „true“. Sollen Pläne nicht zusammengeführt werden, so wird das durch den Wert „false“ des Attributs „mergeable“ ausgedrückt.

Zusammenführen (merging) von Plänen

Nachdem festgestellt worden ist, welche Pläne zusammengeführt werden können, soll jetzt dargelegt werden wie dieses Zusammenführen aussieht und unter welchen Randbedingungen es angewendet werden kann. Es wird angenommen, dass zwei Pläne P_1 und P_2 zusammengeführt werden können.

- Wenn die Pläne vom selben Typ sind, dann kann jeder der beiden Pläne ausgeführt werden. Pläne sind z.B. dann vom selben Typ, wenn sie Instanzen ein und der selben Deklaration eines Planes sind.

- Wenn die Pläne nicht vom selben Typ sind aber sie nur e und keine weiteren Effekte herstellen, dann kann auch hier unbedenklich jeder der beiden Pläne ausgeführt werden.
- Wenn P_1 e und x, wobei x ein potentieller Effekt des Ziels G_1 ist hervorbringt und P_2 nur e, dann kann wiederum jeder der beiden Pläne ausgeführt werden.
- Wenn P_1 e und x erreicht und x dabei ein definitiver Effekt ist und P_2 nur e erreicht, dann muß $P1$ verarbeitet werden, da sonst der notwendige Effekt x nicht hergestellt werden würde.
- Wenn P_1 e und x erreicht und P_2 e und y und $x$und y sind jeweils definitive Effekte für die entsprechenden Ziele und $x \neq y$, dann kann keiner der beiden Pläne stellvertretend ausgeführt werden. Vielmehr müssen diese beiden Pläne individuell ausgeführt werden. Solch eine Situation wird als „useless wait"bezeichnet.

Nach einem Zusammenführen von Plänen muss die Datenstruktur *DMP* folgendermaßen aktualisiert werden.

- Der entsprechende Eintrag im *DMP* wird entfernt.
- Nach vollständiger Bearbeitung der zusammengeführten Pläne, werden die „Effect Summaries" der zu diesen Plänen gehörenden Ziele aktualisiert. Anhand dieser Aktualisierung kann festgestellt werden, ob weitere Interaktionen bezüglich des Effektes e vorhanden sind. Trifft das zu, so können diese Pläne u.U. auch zusammengeführt werden.

4.3.4 Bewertung der Verfahren

Die hier vorgestellten auf „Summary" Information basierenden Verfahren, sind in ihrer Grundstruktur vielseitig verwendbare Verfahren. Sie beziehen sich nicht auf einen Spezialanwendungsfall, sondern sind domänenunabhängig. „Summary" Information repräsentiert dabei beliebige Informationen, die für eine Entscheidung zwischen Zielen bzw. Plänen verwendet werden können. Dies macht sie zu einer sehr interessanten Struktur für eingehendere Betrachtungen.

Im Folgenden werden die in Kapitel 4.3.1, 4.3.2 und 4.3.3 vorgestellten Verfahren anhand der in der Einleitung zu diesem Kapitel beschriebenen Kriterien bewertet.

Interaction Summary:

Die Verwendung einer „Interaction Summary" fordert dem Entwickler eines Agenten ein erhöhtes Maß an Spezifizierungsaufwand ab. Auf der Ebene der Pläne muss er Effekte, Kontextbedingungen, P-Effekte und „Dependency-entries" angeben. Für Ziele ist lediglich die Angabe von Kontextbedingungen notwendig, die restlichen Angaben werden durch das Verfahren berechnet. Dies stellt eine

Fülle von Anforderungen dar, die ein gewisses nicht zu vernachlässigendes Fehlerpotential bergen. Zum Beispiel muss bei der Spezifizierung der Effekte eines Plans darauf geachtet werden, dass die spezifizierten Effekte mit denen, die der Plan in seinem Rumpf deklariert, übereinstimmen. Enstehen hier Inkonsistenzen, so verhält sich das System nicht wie gewünscht und eine Beseitigung des Fehlers wird so eine zeitraubende Angelegenheit. Generell fraglich ist die Spezifizierung von P-Effekten. P-Effekte sind nur dann nutzbar, wenn Pläne eine ausreichend tiefe Baumstruktur aufweisen können. Das soll bedeuten, dass mindestens zwei Subziele bei der Abarbeitung eines Plans vorgesehen sind. Ist dies nicht der Fall, so sind P-Effekte nicht anwendbar. Auch die Spezifizierung der Ausdrücke für Effekte oder Kontextbedingungen bietet Kritikpunkte. Sie ist zu allgemein gehalten. Insbesondere für eine spätere Entscheidung ob eine Ziel aktiviert werden kann ist sie nicht ausreichend konkret für eine Realisierung. Denn ob zwei Ausdrücke kompatibel zu einander sind, ist eine sehr allgemeine Formulierung. Hinter dieser Forderung können sich eine Vielzahl von Abwägungen verbergen. Hier muss spezifischer vorgegangen werden.

Die Entscheidungsmöglichkeiten die dieses Verfahren bietet, sind auf die Feststellung der Kompatibilität von Ausdrücken gegründet. Da, wie schon gesagt, Kompatibilität ein allgemeiner Begriff ist, sind auch die Entscheidunsmöglichkeiten einfach gehalten. So ist in einer ersten Stufe feststellbar, ob ein Ziel aktiviert werden kann oder nicht. Falls es nicht aktiviert werden kann, bietet das Verfahren zwei Auswahlmöglichkeiten. Das Ziel kann doch aktiviert werden wenn die inkompatiblen Ausdrücke nur potentiell auftreten können oder das Ziel wird generell nicht aktiviert. Das Verfahren verschwendet, durch die zu allgemeine Auslegung der Ausdrücke, Potential bei der Entscheidung, ob pro oder contra einer Aktivierung. Für einen spezifischeren Entscheidungsspielraum müssen die Ausdrücke konkretisiert werden. So ist z.B. denkbar, wie im folgenden angesprochen werden wird, Ausdrücke auf Ressourcen einzugrenzen.

Zusammengefasst ist dieses Verfahren ein in den Ansätzen vielversprechendes Verfahren. Jedoch müssen die Ausdrücke unbedingt konkretisiert werden, um die Entscheidungen handhabbarer zu machen. Ebenso ist eine Eingrenzung der in der „Summary“ Information betrachteten Informationen denkbar. Zuviel Information erhöht den Berechnungsaufwand.

Resource Summary:

Die Kritikpunkte die im vorhergehenden Abschnitt erwähnt wurden, scheinen in dem Verfahren basierend auf „Resource Summary“ berücksichtigt worden zu sein. Hier wird sich auf eine bestimmte, konkrete Information konzentriert, die zugleich auch die Ausdrücke mit denen diese Information spezifiziert wird, spezifischer werden lässt. Ressourcen sind ein gängiges zu berücksichtigendes Mittel, wenn es darum geht festzustellen welche Objekte eine System für seine Arbeit benötigt. Sie sind spezifisch genug um eine Kompatibilitätsprüfung zu vereinfachen. Sie sind aber auch allgemein genug, um eine Reihe von unterschiedlichen Objekten zu vereinen. Ressourcen stellen damit eine interessante Verfeinerung der im vorhergehenden Abschnitt beschriebenen Interaktionen dar.

Dadurch, dass die Spezifizierung der benötigten Information auf die Spezifizierung von Ressourcen beschränkt wurde, verringert sich auch der Aufwand mit dem die Information dargestellt werden musste. Es ist jetzt nur noch die Angabe des Bedarfs an einer bestimmten Ressource pro Plan notwendig, um das Verfahren mit ausreichend Information zu versorgen. Für Ziele ist keine Spezifikation von Information notwendig. Diese wird, wie im Fall der Interaktionen, durch das Verfahren generiert. Jedoch besteht auch bei diesem Verfahren erhöhtes Fehlerpotential. Denn die spezifizierten Ressourcen für einen Plan müssen auch mit den tatsächlich benötigten Ressourcen im Rumpf übereinstimmen. Ist dies nicht der Fall, kommt es, wie im Fall der Interaktionen, zu unangenehmen Inkonsistenzen im Verhalten des Verfahrens, welche sich mit einem hohen Aufwand in der Beseitigung niederschlagen können.

„Ressource Summary“ Information bietet im Gegensatz zur „Interaction Summary“ mehr Entscheidungsmöglichkeiten. Durch die klare Spezifikation wann Pläne bzw. Ziele kompatibel sind (siehe `status` Funktion) kann eine Entscheidung gezielter zwischen verschiedenen Situationen wählen. So sind in diesem Verfahren Entscheidungen in bis zu fünf Fällen zu treffen. Dies stellt einen Gewinn im Vergleich zur Interaktion dar.

Zusammengefasst ist das „Resource Summary“ Verfahren eine Verbesserung des „Interaction Summary“ Verfahren. Es ist deutlich konkreter und erlaubt damit mehr Entscheidungsmöglichkeiten bei geringerem Aufwand der Informationsspezifikation.

Effect Summary:

„Effect Summary“ Information ist ein Extrakt aus der „Interaction Summary“ Information. Hier in diesem Verfahren wird sie jedoch zur Erkennung von positiven Interaktionen verwendet. Dadurch das hier nur ein Teil der „Interaction Summary“ verwendet wird, reduziert sich auch der Aufwand mit dem die dem Verfahren zugänglich zu machende Information spezifiziert werden muss. Für einen Plan müssen die Effekte angegeben werden die der Plan hervorbringt. Die Effekte der Ziele berechnen sich aus dem Verfahren.

Die entscheidende Schwachstelle dieses Verfahrens, ist die Entscheidungsmöglichkeit und vor allem ihr Nutzen. Das Verfahren kann nur entscheiden ob Pläne die einen bestimmten Effekt erwirken zusammengefasst werden sollen. Diese Entscheidung ist mit zahlreichen Randbedingungen versehen, die in Kapitel 4.3.3 aufgeführt sind. So ist anzunehmen, dass nur in ganz bestimmten Fällen ein Zusammenführen möglich wird. Abgesehen von dieser starken Einschränkung ist vor allem der Nutzen dieses Verfahrens äußerst fragwürdig. Die Ausnutzung von positiven Interaktionen steht in keinem Verhältnis zu Berechnungsaufwand der dafür geleistet werden muss, so dass hier davon abgesehen werden muss solch ein Verfahren zu implementieren

Zusammengefasst ist das „Effect Summary“ Verfahren zur Ausnutzung von positiven Interaktionen im Spezifikationsaufwand für die Informationen geringer als das „Interaction Summary“ Verfahren und vergleichbar mit den „Resource Summary“ Verfahren. Jedoch macht der sehr niedrige Nutzungsgrad dieses Ver-

fahren es uninteressant gegenüber z.B. dem „Ressource Summary“ Verfahren.

4.4 Regelbasierte Mechanismen

Regelbasierte Systeme gründen sich auf der Verwendung einer Menge von Regeln um geeignete Schlussfolgerungen anstellen zu können. Regeln beschreiben in einer *Wenn ... dann ...* Manier Situationen in denen bestimmte Aktionen getätigt werden sollen. Durch diesen konzeptionellen besonderen Aufbau von Regeln können darauf basierende Verfahren eigentlich nur reagierend aktiv werden. Eine vorrausschauende Verhaltensweise ist mit dieser Art von Verfahren nicht zu implementieren. Insbesondere wenn die Anzahl der Regeln ein überdurchschittliches Maß annimmt kann es zu Effektivitätseinbußen des Systems kommen oder einen erheblichen Verwaltunsaufwand bedeuten.

In diesem Abschnitt sollen zwei Verfahren vorgestellt werden, die grundlegend auf der Verwendung von Regeln basieren. Im ersten Teil wird 3APL vorgestellt. 3APL ist eine Programmiersprache die einen ganzheitlichen Ansatz zur Programmierung von Agenten verspricht. Im zweiten Teil wird dann ein Verfahren erläutert, welches durch die Angabe von Regeln, Kriterien für Charakterisierung von Zielen im Unterschied zu „Desires“ festlegen will.

4.4.1 3APL

3APL [HdBvdHM99, DdBDM2003a, DvRDM2003b] ist eine Programmiersprache für Agenten, die Ähnlichkeiten in der Systemstruktur mit dem in Kapitel 2.2.2 erwähnten „Procedural Reasoning System“ aufweist. Die Autoren dieser Programmiersprache plädieren für eine Unterteilung der Programmierung in zwei Bereiche. Zum einen ist dies der Bereich der durch eine Objektsprache definiert wird und zum anderen ist dass der Bereich der durch eine Metasprache definiert wird.

Der Bereich der durch die Objektsprache definiert wird, beschreibt mentalistische Begriffe wie z.B. Ziel oder Belief die bei der Programmierung eines Agenten verwendet werden können. Die Metasprache hingegen, definiert die Beziehungen zwischen diesen Begriffen. Konkrete Begriffe die 3APL in der Objektsprache unterstützt sind Ziele, Beliefs und „Practical Reasoning Rules“ (PR-Regeln). Mit Hilfe der PR-Regeln ist es möglich, speziell auf dem Konstrukt Ziel zu arbeiten.

Eigentliches Ziel der Programmiersprache 3APL ist es, den sonst statisch implementierten Interpreterzyklus programmierbar zu machen und somit unter anderem zu entscheiden, welches Ziel aus der Menge der Ziele ausgeführt werden soll. Daher wurde diese Sprache mit Konstrukten aus einer Metasprache erweitert. Konstrukte der Metasprache legen fest welche PR-Regeln und welche Ziele ausgewählt werden sollen. Momentan ist die Auswahl der Ziele und der Regeln auf einer vordefinierten Ordnung begründet. Die Konstrukte der Metasprache greifen z.B. bei der Auswahl der Ziele auf diese Ordnung zurück.

Practical Reasoning Rules

Der Zweck der PR-Regeln ist die Manipulation der Ziele eines Agenten. Dabei können PR-Regeln aufgrund ihrer Struktur ein Vielzahl von Aktionen darstellen, auf die im Folgenden näher eingegangen werden wird.

In Abbildung 4.9 sind die drei konzeptionellen Ebenen der Programmiersprache 3APL angegeben und man kann daran erkennen, dass die PR-Regeln auf der höchsten Ebene auf den darunterliegenden Zielen des Agenten arbeiten. Somit kann der Agent durch die Verwendung dieser Regeln, seine Ziele überarbeiten und überwachen, also insgesamt reflektive Fähigkeiten gegenüber seinen Zielen zum Ausdruck bringen. Eine PR-Regel ist laut [HdBvdHM99] definiert als:

PR-Regel: $\pi \leftarrow \phi | \pi'$

Dabei bezeichnet π den Kopf der Regel, π' den Körper der Regel und ϕ die Schutzbedingung (Guard) der Regel. Desweiteren werden auch Variablen innerhalb des Kopf und des Körper einer Regel erlaubt. Anhand diesem grundlegendem Aufbau einer PR-Regel kann man verschiedene Typen von Regeln unterscheiden. Diese Typen von Regeln werden benötigt um eine Ordnung auf Regeln erstellen zu können. Diese Ordnung wird später in der Metasprache benötigt, um eine Auswahl zwischen anwendbaren Regeln zu begünstigen, denn die Standardauswahl ohne Ordnungskriterium ist inherent nichtdeterministisch. Man klassifiziert diese Regeln, in Regeln für Fehler (failure rules) $\mathcal{F}$, reaktive Regeln (reactive rules) $\mathcal{R}$, planende Regeln (plan rules) $\mathcal{M}$ und optimierende Regeln (optimisation rules) $\mathcal{O}$. Eine Ordnung darauf wird folgendermaßen gegeben. Es gilt:

$$\mathcal{F} > \mathcal{R} > \mathcal{M} > \mathcal{O}$$

Die Bedeutung und die Struktur dieser Klassen von Regeln ist folgendermaßen erklärt. Die Idee eines Typus „failure rule“ist, dass in bestimmten Situationen ϕ die Bedingungen für eine weitere Ausführung des Ziels π nicht gegeben sind und somit zweierlei Reaktionen darauf denkbar werden. Zum einen kann das betreffende Ziel π fallengelassen werden, dies wird durch die modifizierte Regel [DvRDM2003b] $\pi \leftarrow \phi | \top$ erreicht indem der Körper der Regel bezüglich eines Ziels leer bleibt, oder zum anderen indem das Ziel π durch ein anderes Ziel π' ersetzt wird.

Die Idee die dem Typ einer reaktiven Regel zugrundeliegt ist folgende: Wenn eine bestimmte Situation ϕ eintritt, so kann dies die Notwendigkeit der Erzeugung eines Ziel zur Folge haben. Dies wird in der modifizierten Regel [DvRDM2003b]: $\top \leftarrow \phi | \pi'$ zum Ausdruck gebracht. Regeln für die Planung geben die Umsetzung von Zielen zu Plänen an. Dabei wird ein Ziel im Kopf einer Regel durch ein Subziel oder im Fall das schon ein Subziel vorliegt in eine atomare Aktion umgesetzt. Die Reihe von Verfeinerung eines Ausgangsziels im Kopf einer Regel durch die Anwendung einer oder mehrere Regeln kann so als die Ausführung eines Plans angesehen werden. Regeln für die Planung werden aber hier nicht weiter betrachtet.

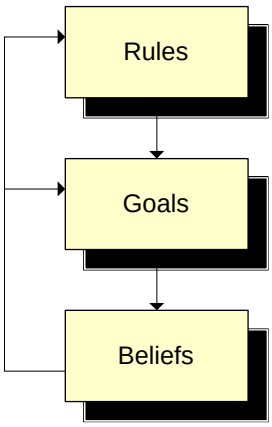

Abbildung 4.9: Konzeptuelle Ebenen in 3APL, aus [HdBvdHM98]

Metasprache

Die Metasprache ist die Sprache mit der der Interpreterzyklus programmiert werden soll. Sie stellt damit ein Kontrollstruktur dar. Kontrolliert werden soll u.a. welches Ziel bearbeitet werden soll und welche der Regeln angewendet werden sollen. Dazu werden Ordnungen auf den Zielen und den Regeln des Agenten definiert. Die Ordnung für Ziele wird unter Berücksichtigung einer Priorität der Ziele gebildet und die Ordnung auf den Regeln ergibt sich aus der oben dargestellten Klassifizierung von Regeln. Das 3APL Verfahren ist also ein Mischverfahren, dass sowohl Regeln wie auch Prioritäten verwendet. Da aber die Hauptarbeit auf Zielen durch die PR Regeln absolviert wird, wurde dieses Verfahren in den Abschnitt Regelbasierte Verfahren eingegliedert. Zur Bestimmung der zu berücksichtigenden Regeln und Ziele wird eine Funktion *max* definiert. Es gilt für Ziele g:

$$max_g(\Pi) = \{\pi \in \Pi | \neg \exists \pi' : \pi' >_g \pi\}$$

Diese Funktion gibt jeweils das Ziel bzw. die Regel an die maximal bezüglich der Ordnung auf den betreffenden Objekten ist. Sie wird u.a. in einer weiteren Operation der Metasprache verwendet. Diese ist die Operation *selex*. Diese Operation wählt in der Menge der Ziele ein Ziel aus, welches maximal bezüglich der definierten Ordnung und welches ausführbar ist. Es sind weitere Funktionen auf den Termen einer Metasprache vorstellbar [DdBDM2003a]. So kann man z.B. die Funktion *gain*: individualGoal $\rightarrow N$ definieren. Diese Funktion gibt den Gewinn den der Agent durch die Bearbeitung dieses Ziels erhalten würde an. Diese Funktionen können dann wiederum in den Operationen der Metasprache verwendet werden um geeignete Effekte zu erreichen.

Bewertung des Verfahrens

Objektebene: Die Programmiersprache 3APL bietet durch die Anwendung der „Practical Reasoning Rules“ grundlegende Möglichkeiten Ziele zu beeinflussen. Dies geschieht, indem der Entwickler eines Agenten, diejenige Situation durch die Spezifizierung von PR Regeln codiert, die relevant für Entscheidungen auf Zielen sind. So kann er z.B. festlegen in welcher Situation der Agent ein bestimmtes Ziel fallenlassen soll oder in welcher Situation er ein Ziel in seine Subziele verfeinern soll. So bestimmt der Entwickler alle Situationen, die aus seiner Sicht wichtig für Entscheidungen auf einem Ziel werden können. Das kann bei einer großen Menge an Zielen schnell zu einer erheblichen Anzahl an Regeln führen. In Jadex ist diese Spezifikation der Situationen erheblich einfacher. Dort werden z.B. in „Dropcondition“ oder „Failurecondition“ ebenso Situationen spezifiziert, welche zu einem Abbruch des Ziels und zu dessen Fallenlassen führen sollen. Daneben gibt es aber auch die „Contextcondition“ mit der man die Regel repräsentieren kann, die die Situation beschreibt die ursächlich für das Erzeugen eines Ziels ist. Die Spezifizierung von Situationen in denen ein Ziel in bestimmte Subziele verfeinert werden kann, ist in Jadex hingegen so nicht möglich. 3APL bietet damit auf dieser Ebene der Objektsprache keine wesentlichen Vorteile hinsichtlich einer Deliberation.

Metaebene: Auf der Ebene der Metasprache hat 3APL den Anspruch den Zyklus des Standard BDI Interpreter variabel zu gestalten. Durch die Verwendung einfacher Programmierkonstrukte kann der Entwickler des Agenten selbst festlegen, welche Aktionen wann ausgeführt werden sollen. Hier geht es mehr um die Verwirklichung eines ganzheitlichen Ansatzes. Es soll der komplette Vorgang der Deliberation und des „Practical Reasoning“ variabel gestaltetet werden. 3APL bietet hier zusätzlich für Ziele lediglich die Festlegung einer Ordnung auf Zielen. Diese Ordnung wird für eine Auswahl der Ziele benötigt, ist aber sehr spartanisch und statisch. In Jadex ist ein ähnliches Konzept in jüngster Zeit integriert worden. Es verwendet die in Kapitel 2.3 beschriebenen Meta-Aktionen und eine Agenda. In diesem Sinne kann Jadex ebenso mit dem Ziel, einen variablen Interpreter spezifizieren zu können aufwarten. Durch die zusätzliche Verwendung der Meta-Aktionen kann Jadex ebenso eine mögliche Ordnung auf Ziele ausnutzen aber weitere zusätzliche Informationen verarbeiten. So kommt man zu dem Fazit, dass auch die Metasprache keine wesentlichen Neuerungen hervorbringt, die in einem möglichen Deliberationsverfahren verwendet werden können.

Zusammengefasst kann man sagen, dass 3APL Möglichkeiten bietet die bereits grösstenteils in Jadex implementiert sind und somit keine wesentlichen Neuerungen bietet die interessant wären für eine weitere eingehendere Betrachtung.

4.4.2 Konstruktionsregeln für Ziele

In [TPH2002a] wird ein „Framework“ diskutiert, welches bei einer gegebenen Menge von Wünschen (Desires) durch die Anwendung von Regeln und Prioritä-

ten eine konsistente Menge von Zielen generieren soll. Ziel dieser Regeln ist es, die Anforderungen an die Menge der Ziele eines Agenten genau festzulegen. Insbesondere um die Konsistenz zwischen zwei Zielen ausdrücken zu können wird eine gewisse Repräsentation eines Ziel erwartet. Das hier beschrieben Verfahren ist wie das 3APL Verfahren ein Mischverfahren. Es berücksichtigt sowohl Regeln wie auch Prioritäten bzw. Präferenzen. Da aber die Grundlage dieses Verfahrens die Anwendung der Regeln ist wurde dieses Verfahren ebenfalls in den Abschnitt über Regelbasierte Verfahren integriert.

Repräsentation von Zielen

Der Kern eines Ziels wird definiert als Konjunktionen von grundlegenden Formeln. Grundlegende Formeln bestehen dabei aus Propositionen oder deren Negationen, sowie alternativ aus einfachen Beschränkungen. Einfache Beschränkungen sind wiederum aus Attributen, einer Relation und einer Zahl aufgebaut, wobei eine Relation aus der Menge $\{=, <, \leq, >, \geq\}$ ist. Ein Beispiel für eine Formel die alle genannten Bestandteile enthält ist: $p = buy \wedge \neg buy \wedge (Price \leq 100,000)$. Für zwei Zielzustände G_1, G_2 besteht genau dann eine Inkonsistenz wenn:

- $p \in G_1$ und $\neg p \in G_2$ für ein Teilziel p eines Zielszustandes oder
- Für ein Attribut a, welches in beiden Zielzustände enthalten ist, gibt es keine Lösung für die Vereinigung der Beschränkung, die dieses Attribut betrifft

Für die Beschreibung der Konsistenz wird das Prädikat $Consistent(\alpha, \beta)$ definiert. Dieses Prädikat ist wahr, wenn die Zustände α, β konsistent sind.
Gilt $\neg Consistent(\alpha, \beta)$, so stehen die Zustände im Konflikt zueinander und es wird dafür $Con(\alpha, \beta)$ definiert. Ebenso wird das Vorhandensein einer Prioritätsfunktion Pr angenommen. Diese Funktion gibt die Priorität eines Ziels durch einen numerischen Wert an. Es ist ein Ziel α, wichtiger als ein Ziel β, wenn $Pr(\alpha) > Pr(\beta)$ gilt.

Es wird vorerst angenommen, dass, wenn ein Ziel wichtiger als ein anderes Ziel ist, dieses wichtigere Ziel angenommen wird und das unwichtigere Ziel, falls es bereits bearbeitet wird, abgebrochen oder vorläufig suspendiert wird. Die Prädikate *Des* und *Goal* definieren für einen Zustand ϕ, dass es sich bei $Des(\phi)$ um einen Wunschzustand (Desire) und andererseits bei $Goal(\phi)$ um ein Zielzustand handelt. Ziele kann man als Wünsche betrachten, die einen gewissen Filter passiert haben. Es wird also eine Menge an Wünschen (Desires) betrachtet, die durch die im Anschluss definierten Regeln so beschränkt wird, dass daraus eine Menge von Zielen entsteht. Dabei betrachtet man auch die aktiven Ziele.

Es wird eine Ordnung zwischen Zielen wie folgendermaßen definiert. Für Zielzustände G_1, G_2 definiert man die Präferenz von G_1 über G_2 als $G_1 >> G_2$ wenn eine der folgenden Bedingungen zutrifft.

1. $\forall G \in G_2\ \exists G' \in G_1$ so dass $Pr(G') > Pr(G)$

2. $G_1 \supseteq G_2$

Das bedeutet zum einen, dass Zielzustände mit einem hoch priorisierten Teilziel, jenen mit vielen niedrig priorisierten Teilzielen, vorgezogen werden. Zum anderen reduziert sich die Präferenz eines Zielzustandes gegenüber einem anderen, wenn der erste Fall nicht zutrifft, auf Anzahl der Teilzustände eines Zielzustandes. In diesem Fall wird ein Zielzustand mit einer höheren Anzahl an Teilzuständen (Teilzielen) einem solchen mit einer niedrigeren Anzahl an Teilzuständen vorgezogen.

In einem nächsten Schritt werden Regeln definiert, denen Zielzustände entsprechen sollen.

$\mathbf{R_1}: \; Goal(\alpha) \wedge Goal(\beta) \rightarrow \neg Con(\alpha, \beta)$

Zwei Ziele α, β eines Agenten stehen in keinem Konflikt zueinander.

$\mathbf{R_2}: \; Des(\alpha) \wedge Des(\beta) \wedge \neg Imp(\alpha) \wedge Con(\alpha, \beta) \wedge (Pr(\alpha) > Pr(\beta)) \rightarrow$
$Goal(\alpha) \wedge \neg Goal(\beta)$

Für zwei Zielkandidaten gilt: Stehen diese beiden Kandidaten im Konflikt miteinander und ist der Kandidat α höher priorisert als der Kandidat β und wird der Kandidat α durch keinen anderen Zielzustand, der höher priorisiert ist und in Konflikt mit eben jenem Zustand α steht, verhindert, so wird der Zielzustand α als ein Ziel des Agenten angenommen.

$\mathbf{R_3}: \; Des(\alpha) \wedge Goal(\beta) \wedge Con(\alpha, \beta) \wedge (Pr(\alpha) = Pr(\beta)) \rightarrow \neg Goal(\alpha)$

Das bedeutet: Wenn ein Wunschzustand der die gleiche Priorität wie ein bereits aufgenommenes Ziel hat, in Konflikt mit diesem Ziel steht, so wird dieser Wunsch nicht als ein Ziel angenommen.

Diese drei Regeln spezifizieren wie man aus Wunschzuständen eines Agenten Ziele ableiten kann. Jedoch können diese Regeln unter Umständen zu einfach sein. Das bedeutet, sie erlauben keine erweiterte Betrachtung nach der Annahme eines Plans, zur Bearbeitung eines ausgewählten Ziels. So kann es z.B. passieren, dass bei der Abarbeitung eines Planes Zustände entstehen können, die in Konflikt mit den Zuständen aktiver Ziele stehen. Solche Konflikte wurden in 3.6.1 als sekundäre Zielkonflikte bezeichnet. Im Folgenden werden die vorhandenen Regeln erweitert und modifiziert, um dieser Feststellung entgegenzuwirken. Für die Vereinfachung, der im Folgenden aufgeführten Regeln, werden zusätzliche Prädikate spezifiziert.

Plan(P,G) bezeichnet, dass es einen Plan P für ein Ziel G gibt. Ex(P) bezeichnet, dass P der Plan zu einem G ausgeführt wird. Step(P,S) bezeichnet das S ein Schritt in der Ausführung eines Planes P ist. Pref(S1,S2) gibt an das der Schritt S1 dem Schritt S2 vorgezogen wird.

Wenn nun ein Schritt eines Planes mit einem Schritt eines anderen Plans in Konflikt steht, dann ist es ratsam einen anderen Plan zu suchen, dessen Schritte nicht mit den betreffenden Ziel in Konflikt stehen. Es werden zusätzlich die Relationen *ExStep* und *AltPlan* definiert:

ExStep: $Plan(P_1, \alpha) \wedge Ex(P_1) \wedge Step(\alpha, P_1, \phi) \rightarrow Goal(\alpha) \wedge ExStep(\alpha, P_1, \phi)$

Das bedeutet: Der Plan P_1 zur Bearbeitung von Ziel α wird gerade ausgeführt und ϕ ist ein Schritt während der Bearbeitung.

AltPlan: $\exists P : Plan(P, \alpha) \wedge (P \neq P_1) \wedge (\forall \gamma \psi : Step(\alpha, P, \gamma) \wedge Goal(\psi) \rightarrow \neg Con(\gamma, \psi))) \rightarrow AltPlan(\alpha, P_1)$

Das bedeutet: Zu dem Ziel α gibt es einen anderen Plan als P_1 der ebenfalls das Ziel α bearbeitet, aber deren Schritte nicht in Konflikt mit allen anderen Zielen ψ des Agenten stehen.

In der nächsten Regel werden Schritte und alternative Pläne für diese Schritte berücksichtigt, falls sie in Konflikt mit Zielen stehen.

$\mathbf{R_4}$: $ExStep(\alpha, P_1, \phi) \wedge ExStep(\beta, P_2, \phi') \wedge Con(\phi, \phi') \wedge AltPlan(\alpha, P_1) \wedge \neg AltPlan(\beta, P_2) \rightarrow Pref(\phi', \phi)$

Als nächstes wird eine Regel eingeführt, mit der man im Fall eines Konflikts eines Wunschzustands mit einem Schritt in einem Plan, den Wunschzustand vorziehen kann, wenn es einen alternativen Plan für das betreffende Ziel gibt.

$\mathbf{R_5}$: $ExStep(\alpha, P_1, \phi) \wedge Des(\phi') \wedge Con(\phi, \phi') \wedge AltPlan(\alpha, P_1) \rightarrow Pref(\phi', \phi)$

Jetzt werden die Regeln R_2, R_3 dahingehend modifiziert, Prioritäten nur anzuwenden, wenn das *AltPlan* Prädikat nicht anwendbar ist. Das passiert in den Fällen in denen keine alternativen Pläne oder für beide Zustände alternative Pläne zur Verfügung stehen und somit keine Präferenz im Sinne des Prädikates *Pref* ausgesprochen werden kann.

$\mathbf{R_2'}$: $Des(\alpha) \wedge Des(\beta) \wedge \neg Imp(\alpha) \wedge Con(\alpha, \beta) \wedge (Pr(\alpha) > Pr(\beta)) \wedge \neg Pref(\alpha, \beta) \wedge \neg Pref(\beta, \alpha) \rightarrow Goal(\alpha \wedge \neg Goal(\beta)$

$\mathbf{R_3'}$: $Des(\alpha) \wedge Goal(\beta) \wedge Con(\alpha, \beta) \wedge (Pr(\alpha) = Pr(\beta)) \wedge \neg Pref(\alpha, \beta) \wedge \neg Pref(\beta, \alpha) \rightarrow \neg Goal(\alpha)$

Abschliessend noch eine letzte Regel die für die Konsistenz zwischen Präferenz und der Annahme von Zielen sorgt.

$\mathbf{R_6}$: $Pref(S_1, S_2) \rightarrow Goal(S_1) \wedge \neg Goal(S_2)$

Die Schritte der Regel R_6 sind Subziele eines Plans.

Durch diese neuen Regeln wird die Präferenzrelation verfeinert. Zwei Zielzustände G_1, G_2 stehen im Verhältnis $G_1 >> G_2$ wenn eine der folgenden Bedingungen gilt:

1. $\forall G \in G_2\ \exists G' \in G_1$ so dass $Pref(G', G)$
2. $\forall G \in G_2\ \exists G' \in G_1$ so dass $Pr(G') > Pr(G)$
3. $G_1 \supseteq G_2$

Das bedeutet, dass ein Zielzustand G_1 einem Zielzustand G_2 vorgezogen wird, wenn alle Teilziele in G_2 alternative Pläne für ein Teilziel aus G_1 bieten können. Oder wenn das nicht zutrifft dann zurückfallen auf die alte Spezifikation der Präferenz zwischen Zielzuständen.

Durch die Verwendung dieser unterschiedlichen Regelmengen können unterschiedlich feine Strategien zur Deliberation auf Zielen herausgebildet werden. Die Regelmenge R_1, R_2, R_3 ist dabei eine gröbere Strategie gegenüber der Strategie die durch die Regelmenge $R_1, R_2', R_3', R_4, R_5, R_6$ gebildet wird. Weiter Regelmengen sind denkbar. So wird z.B. in [TPH2002a] vorgeschlagen, die Anforderung an alternative Pläne abzuschwächen. Schritte die durch alternative Pläne hervorgebracht werden, müssen nicht unbedingt konsistent zu allen anderen Zielen sein. Es könnte auch ausreichen, wenn diese Schritte nur konsistent zu Zielen bzw. Schritten mit gleicher oder höherer Priorität sind. Ebenso denkbar, aber doch relativ kompliziert, ist die Idee, dass Ziele fallengelassen werden können, wenn sie mit einer größeren Anzahl an Zielen in Konflikt stehen als ein anderes Ziel oder ein Zielkandidat.

Bewertung des Verfahrens

Das hier beschriebene Verfahren spezifiziert Regeln, die angewendet werden sollten, um eine konsistente Menge von Zielen, aus einer möglicherweise inkonsistenten Menge von Wünschen, zu generieren. Dieses bedeutet aber nicht zwangsläufig, dass dieses Verfahren auch in Form von Regeln implementiert werden muss. Grundsätzlich sollte jedoch die Aussage die hinter einer jeden Regel steht umgesetzt werden.

Basis Regelsatz: Der Spezifizierungsaufwand, der für dieses System geleistet werden muss, äußert sich in der Darstellung der Zustände, die betreffende Ziele beschreiben. Hier wird eine einfache eingeschränkte Logik verwendet, die verschiedenste Situationen beschreiben kann. In dieser Zustandsbeschreibung können, da sie sehr allgemein ist, wieder verschieden Situationen vermischt werden. So lassen sich dort Kontextbedingungen, Effekte aber auch Zustände die Situationen beschreiben in denen es um Ressourcen geht, darstellen. Dementsprechen unkonkret ist auch die Feststellung, ob zwei Zielkandidaten (Wünsche) konsistent sind. Wie bereits in der Bewertung zur „Interaction Summary“ dargelegt, kann eine Prüfung auf Kompatibilität (im Fall der „Interaction Summary“) oder wie hier auf Konsistenz so vielseitig und vor allem unterschiedlich sein, dass hier abermals für eine Eingrenzung der Art der Zustände, die spezifiziert und somit auf Konsistenz bzw. Kompatibilität überprüft werden, plädiert wird.

Jadex bietet für diesen Fall bereits eine komfortablere Möglichkeit. So kann in Jadex, durch die Spezifikation von den in Kapitel 4.2 vorgestellten Hemmungskanten, im gewissen Sinne Inkonsistenzen und Präferenzen nachgebildet werden. Hemmt ein Ziel a ein Ziel b, so kann man davon ausgehen, dass a mit b nicht konsistent ist. Beide Ziele sollen nicht gleichzeitig ausgeführt werden. Und

weil a, b hemmt hat a demnach eine höhere Priorität als b. Der Aufwand mit dem in Jadex spezifiziert werden muss, ist somit wesentlich geringer.

Erweiterter Regelsatz: Die Idee Zustände zu berücksichtigen, die von Plänen während ihrer Abarbeitung erwirkt werden und somit auch sekundäre Zielkonflikte zu erkennen, ist eine interessante Idee. Durch dieses zusätzliche Potential, erhält man eine größeren Entscheidungsspielraum, wenn entschieden werden soll, ob ein Zielkandidat (Wunsch) zum Ziel gemacht werden kann. Dies wurde auch bereits in der Bewertung zur „Summary" Information festgestellt.

Jadex bietet jedoch nicht die Berücksichtigung von sekundären Zielkonflikten. Deshalb sind die Anmerkungen die in diesem Verfahren und im Verfahren zu „Resource Summary" getätigt wurden, von so besonderem Interesse.

Zusammengefasst kann gesagt werden, dass das hier beschriebene Verfahren als Vorstufe zu den auf „Summary" Information basierenden Verfahren gesehen werden kann. Es erwähnt die Konsistenzprüfung auf Zielebene aber lässt dafür die konkreten Prüfungen auf Planebene noch recht im Dunkeln. Da sind im Teil zu „Summary" Informationen konkretere Ansätze sichtbar geworden (vgl. Kapitel 4.3.2).

4.5 Intensität

4.5.1 Alarms

Das Konzept der „Alarms" welches im Folgenden angesprochen werden wird, ist in eine der BDI-Architektur eng verwandte Architektur für motivierte Agenten eingebettet. Laut [NL96] ist diese Architektur, als eine Ergänzung zur BDI-Architektur zu sehen. Die Architektur für motivierte Agenten unterscheidet sich hauptsächlich in dem Konzept der Motive und der damit verbundenen Erzeugung von Zielen, von der originalen BDI-Architektur. Die Interna wie Deliberation und „Means-end Reasoning "sind aber in beiden Architekturen identisch und somit vergleichbar.

Die Rolle die „Alarms" in dieser Architektur spielen ist die einer Aktivierung von Zielen, was gleichbedeutend[4] mit dem Treffen einer Auswahl aus der Menge der vorhanden Ziele ist. Diese Rolle ist vergleichbar mit der Rolle der Deliberation im originalen BDI-Modell, die ebenfalls aus der Menge der zur Verfügung stehenden Optionen diejenigen auswählt, die der Agent bearbeiten will. In Abbildung 4.10 kann man die Unterschiede, aber auch die strukturellen Gemeinsamkeiten sehr gut erkennen. „Alarms" sind ein heuristisches Manage-

[4]in dem Sinn, dass aus der Menge der Ziele ausgewählt wird. Der Autor in [NL96] weist jedoch darauf hin, dass eine Äquivalenz zur Auswahl in der BDI-Architektur nicht direkt vorhanden ist. Dies ist damit begründet, dass die Auswahl auf die sich der Autor bezieht in einem zweiten Schritt nach der Aktivierung der Ziele, also innerhalb des Planers, geschieht. Eine Auswahl anhand einer Aktivierung wie in [NL96] ist in der originalen BDI-Architektur nicht vorhanden. Deshalb wird hier die Auswahl als eine Teilauswahl interpretiert und als eine Möglichkeit zwischen Zielen zu wählen.

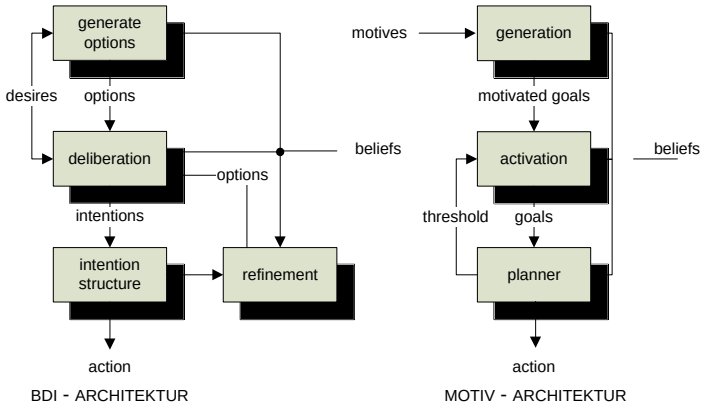

Abbildung 4.10: Vergleich BDI-Architektur - Architektur für motivierte Agenten, Einzelabbildungen aus [NL96]

mentverfahren für die Ziele eines Agenten. Sie führen, durch ihre Definition und dem zusätzlichen Vorhandensein eines Schwellwertes, zu einer Fokussierung auf die im Augenblick am dringendsten zu bearbeitende Ziele und begünstigen demnach die Auswahl zwischen den Zielen eines Agenten.

Ein „Alarm“ α [NL96] ist eine Datenstruktur, die einem Ziel g eine Intensität zuordnet. Diese Intentsität ist eine Funktion des Zustandes der Domäne. Das Ziel g, welchem diese Intensität zu geordnet wird, besteht aus einer Proposition p, einer Wichtigkeit i_{max} und einem zeitlichen Kontext in dem das Ziel g erfüllt werden soll. Dieser zeitlicher Kontext, setzt sich aus den Zeiten t_{dt}, t_{dl} und aus dem Zeitintervall $\Delta_a t$ zusammen. Dabei steht t_{dt} für den Zeitpunkt vor dem die Bearbeitung des Ziels g nicht gestartet werden soll, $\Delta_a t$ für den Zeitraum der wahrscheinlich für die Bearbeitung des Ziels g benötigt wird und t_{dl} für den Zeitpunkt ab dem der Agent will, dass das Ziel g erfolgreich bearbeitet wurde. Somit lässt sich ein maximaler Zeitpunkt ermitteln, ab dem die Bearbeitung des Ziels letztmöglich begonnen werden kann. Dieser Zeitpunkt wird mit t_{max} bezeichnet und als: $t_{max} = t_{dl} - \Delta_a t$ definiert. Anhand dieser Tatsachen kann man schlussfolgern, dass jedes Ziel in dieser Architektur ein begrenztes Leben hat.

Die Wichtigkeit i_{max} und der zeitliche Kontext eines Ziels g werden dazu verwendet, um eine Intensität des betreffenden Ziels zu berechnen. Die Berechnung erfolgt durch eine „Alarm“ Funktion $f(t)$ welche wie folgt definiert ist.

Definition: Eine „Alarm“ Funktion $f(t)$ eines „Alarm“ α ist definiert als 0 vor t_{dt}, wachsend in dem Zeitraum von t_{dt} nach t_{max} und maximal ab t_{max}. Formal bedeutet das:

$$f(t) = \begin{cases} 0 & : \textbf{if } before(t, t_{dt}) \\ i_{max} & : \textbf{if } \neg before(t, t_{max}) \\ \frac{t - t_{dt}}{t_{max} - t_{dt}} & : \textbf{otherwise} \end{cases}$$

mit $t_{max} = t_{dl} - \Delta_a t$

Zu jeder Zeit nach dem Festlegen eines „Alarms“ für ein Ziel g, können die aktuelle Situation, Veränderungen von für diesen „Alarm “ getroffenen Bedingungen oder die Auswirkungen der Planung für zu bearbeitende aktive Ziele, die Intensität des Ziels g beeinflussen. Im Folgenden sollen diese Einflüsse näher beschrieben werden.

Gelegenheiten (opportunities) für Ziele

So können für einen „Alarm“ α verschiedene sogenannte „opportunities“ spezifiziert werden. Diese „opportunities“ [NL96] sind Aktionen mit Vor- und Nachbedingungen. Aufgrund der Tatsache, dass die Nachbedingungen der „opportunities“ , der Proposition p des im „alarm“spezifizierten Ziel entsprechen, können diese Aktionen als Möglichkeiten einer erfolgreichen Bearbeitung des betreffenden Ziels gesehen werden. Diese erfolgreiche Bearbeitung ist aber nur in den speziellen, in den Vorbedingungen der „opportunities“ angegebenen Situation, anwendbar.

Der Zweck solcher „opportunities“ ist das Erkennen und Ausnutzen von bestimmten Situationen, die eine erfolgreiche Bearbeitung der Ziele des Agenten begünstigen. Sollte durch den Agenten festgestellt werden, dass eine „opportunity“ in der vorliegenden Situation zutreffend ist und die aktuelle Zeit $t_{now} \geq t_{dt}$, so resultiert das in dem Effekt eines Intentsitätimpulses. Dieser Impuls führt dazu, dass der Intensitätswert des betreffenden „Alarm“ augenblicklich auf seinen Höchstwert i_{max} schnellt. Der Wert der Alarmfunktion ist in diesem Fall nicht mehr ausschlaggebend. Damit wird es für das Ziel welches durch diesen „Alarm“ gekapselt wird einfacher in den Fokus der Betrachtung des Agenten zu gelangen und somit ausgeführt zu werden. Es ist zu beachten, dass „Opportunities“ für Ziele mit einer hohen i_{max} eine höhere Wirkung besitzen als für Ziele die ein niedriges i_{max} besitzen.

Gefahren (Dangers) für Ziele

Eine weitere Möglichkeit die eine Modifizierung eines „Alarm“ notwendig macht ist die Veränderung des Zeitintervall $\Delta_a t$. Solch eine Veränderung kann durch andere Agenten, durch Operationen des betreffenden Agenten selbst oder durch allgemeine Veränderungen der Umwelt entstehen. Die gerade beschriebenen Ursachen solch einer Änderung werden in [NL96] als sogenannte „Dangers“ bezeichnet. Die Handhabung solcher „Dangers“ soll im Folgenden erläutert werden.

Bei der Erstellung eines „Alarms“ für ein Ziel, trifft der Agent hinsichtlich der Bestimmung von $\Delta_a t$ verschiedene Annahmen. Diese Annahmen stellen Zustände der Welt dar, die für eine erfolgreiche Bearbeitung des Ziels innerhalb $\Delta_a t$ garantieren. Angegeben werden diese Annahmen während der Erstellungsphase des „Alarms“, durch sogenannte „angemessene Bedingungen“(Appropriateness Conditions). Diese werden als eine Liste von Tupeln der Form $\{p, \Delta t\}$ mit p als Proposition und Δt als ein Zeitintervall beschrieben.

Wird durch eine Aktion bzw. einen Plan eines Agenten solch eine Proposition negiert, so gibt die Zeitspanne hinter dieser Proposition an wie lange der Agent benötigt um diese Negation wieder rückgängig zu machen, das heißt die ursprüngliche Annahme wieder herzustellen. Wurde z.B. bei der Bestimmung von $\Delta_a t$ eines „Alarms" Annahmen über den Aufenthaltsort des Agenten getroffen, so muss bei einer Veränderung des Aufenthaltsorts des Agenten logischerweise diese Veränderung und die daraus resultierende Veränderung der Zeit, die für eine Bearbeitung des betreffenden Ziels benötigt wird, berücksichtigt werden. Der zeitliche Rahmen Δt der jetzt zusätzlich benötigt wird, muss entsprechend bei der Bestimmung des neuen $\Delta_a t$ berücksichtigt werden.

Wurde eine solche Annahme negiert, so ist es notwendig sie wieder rückgängig zu machen. Ist der Zeitpunkt t nach einer solchen Wiederherstellung $t \leq t_{max}$, so besteht kein Grund zur Korrektur des Intervalls $\Delta_a t$. Das Ziel mit dem entsprechendem korrigiertem „Alarm" hat weiterhin die Möglichkeit in der Zeit $\Delta_a t$ ausgeführt zu werden. Sollte nun aber durch widrige Umstände die Möglichkeit der Rückstellung der Annahme erst ab einem Zeitpunkt t_x beginnen können, für den gilt: $t_x + \Delta t > t_{max}$. So besteht die Gefahr, dass das Ziel nicht mehr rechtzeitig ausgeführt werden kann. Es muss dann laut [NL96] das Ziel welches beeinträchtigt werden würde, einen entsprechenden Zeitraum vor t_{max} aktiviert werden um den Konflikt zwischen den beiden Zielen zu erkennen und ihn aufzulösen. Dies wird folgendermaßen gelöst.

Die maximale Intensität i_{max} des betreffenden Ziels wird durch das Vorverlegen des t_{max} Zeitpunktes entsprechend früher erreicht. Wie genau das berechnet wird, soll im Folgenden erläutert werden. Es wird angenommen das während der Ausführung eines Planes π eine Bedingung p_{pcs} (sogenannter Preparatory Effect) bis zu einem bestimmten Zeitpunkt t_{pcs} sichergestellt werden muss. Desweiteren wird angenommen, dass p_{pcs} mit der „j'ten" „Appropriateness Condition" $(p_j, \Delta_j t)$ in Konflikt steht und zeitlich mit dem Intervall $[(t_{max} - \Delta_j t), t_{dl}]$ überlappt. Dann wird der „Alarm" folgendermaßen verschoben. Zusätzlich werden definiert: $a :=$ Aktion des Planes $\pi, duration(a) :=$ Dauer der Aktion a , $end(a) :=$ Zeitpunkt vor der die Aktion erfolgreich abgearbeitet ist. Die Vorbedingung für die Anwendung der folgenden Gleichung ist:

$$A_1 := (a, p_{pcs}, t_{tpcs}) \in pcs(\pi) \wedge (p_j, \Delta_j t) \in app(\alpha)$$
$$B_2 := (conflict(p_j, p_{pcs}) \wedge \neg(before(t_{pcs}, (t_{max} - \Delta_j t))$$
$$\text{Gesamtbedingung} := (A_1 \wedge B_2) \vee before(t_{dl}, (end(a) - duration(a))$$

$$\Rightarrow \Delta d \rightarrow t = \begin{cases} (t_{dl} - end(a)) + duration(a) & : \textbf{if } before(end(a), t_{dl}) \\ duration(a) - (end(a) - t_{dl}) & : \textbf{if } before(t_{dl}, end(a)) \\ duration(a) & : \textbf{otherwise} \end{cases}$$

Diese Gleichung gilt sowohl für beabsichtigte Bedingungen (Effekte) p_{pcs}wie auch für unbeabsichtigte Bedingungen (Seiteneffekte) a. Für Seiteneffekte gilt die Vorbedingung:

$$A_2 := a \in acts(\pi) \wedge p \in post(a) \wedge (p_j, \Delta_j t) \in app(\alpha)$$
$$B_2 := conflict(p_j, p) \wedge \neg(before(end(a), (t_{max} - \Delta_j t))$$

$$\text{Gesamtbedingung} := (A_2 \wedge B_2) \vee before(t_{dl}, (end(a) - duration(a))$$

Werden dagegen Veränderungen von Seiten anderer Agenten oder Prozesse der Umwelt erwirkt, so hat der Agent nicht die Möglichkeit im Voraus festzustellen ob diese Veränderungen zeitlich so ungünstig liegen, dass sie die Bearbeitung des Ziels innerhalb von $\Delta_a t$ unmöglich wird. Dies bedeutet, dass die Annahmen die bei der Berechnung von $\Delta_a t$ zu Grunde lagen, nicht nur durch den Agenten selbst negiert werden. Die Möglichkeit die dem Agent mit veränderter Annahme jetzt bleibt, ist die „Alarm" Funktion so zu verschieben, dass noch genügend Zeit verbleibt die Negation zurückzustellen. Es gilt:
Wenn eine Proposition in Konflikt mit der „j'ten" angemessenen Bedingung eines „Alarms" α steht, dann sollte der „Alarm" so verschoben werden so dass er zum Zeitpunkt $t_{max} - \Delta_j t$ bereits sein Maximum erreicht.

$$p \in \mathcal{D} \wedge (p_j, \Delta_j t) \in apps(\alpha) \wedge conflict(p_j, p) \Rightarrow \Delta_{d\rightarrow} t = \Delta_j t$$

Zeitliche Verpflichtungen für Ziele

Eine andere Möglichkeit wie „Alarms" negativ beeinflusst werden können, äußert sich folgendermaßen. Für das Bearbeiten eines Ziels erstellt oder besitzt der Agent einen Plan. Dieser Plan besitzt, im Kontext der Anwendung von „Alarms", einen minimalen zeitlichen Rahmen in denen all seine Aktionen abgearbeitet sein müssen. Es handelt sich hierbei um das Intervall $[t_{max}, t_{dl}]$. So ist sichergestellt, dass auch bei einem Start von spätestens t_{max} das betreffende Ziel vor t_{dl} erfolgreich bearbeitet ist. Sollte nun aber das Intervall $[t_{max}, t_{dl}]$ des einen Ziels das betreffende Intervall eines zur Abarbeitung vorgesehenen anderen Ziels überlappen, so besteht ein zeitlicher Konflikt zwischen der Bearbeitung der beiden Ziele. Die komplette Zeit wird für die Bearbeitung von einem Ziel benötigt. Wie kann diesem Konflikt begegnet werden? Indem die Abarbeitung einer der Pläne nicht auf den letztmöglichen Zeitpunkt t_{max} gelegt wird, sondern bereits früher begonnen wird das Ziel und somit den Plan für dieses Ziel zu bearbeiten, besteht ein zeitlicher Spielraum, der z.B. für die Bearbeitung von anderen Plänen genutzt werden kann. Dies geschieht wie im Fall der sogenannten „Dangers" durch einen Verschiebung des „Alarms" auf einen entsprechenden Zeitpunkt vor t_{max}. Bei dieser Beeinflussung durch zeitliche Verpflichtungen handelt es sich also genau um das Gegenteil zur Beeinflussung durch „Alarms" gefährdende Situationen (Dangers). Hier werden nicht die angemessenen Bedingungen negiert und somit eine Vergrößerung von $\Delta_a t$ notwendig, sondern die Zeit zum Bearbeiten des Ziels während des Intervalls $[t_{max}, t_{dl}]$ fehlt. Für die Bestimmung der Verschiebung des „Alarms" in so einem Fall ist in [NL96] folgende Berechnung vorgesehen. Zu beachten ist, dass solch eine Verschiebung wiederum in einem zeitlichen Konflikt resultieren kann. Deswegen wird die im Folgenden angegebene Berechnung solange wiederholt bis das Intervall $[(t_{max} - \Delta_{tc\rightarrow} t), (t_{dl} - \Delta_{tc\rightarrow} t)]$ mit keiner zeitlichen Verpflichtung $(\Delta_{tc} t, t_{tc}) \in tcs(\pi)$ überlappt. $tcs(\pi)$ bedeutet dabei die Zeitverpflichtung des Planes π.

Eine zeitliche Verpflichtung $\Delta_{tc} t, t_{tc}$ steht im Konflikt mit einem „Alarm" wenn das Intervall $[(t_{tc} - \Delta_{tc} t), t_{tc}]$ das Intervall $[t_{max}, t_{dl}]$ überschneidet. In

diesem Fall muss der „Alarm“ derart verschoben werden, dass er sein Maximum zu einer entsprechenden Zeit vor $t_{tc} - \Delta_{tc}t$ erreicht. Der „Alarm“ wird um $\Delta_{\rightarrow t}t$ verschoben. Es gilt für die folgende Gleichung folgende Vorbedingung:

$$\Delta_{tc}t, t_{tc} \in tcs(\pi) \wedge \neg before(t_{tc}, t_{max}) \vee before((t_{tc} - \Delta_{tc}t), t_{dl})$$

$$\Rightarrow \Delta tc \rightarrow t = \begin{cases} (t_{dl} - t_{tc}) + \Delta_{tc}t & : \textbf{if } before(t_{tc}, t_{dl}) \\ \Delta_{tc}t - (t_{tc} - t_{dl}) & : \textbf{if } before(t_{dl}, t_{tc}) \\ \Delta_{tc}t & : \textbf{otherwise} \end{cases}$$

Schwellwert

Was bis jetzt noch nicht angesprochen wurde, aber unmittelbar mit der Aktivierung von Zielen zu tun hat, ist die Verwendung eines Schwellwertes (threshold). Mit diesem Schwellwert kann die Häufigkeit der Aktivierung der Ziele durch „Alarms“ gesteuert werden. Ist der Schwellwert niedrig, so gelangen sowohl Ziele mit einer relativ niedrigen Intensität, wie aber auch Ziele mit einer relativ hohen Intensität in den Fokus des Agenten. Ist aber der Schwellwert hoch angesetzt, so gelangen nur Ziele mit einer für diesen Schwellwert relativ hohen Intensität in den Fokus des Agenten. Dies kann den Vorteil haben, das wenn ein Agent ein wichtiges Ziel bearbeitet nicht durch das Aktivieren eines weniger wichtigen Ziels gestört wird. Der Schwellwert repräsentiert die Sensitivität des Agenten neue Ziele in Betracht zu ziehen. Ist der Schwellwert hoch ist der Agent unempfindlicher, ist der Schwellwert niedrig ist der Agent sensitiver.

Das Setzen eines adäquaten Schwellwertes ist unabdingbar für das richtige funktionieren des Verfahrens basierend auf „Alarms“. Er drückt die momentane kognitive Belastung des Agenten aus und regelt somit die Aktivierungsrate von „Alarms“. Für den Schwellwert gilt: alle Ziele deren Intensität $f(t)$ größer oder gleich dem Schwellwert sind, werden vom Agenten aktiviert und einer weiteren Bearbeitung zugeführt. Unter diesen dann aktivierten Zielen kann der Agent noch weiter auswählen z.B. anhand der Wichtigkeit. Diese Auswahl nach z.B. Wichtigkeit ist im Zusammenhang mit der in der Einleitung zu diesem Verfahren gesetzten Fußnote, als die Zweitauswahl zusehen. Die Erstauswahl bezieht sich auf die Intensität.

Die geeignete Spezifizierung eines Schwellwertes hat also Auswirkung auf die Aktivierung von Zielen bereits vor t_{max}. Allgemein kann festgehalten werden, dass bei der Bearbeitung von wichtigen Zielen der Schwellwert entsprechend hoch angesetzt werden sollte um eine Ausführung dieser nicht zu stören. Und bei der Bearbeitung von unwichtigeren Zielen sollte der Schwellwert niedriger angesetzt werden, um relativ zu diesen Zielen wichtigere Ziele entgegen nehmen zu können. In [Nor97] ist eine sehr aufwendige Berechnung für die Bestimmung des Schwellwertes angegeben. Diese berücksichtigt die Dauer der Ausführung der einzelnen Aktionen. Das hat Auswirkungen auf die geistige Beanspruchung des Agenten (cognitive load), was wiederum der Höhe des Schwellwertes bestimmt. Diese Berechnung soll hier nicht aufgeführt werden. Vielmehr soll ein Verständnis für das Einstellen des Schwellwertes vermittelt werden.

Wurde schließlich ein „Alarm“ ausgelöst, das bedeutet die Intensität des betreffenden Ziels überstieg den Schwellwert, so muss entschieden werden, was mit diesem aktivierten Ziel passieren soll. Dem Agenten stehen folgende Möglichkeiten zur Auswahl bereit:

1. Aktivierung des Ziels, Löschung des „Alarms“
2. Löschung des Alarms, ohne Aktivierung des Ziels
3. Verminderung des Alarms

1.Fall: Falls der Agent entscheidet das Ziel zu bearbeiten, so aktiviert er es und deaktiviert damit zugleich den dazugehörigen „Alarm“. Die Auswahl ob aktiviert werden soll oder nicht kann dabei durch folgenden Prinzipien geleitet werden.

- Versuche soviel Ziele wie möglich zu bearbeiten,
- Besteht ausreichend Zeit dieses Ziel zu bearbeiten?
- Ein Ziel mit hoher Priorität hat Vorrang vor Zielen mit niedriger Priorität.

2.Fall: Falls der Agent feststellt, dass das Ziel bereits erfüllt ist, dass das Ziel nicht erfüllt werden kann oder dass der Agent das Ziel nicht mehr erfüllen will, so löscht er den „Alarm“ zum dazugehörigen Ziel. Das Ziel wird damit unbedeutend für den Agenten.

3.Fall: Wird der „Alarm“ für ein Ziel ausgelöst und entscheidet der Agent momentan nicht dieses Ziel zu bearbeiten, so wird wenn der „Alarm“ nicht gelöscht wird, d.h. das Ziel ist weiterhin relevant, die Intensität des „Alarms“ vermindert. Die Intensität des „Alarms“ steigt danach wieder linear an, so das sie spätestens ab t_{max}, i_{max} erreicht wird. Natürlich ist es auch möglich, dass der Verlauf der Intensitätskurve vor t_{max} sein Maximum erreicht und somit dann zu Disposition steht. Sollte das Ziel dann abermals nicht aktiviert werden, so beginnt der Vorgang der Verminderung von neuem. Für die Berechnung des Verlaufs der Intensitätskurve nach einer Verminderung ist in [Nor97] eine Formel angegeben, auf die hier nicht näher eingegangen wird.

Diese Verminderung der Intensität wird in die Eingangs erwähnte Formel zur Berechnung der Intensität eines Ziels integriert. Daraus folgt dann für die Bestimmung der Intensität eines Ziels.

$$intensity(\alpha, t_{now}) = \begin{cases} t_{max} & : \textbf{if } opportunities(\alpha) \neq \emptyset \\ f(t_{eval}) + m(t_{eval}) & : \textbf{otherwise} \end{cases}$$

mit $t_{eval} = t_{now} - \Delta_{d\rightarrow}t - \Delta_{tc\rightarrow}t$

Besitzt der Agent nur eine geringe Menge an Zielen die ohne Beeinträchtigung seiner Leistung bearbeiten kann, so bietet sich die Verwendung von „Alarms“ nicht an. Jedoch ist die Verwendung der „Alarms“ mit so einem geringen „Overhead“ belastet, dass sich dies praktisch kaum spürbar auswirken würde.

Bewertung des Verfahrens

Das hier beschriebene Verfahren, berücksichtigt die Verwendung der speziellen Ressource Zeit und stellt somit an alle Ziele die in diesem Verfahren verwendet werden temporale Anforderungen. Das Verfahren berücksichtigt keine Interaktionen die zwischen den Propositionen p der verwendeten Ziele auftreten können. Damit das Verfahren sein Potential entfalten kann, muss für jedes Ziel der zeitliche Kontext spezifiziert werden in dem es bearbeitet werden soll. So muss der Zeitpunkt angegeben werden vor dem das Ziel nicht aktiviert werden soll (t_{dt}), der Zeitpunkt an dem das Ziel erfolgreich bearbeitet wurde (t_{dl}) und die Dauer ($\Delta_a t$) der Bearbeitung. Die Spezifizierung dieser Werte stellt ein gewisses Problem dar. Es gibt keine universell richtige Darstellung von Zeit. Die Darstellung orientiert sich an der Anwendungsdomäne in der die Repräsentation von Zeit benötigt wird. Dies stellt ein Indiz contra einen generischen Charakter dieses Verfahrens dar. Desweiteren ist nicht von der Hand zu weisen, dass auch die Einschätzung der Zeit, die der Agent zur Bearbeitung eines Ziels benötigt eine besondere Anstrengung erfordert. Wie soll ein Entwickler festlegen können wie lange Ziel X dauert. Soll er den Durchschnitt berechnen oder soll er die Situation heranziehen, die er als die wahrscheinlichste empfindet in der sich der Agent zum Start der Bearbeitung des Ziels befindet. Dies sind alles Fragen die sehr von der speziellen Situation in einer speziellen Domäne abhängen. Es ist somit davon auszugehen, dass eine Spezifizierung der temporalen Information nicht trivial ist und ein erhebliches Verständnis der möglichen Zustände, die das System einnehmen kann, erfordert.

Dangers: Ähnliche Schwierigkeiten wie in der Spezifizierung der Dauer eines Ziels sind in der Spezifizierung der sogenannten angemessenen Bedingungen (appropriate conditions) auszumachen. Diese Bedingungen sollen Situationen und eine Dauer angeben, die, wenn sie verletzt sind, auf die Gesamtdauer des betreffenden Ziels addiert werden müssen. Bei einer relativ komplexen Anwendung ist anzunehmen das es schwierig werden wird alle relevanten Situationen zu erfassen, in denen sich die Dauer der Bearbeitung des Ziels verändern könnte. Hier kann damit gerechnet werden, dass ein erheblicher Spezifizierungsaufwand erforderlich ist.

Zeitliche Verpflichtungen: Auch die Spezifizierung des zeitlichen Rahmen in denen die Pläne zu den Zielen aktiv werden ist aufwendig. Sie dient der Erkennung von möglichen zeitlichen Konflikten um so t_{max} des „Alarms“ zu verschieben, erhöht aber den schon starken Spezifizierungsaufwand nur noch mehr.

Zusammengefasst kann gesagt werden, dass dieses hier beschriebene Verfahren nur als Teil eines Gesamtverfahrens angesehen werden kann. Es fordert von dem Entwickler eines Agenten erstens einen sehr hohen Spezifizierungsaufwand, sowohl in der Menge wie auch in der Qualität der Informationen ab und zweitens ist die Entscheidungsmöglichkeit die man aus diesem Verfahren gewinnt

nur auf „bearbeite Ziel“ und „bearbeite Ziel noch nicht“ beschränkt. Weitere Interaktionen werden nicht berücksichtigt.

4.6 Nutzen

4.6.1 JAM

Die Agentenarchitektur JAM [Hub99] lässt Anzeichen eines auf Nutzen basierenden Deliberationsverfahren erkennen. JAM ist als eine Weiterentwicklung des „Procedural Reasoning System“ (PRS) aus Kapitel 2.3 zu sehen.

Die Ziele eines Agenten in Jam spezifizieren neben dem Typ und einem Bezeichner auch ein optionales Nutzen (utility) Attribut. Dieses Attribut kann dazu verwendet werden, um einen numerischen Wert, explizit durch Angabe einer Zahl oder implizit durch Berechnung einer Funktion, anzugeben, der einen Nutzen für das betreffende Ziel spezifiziert. Ebenso besitzen auch die Pläne eines JAM Agenten ein Attribut Nutzen. Dieses Attribut kann, wie das entsprechende Attribut eines Ziels, explizit oder implizit spezifiziert werden. JAM bietet somit neben einem „Meta-level Reasoning“ auch ein entsprechendes „Utility-based Reasoning“ an. Das „Utilityattribut“ hat einen Standardwert von 0.0 wenn explizit nichts anderes angegeben wird.

Die Bearbeitung von Zielen, auf der Grundlage eines „Utility-based Reasoning“ läuft folgendermaßen ab. Für ein jedes Ziel prüft der Agent, ob Pläne für dessen Bearbeitung anwendbar sind. Speziell bedeutet das, die Kontext- und Vorbedingung eines jeden Plans, der das zu prüfende Ziel spezifiziert, zu verifizieren. Die anwendbaren Ziele werden in der Datenstruktur „Applicable Plan List “(APL) gespeichert. Der Wert des Nutzenattributs eines jeden in der APL vorhanden Plan, wird berechnet. Derjenige Plan mit dem höchsten Wert wird zur Bearbeitung für das betreffende Ziel ausgewählt und dem betreffenden Ziel zugeordnet. Dieser Vorgang bildet eine Intention des Agenten bezüglich des Ziels. Bei der Zuordnung des ausgewählten Plans zu seinem Ziel, wird der Wert des Nutzenattributs des betreffenden Ziels mit dem Wert des Nutzenattributs des Plans kombiniert. Dies geschieht hier durch einfache Anwendung der Addition. Der Nutzen eines Ziels (Intention) setzt sich also aus dem im Attribut des Ziels spezifizierten Wert plus dem Wert des ausgewählten Plans zusammen. Für Subziele die innerhalb eines Plans erzeugt werden können, wiederholt sich dieser Vorgang. Dies impliziert, dass ein Subziel eines sehr nützlichen Ziels nicht unbedingt auch ein sehr nützliches Ziel sein muss.

Zur Bestimmung auf welches Ziel, unter den Zielen für die der Agent Intentionen gebildet, die Auswahl fällt, werden die Werten der Nutzenattribute untereinander verglichen. Dasjenige Ziel welches den höchsten Wert aufweist, wird zur Bearbeitung ausgewählt und somit der dazugehörige Plan ausgeführt.

Der Nutzen einer Intention eines Agenten kann sich z.B. durch die Bildung von Subzielen verändern. Daraus folgt, dass der Agent, wenn die gerade bearbeitete Intention nicht mehr den höchsten Nutzwert besitzt, zu einem anderen Ziel mit höherem Wert wechselt. Das bedeutet zum Einen wiederum, dass der Agent

jeweils nur die „Blätter“ eines „Intentionthreads“ bei seiner Auswahl berücksichtigt und zum Anderen, so der Anspruch auf einen den Nutzen maximierenden Agenten erhoben werden kann. Pläne Subziele bilden werden nicht mehr in der Auswahl des zu verfolgenden Ziels berücksichtigt. So kann es vorkommen, dass die Bearbeitung eines Ziel vorübergehend beendet wird und eine anderes Ziel zur Bearbeitung ausgewählt wird.

Bewertung des Verfahrens

Spezifikation JAM verwendet zur Entscheidung welche Ziele bzw. welche Pläne bearbeitet werden den Begriff Nutzen. Dieser Nutzen kann in numerischer Form angegeben oder durch einen Ausdruck berechnet werden. Der Aufwand der bei dieser Spezifizierung erbracht werden muss, ist vergleichsweise gering. So reicht beispielsweise die Spezifikation einer einfachen Ordnung (numerisch) auf den Zielen bzw. Plänen, um dem System eine Möglichkeit der Entscheidung anzubieten.

Jedoch lässt sich darüber streiten, ob die getrennte Spezifizierung von Nutzen für Pläne und Ziele zu ungewünschten Verhalten führen kann. Den Nutzen den man für einen Plan spezifiziert, stellt den Nutzen dar, den ein Plan in einer gegebenen Situation, gegenüber anderen für dieses Ziel spezifizierten Plänen, darstellt. Der Nutzen eines Ziels, stellt den Nutzen dar, den der Agent von der erfolgreichen Bearbeitung dieses Ziels hat. Hat nun aber ein Plan für ein weniger nützliches Ziel, einen hohen Nutzen für dieses Ziel, so kann die Entscheidung welches Ziel bearbeitet werden sollte, ungünstig beeinflusst werden, indem womöglich das weniger nützliche Ziel bearbeitet wird. Das würde eine rationale Verhaltensweise des Agenten konterkarieren.

Entscheidung JAM entscheidet grundsätzlich ohne Vorliegen eines Grundes, wie z.B. eines Konflikts, nach dem Nutzen eines Ziels. Dabei werden Randbedingungen wie Vorbedingungen und Kontextbedingungen nicht berücksichtigt. Dies stellt einen Ansatzpunkt für Kritik an diesem Verfahren dar, der durch den Umgang mit zwischenzeitlich suspendierten Zielen begründet werden kann.

Wenn aufgrund der Tatsache, dass ein anderes Ziel einen höheren Nutzen hat, als das augenblickliche Ziel und somit die Bearbeitung des einen Ziels suspendiert wird und die Bearbeitung des anderen Ziels aufgenommen wird, so wird nicht beachtet, dass die Bearbeitung des neuen Ziels den Kontext der für die Bearbeitung des anderen Ziels notwendig ist, verändern kann. Das kann zur Folge haben, dass das suspendierte Ziel, wenn es wieder zur Bearbeitung ausgewählt wird, nicht mehr weiter bearbeitet werden kann, da der notwendige Kontext verloren gegangen ist. Im schlimmsten Fall ist eine Bedingung eingetreten, die sogar die Aufgabe des betreffenden Ziels notwendig macht. Dies wird so vom Autor bewusst hingenommen. Der Autor macht anschließend geltend, dass durch die Fokussierung auf das Ziel mit dem höchsten Nutzen, ein den Nutzen maximierender Agent entstanden ist, welches rationalen Gesichtspunkten des Autors genügt. Dem kann so nicht ganz zugestimmt werden.

Auch berücksichtigt das Verfahren keine Hierarchien zwischen Zielen. Zwar existieren Ziele und Subziele, aber hinsichtlich der Auswahl durch den Nutzen können Subziele eines Ziels den Wert des Ziels vermindern mit der Gefahr, dass aufgrund dieser Tatsache eine Ausführung eines anderen Ziels bevorzugt wird.

Abschließend kann man sagen, dass in [Hub99] Ansätze einer Möglichkeit eines Entscheidungskriterium aufgeführt sind, aber weitergehende Betrachtungen nicht ausgeführt wurden. Das hier beschriebene, auf Nutzen basierende Verfahren ist zu grob, als das es die vielfältigen Interaktionen zwischen Zielen und Plänen in einem einzigen Attribut zusammenführen könnte. Es könnte als Bestandteil eines Systems verwendet werden, um Aspekte des Nutzen von Zielen zu spezifizieren. Aber als alleiniges Verfahren für das System Agent ist es zu grob und zu einfach. Es ist ebenso hervorzuheben, dass eine Entscheidung nach dem Nutzen eines Ziels im allgemeinen identisch mit der Entscheidung nach der Priorität ist. Dies wird vor allem durch folgendes Zitat aus [Hub2001] deutlich:

> „As the utilities of the various goals change as the situation changes, the agent will switch between goals in order to continually pursue the highest-priority goal." [Hub2001]

Dort wird die Entscheidung nach Nutzen mit der Entscheidung das am höchsten priorisierte Ziel zu bearbeiten gleichgesetzt.

4.7 Priorität

4.7.1 InterRap

Interrap ist eine hybride, geschichtete Architektur für intelligente Agenten [MP93]. Hybrid bedeutet grob gesagt, dass diese Architektur sowohl reaktive wie auch proaktive Komponenten in sich vereint. Interrap bietet einem auf dieser Architektur basierenden Agenten, eine Entscheidungskomponente, bezüglich der Auswahl zwischen aktiven Zielen an. Diese Entscheidungskomponente trifft eine Auswahl aufgrund einer gewissen Strukturierung der Ziele. Diese Strukturierung wird im Folgenden der entscheidende Fokus der Betrachtungen sein.

Die Ziele eines Agenten werden in einer zwei dimensionalen, hierarchischen Struktur zusammengefasst. Die erste Dimension priorisiert Ziele, während die zweite Dimension Verfeinerungen der jeweiligen Ziele darstellt.

Prioritäten von Zielen

Zwecks einer Entscheidung wird mit jedem aktiven Ziel eine Priorität assoziiert. Diese Priorität berechnet sich aufgrund einer Prioritätsfunktion: $P : \mathcal{G} \rightarrow \mathcal{N}$ mit $\mathcal{G}$ als der Menge der Ziele und $\mathcal{N}$ als die Menge der natürlichen Zahlen. Dasjenige Ziel mit der höchsten Priorität wird im nächsten Interpreterzyklus bearbeitet.

Die Berechnung der Priorität eines Ziels setzt sich dabei aus der Berücksichtigung einer statischen und einer dynamischen Priorität zusammen.

Statische Priorität

Zur Berechnung einer statischen Priorität wird in [MP93] auf ein Modell des Psychologen Maslow zurückgegriffen. In diesem Modell werden menschliche Bedürfnisse in fünf Ebenen unterteilt. Diese fünf Ebenen sind in Abbildung 4.11

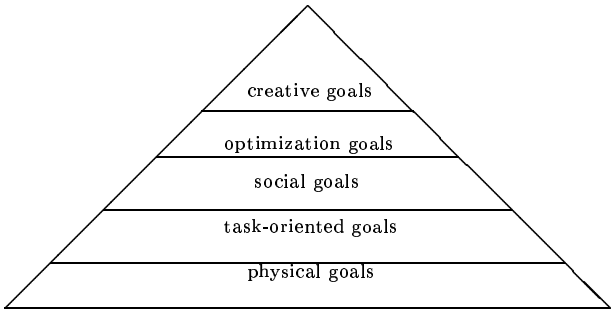

Abbildung 4.11: Maslow Pyramide der menschlichen Bedürfnisse aus [MP93]

angegeben. Laut Maslow haben Bedürfnisse bzw. Ziele der unteren Ebenen eine höhere Priorität als Bedürfnisse bzw. Ziele der höheren Ebenen und Ziele der unteren Ebenen müssen vor den Zielen der höher gelegenen Ebenen bearbeitet werden. Das bedeutet, dass Ziele einer höheren Ebene erst bearbeitet werden können, wenn Ziele der darüberliegenden Ebene abgearbeitet wurden.

Das Modell von Maslow wird wie folgt übertragen. Physische Ziele korrespondieren mit den grundlegenden „Top-level“ Zielen eines Agenten und dienen dazu dem Agenten seine physische Unversehrtheit zu bewahren. Aufgabenorientierte Ziele entsprechen den Zielen, die der Agent zur Bearbeitung aufgetragen bekommt. Soziale Ziele entsprechen Zielen die in Verbindung mit anderen Agenten gesehen werden können und optimierende Ziele sind Ziele die z.B. die Bearbeitung von anderen Zielen durch bestimmte Pläne optimieren. Die Ebene der kreativen Ziele korrespondiert nur schwach mit der aktuellen Forschung im Bereich der künstlichen Intelligenz und wird daher durch die Autoren in [MP93] nicht in bestimmte Ziele umgesetzt.

Von der Ordnung auf Zielen die durch das Modell von Maslow impliziert wird sollte in einigen Fällen abgewichen werden. So kann es sein, dass ein Ziel einer höheren Ebene, unabhängig von der Anordnung in der Pyramide, eine höhere Priorität hat als ein Ziel auf einer unteren Ebene. Dadurch wird es notwendig, dass dieses Ziel, Ziele auf der unteren Ebene in ihrer Ausführung hemmt. Ebenso kann es aber auch sein, dass ein Ziel auf einer niedrigeren Ebene optimal ausgeführt werden könnte, wenn z.B. ein soziales Ziel in die Bearbeitung des jeweiligen Ziel integriert werden könnte. Hierbei zu erwähnen wäre der Vorgang der Kooperation.

Um diese Abweichung von der statischen Priorität zu implementieren wird in [MP93] auf die durch Brooks geprägten Mechanismen Suppression und Inhibition zurückgegriffen. Inhibition realisiert dabei den Vorgang, dass ein Ziel einer höheren Schicht die Bearbeitung eines Ziel einer unteren Ebene zu gunsten

der eigene Bearbeitung hemmen kann. Suppression wird dabei als ein Konzept ähnlich zu der im Folgenden erwähnten dynamischen Priorität gesehen.

Dynamische Priorität

Der Anteil der dynamischen Priorität an der Gesamtpriorität soll den Zweck der Berücksichtigung einer relativen Wichtigkeit haben. Hinter dem Konzept der relativen Wichtigkeit steht die Idee eines „Grades an Zufriedenheit“ (degree of satisfaction). Dieser Grad wird bei der Erstellung eines Ziel instantiiert. Mit fortschreitender Zeit und im Hinblick auf mögliche zeitliche Beschränkungen des betreffenden Ziels verringert sich der Grad an Zufriedenheit und erhöht sich somit die Wichtigkeit des Ziels. Bearbeitet der Agent solch ein spezifiziertes Ziel, so erhöht sich der Grad an Zufriedenheit und erniedrigt sich damit wieder die Wichtigkeit dieses Ziels. Laut [MP93] sind durch die Verwendung beider Typen von Prioritäten angemessene Entscheidungen möglich. Die Prioritätsfunktion wird nun wie folgt genauer spezifiziert:

Für $\mathcal{G}$ als Menge der Ziele und $G \in \mathcal{G}$ definiert man die Prioritätsfunktion $f : \mathcal{G} \to \mathcal{N}$ als:

$$f(G) = f_{stat}(G) + f_{dyn}(G)$$

Eine genaue Spezifizierung wie eine dynamische Priorität berechnet werden soll bleiben die Autoren in [MP93] schuldig. Sie erwähnen jedoch, dass solche ein Berechnung in einem hohen Maß domänenabhängig ist und schlagen vor, heuristische Kriterien wie zeitliche Beschränkungen von Zielen, Verfügbarkeit oder dem Mangel an Ressourcen zu verwenden.

Algorithmus zur Auswahl von Zielen

In Abhängigkeit des Vorhandenseins von statischer bzw. dynamischer Priorität unterscheidet sich der Algorithmus zur Auswahl von Zielen. Dabei ist zu berücksichtigen, dass die Autoren in [MP93] von Verfeinerungen zur Bearbeitung der Ziele ausgehen, die „Und“ verknüpft sind. Dies bedeutet, dass nur eine Auswahl und dies auf der ersten Ebene (Ebene 0) getroffen werden muss. Bei statischer Priorität wird folgendermaßen vorgegangen.

- Wähle die kleinstmöglichste Ebene mit aktiven Zielen aus.
- Befindet sich auf dieser Ebene nur ein aktives Ziel, so wähle dieses aus.
- Befinden sich mehrere aktive Ziele auf dieser Ebene, so wähle zufällig eines dieser Ziele aus.
- Führe den Plan zu diesem Ziel aus.
- Nach Beendigung des Plans beginne wieder von vorne.

Bei Berücksichtigung von dynamischer Priorität, wird eine Agenda mit allen aktiven Zielen unterhalten und der Algorithmus lautet wie folgt:

- Berechne die Priorität der aktiven Ziele und ordne diese Ziele bezüglich ihrer Priorität in der Agenda an.
- Wähle das erste Ziel aus.
- Führe Plan zu diesem Ziel aus.
- Nach Beendigung des Plans beginne wieder von vorne.

Bewertung des Verfahrens

Spezifikation Die Autoren in [MP93] erkannten bereits, dass einerseits die Spezifizierung der Priorität von Zielen anhand einer Abstufung und der daraus-folgenden vorgegebenen sequentiellen Abarbeitung ein zu starres Konzept für die Anforderungen an Agenten in dynamischen Umgebungen darstellten und führten die Möglichkeit der Spezifizierung einer Inhibition ein. Andererseits war aber auch die Verwendung von Konstrukten aus anderen wissenschaftlichen Bereichen (Maslow Pyramide), durch eine unzureichende übereinstimmung mit den Anforderungen eines Agenten negativ begleitet. Daher kann man der Spezifizierung von statischen Prioritäten aufgrund der vorgestellten Pyramide skeptisch gegenüber stehen. Auch wurde die Spezifikation einer dynamischen Priorität zu wenig ausformuliert. Das liegt zu großen Teilen daran das eine dynamische Priorität stark domänenabhängig ist und somit in einer allgemeinen Art und Weise nur schwer zu erfassen sein wird.

Desweiteren hegten die Autoren in [MP93] Bedenken gegenüber dem Spezifizierungsaufwand der mit dem Design der zweidimensionalen Entscheidungskomponente einhergeht. Sie befürchteten, dass sie sich damit einen erheblichen Spezifikationsaufwand eingehandelt zu haben. In [MP93] führen sie dazu aus:

> „A Major criticism of the model is that, in complex real-world domains, enumerating all the goals an agent may have and thus also describing all the situations an agent have to react to, may be too expensive, or even impossible."

Jedoch relativieren sie ihre Bedenken, da aufgrund empirischer Untersuchungen diese Befürchtungen bis jetzt nicht verifiziert werden konnten.

Sinnvoll, und auch zu beobachten in Literatur zum Verständnis von Texten, erscheint mir die allgemeine Idee die hinter dieser Pyramide steckt. Die Priorisierung von Zielen aufgrund der Klasse des jeweiligen Ziels lässt sich meiner Meinung besser auf die Typen von Zielen wie „Achieve" oder „Maintain" übertragen. So könnte man sich z.B. vorstellen, dass ein Ziel vom Typ „Maintain" generell eine höhere Priorität besitzt als ein Ziel vom Typ „Achieve".

Entscheidung Interrap entscheidet nur zwischen Zielen auf der obersten Ebene. Es unterstützt keine Entscheidungen auf tieferen Ebene. Dadurch geht wichtiges Potential verloren. Das hier beschriebene Verfahren ist also im Hinblick auf seine Entscheidungsmöglichkeiten recht dürftig ausgestattet.

Zusammengefasst kann gesagt werden, dass das hier beschriebene Verfahren versucht eine Vielzahl an Einflüssen auf ein Ziel in einer Entscheidung zu bündeln. Jedoch ist erkennbar, dass diese Einflüsse entweder keinen so rechten Bezug zur Materie der intelligenten Agenten herstellen konnten (Maslow Pyramide) oder zu vage formuliert waren (dynamische Priorität). Negativ ins Auge fällt der geringe Entscheidungsspielraum dem man mit diesem System hat. Durch eine Entscheidung lediglich auf der obersten Ebene eines Ziel verschenkt man viel Potential.

4.7.2 HAP

Hap [Loy97] ist eine Teilarchitektur des Systems Tok für sogenannte glaubwürdige Agenten (believable agents). Glaubwürdige Agenten sind dabei eine Kombination aus autonomen Agenten und Darstellern ähnlich denen aus Theater oder Film. Hap bietet dabei die Möglichkeit mehrere aktive Ziele zu verfolgen und jeweils eines für die Bearbeitung auszuwählen. Diese Auswahl wird im folgenden das Thema der Betrachtungen sein.

Hap verwendet einen sogenannten „Active Behaviour Tree“ (ATB) zur Organisation aller vom Agenten verfolgten Ziele und verwendeten „Behaviours“ . Dabei kann man sich die sogenannten „Behaviours“ als Pläne vorstellen die die gewohnten Vorbedingungen (preconditions) und Kontextbedingungen (context-condition) aufweisen. Behaviours dienen der Bearbeitung der definierten Ziele und bestehen aus Operationen die weitere Ziele erzeugen oder Operationen die atomare Aktionen kapseln. Desweiteren sind Behaviours einem bestimmten Typ zugeordnet. So können sequentielle, parallele und gemischte Typen definiert werden. In Abbildung 4.12 ist ein ATB dargestellt. Der Wurzelknoten wird durch

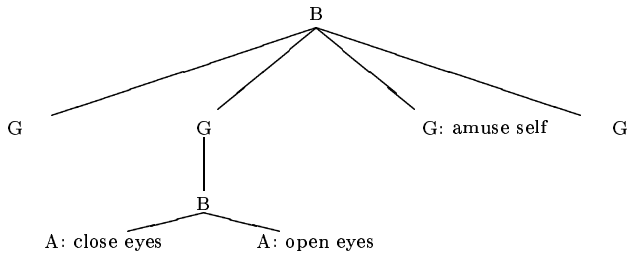

Abbildung 4.12: Active Behaviour Tree

ein „Behaviour“ definiert, welches vom Typ „gemischt“ ist. Die Ziele die durch dieses Wurzelbehaviour spezifiziert werden, sind die „Top-level“ Ziele des Agenten. Bei einer erfolgreichen oder bei einer fehlgeschlagenen Bearbeitung eines Knotens des ATB, wird dieser entsprechend modifiziert. Ist z.B. ein Knoten erfolgreich bearbeitet worden und handelt es sich bei diesem Knoten um ein Ziel eines sequentiellen „Behaviours“, so wird geprüft, ob es sich bei dem bearbeiteten Knoten um den letzten Knoten in der Abarbeitung der sequentiellen Knoten

des Vaterknoten handelt. Ist dies der Fall, so ist damit auch der Vaterknoten also das sequentielle „Behaviour“ erfolgreich bearbeitet und kann samt seines Teilbaumes entfernt werden.

Durch die Spezifizierung des Wurzelnoten als ein „Behaviour“ vom Typ gemischt, können die „Top-level“ Ziele nebenläufig zueinander ausgeführt werden. Das bedeutet aber auch, dass der Agent damit möglicherweise auftretende Konflikte zwischen diesen nebenläufig abzuarbeitenden Prozessen berücksichtigen muss. In [Loy97] sind verschiedene domänenspezifische Typen von Konflikte beispielhaft dargestellt. So ist dort z.B. von physischen Konflikten die Rede. Allgemein betrachtet bedient sich der Hap-Agent aber eine Konfliktliste in der Paare von Zielen aufgeführt sind die inkompatibel zueinander sind und damit nicht gleichzeitig ausgeführt werden sollen. Wenn zwei in Konflikt stehende Ziel ausgeführt werden sollen, so wird dasjenige Ziel welches eine geringere Priorität aufweist, vorläufig in seiner Bearbeitung ausgesetzt. Aber auch bei Zielen mit gleicher Priorität und einem Eintrag in der Konfliktliste wird einer der beiden Ziele ausgesetzt. Das wird jenes Ziel sein, welches nicht zur Bearbeitung ausgewählt worden ist.

Mit jedem Ziel des ATB wird eine Priorität assoziiert. Diese Priorität wird numerisch angegeben und definiert in ihrer Gesamtheit eine partielle Ordnung auf den Ziele (auch Schritte genannt) des ATB. Diese Priorität wird mit Instanzen von Zielen assoziiert, nicht mit Typen von Zielen. Das hat den Vorteil, dass verschiedene Instanzen eines Zieltyps, in Abhängigkeit des Kontextes, unterschiedliche Prioritäten herausbilden können. Aber auch dynamische Aspekte der Priorität von Zielen sind in [Loy97] erwähnt. So wird angemerkt, dass eine offensichtliche Erweiterung von Hap, die dynamische Änderung von Prioritäten aufgrund von Änderungen der Situation darstellen könnte. Prioritäten von Zielen in Hap unterscheiden sich somit nicht nur zwischen Instanzen, sondern auch innerhalb eines Ziels ansich während der Laufzeit.

Prioritäten von initialen Zielen werden durch den Entwickler spezifiziert. Er gibt an, welche Ordnung der Agent auf seinen Zielen definiert haben will.

Prioritäten von Zielen einer Ebene unter der Ebene ihrer Väter besitzen zudem noch einen Prioritätenmodifizierer. Dieser Modifizierer gibt Auskunft darüber wie die Priorität des betreffenden Knotens berechnet werden soll. Der im Modifizierer angegebene Wert wird zu dem Wert des Vaterknotens addiert. So können Subziele einen höheren Wert, den gleichen Wert oder einen niedrigeren Wert als ihr Vaterziel aufweisen. Dies hat folgenden Grund. Der Ausführungszyklus eines Hap Agenten wählt immer ein Blatt zur Ausführung aus. Da hier Ziele wie auch Aktionen auftreten können handelt es sich im Sinne von Hap immer um ein Ziel Durch den Modifizierer kann man jetzt steuern in welchen Verhältnis die Blätter eines zu bearbeitenden Ziel zu den restlichen Blättern stehen sollen. Sollen sie wichtiger oder unwichtiger als jene sein oder sollen sie gleichbedeutend wichtig mit den restlichen Blättern, also Zielen eines Hap Agenten sein. Bei Blättern die einen identischen Prioritätswert haben werden jene bevorzugt die sich in einen Teilbaum eines gerade ausgeführten Ziels befinden. Das bedeutet, dass der Agent seine Energie in die Arbeit investiert, die er schon zu einem Teil vollendet hat.

Bewertung des Verfahrens

Spezifikation Das hier beschriebene Verfahren ermöglicht die Angabe von Prioritäten. Diese Prioritäten werden numerisch spezifiziert. Das bedeutet aber wiederum, dass der Entwickler eines Agenten abermals eine Ordnung auf den Zielen eines Agenten definieren muss. Zusätzlich dazu kann Hap, im Gegensatz zu Interrap, in Konflikt stehende Ziele in einer Konfliktliste vermerken. Durch die Angabe des Prioritätenmodifizierer kann die spezifizierte Priorität des Ursprungziels modifiziert werden. Dadurch sind die Prioritäten der Subziele eines Ziels stärker an die vorgegebene Priorität des Ursprungziels gebunden. Die Verwendung des ATB ist nur nötig, da Hap kein erweitertes Zielkonzept besitzt. Hap muss so die Struktur eines Ziels nachbilden.

Entscheidung Das Verfahren trifft nur in dem Fall, in dem ein Ziel in Konflikt mit einem anderen auszuführenden Ziel steht, eine Entscheidung auf Basis von Prioritäten. Diese Entscheidungen können nicht nur einmal auf oberster Ebene, sondern jedesmal wenn Ziele in Konflikt stehen ausgeführt werden. Im Gegensatz zu dem in 4.7.1 beschriebenen Verfahren stellt dies einen entscheidenden Vorteil dar.

Zusammenfassend kann man dieses Verfahren als durchaus interessant bezeichnen. Es bietet die Erkennung von Konflikten und eine Entscheidung die im Fall eines Konflikts sich nach der Priorität des jeweiligen Ziel richtet. Jedoch ist Priorität als ein abstrakter Begriff für die Bevorzugung eines Ziels ein noch zu einfacher allgemeiner Begriff. Ebenso fordert eine Priorität eine Spezifikation einer globalen Ordnung auf Zielen. Das dies Schwierigkeiten bereiten kann, wurde bereits in 4.2.3 angesprochen.

4.8 Zusammenfassung

In diesem Kapitel wurden verschiedene Verfahren zur Deliberation auf Zielen beschrieben. Diese Verfahren unterschieden sich zum Teil erheblich. So wurden zum Einen bereits sehr konkrete Verfahren (siehe 4.3.2) aufgeführt, die eine Vorstellung von dem erahnen lassen konnten, was mit dem betreffenden Verfahren geleistet werden kann. Zum Anderen wurden aber auch Verfahren vorgestellt, die nicht so konkret ausformuliert waren bzw. sehr einfache Entscheidungsstrukturen bereitstellten. Dies liegt, im Fall der einfachen Verfahren, nicht an dem Vermögen der Autoren, sondern vielmehr an den Gegebenheiten mit denen diese umgehen mussten. So haben generell die meisten, wenn nicht sogar alle hier aufgezeigten Verfahren[5] ein Defizit in der Darstellung ihrer Ziele. Viele begnügen sich damit, den prozeduralen Aspekt eines Ziels zu verwenden. Damit sind dann aber auch die Fähigkeiten deliberativ aktiv zu werden, erheblich eingeschränkt. Vielfach kam es den Entwicklern der beschriebenen Systeme auch nur darauf an „irgendein“ Selektionskriterium zu besitzen um aus der Vielzahl der Ziele eine

[5] Das Verfahren aus Jadex, Easy Deliberation, ist von dieser Beurteilung auszunehmen

Auswahl treffen zu können. Dabei fiel die Wahl häufig auf die Verwendung von Prioritäten.

Für die Auswahl eines dieser Verfahren kam es darauf an, ein Verfahren zu selektieren, welches einen gewissen generischen Charakter besitzt, also in einer Vielzahl von Anwendungen einsetzbar ist und welches möglicherweise das bestehende Verfahren in Jadex, die „Easy Deliberation“ in seinen Schwachpunkten unterstützt. Ebenso von Interesse war, dass das ausgewählte Verfahren einen Spezifikationsaufwand fordert der in gewissen Grenzen liegen sollte. Es sollte möglich sein, einfach und schnell die Information spezifizieren zu können die von dem Verfahren benötigt wird. Es sollte nicht notwendig werden, komplizierte Funktionen oder Zustandsbeschreibungen zu konstruieren, denn dies würde der Philosophie wiedersprechen die hinter Jadex steht.

Die Wahl des Verfahrens, welches in dieser Diplomarbeit implementiert werden sollte, fiel dabei auf jenes welches die „Resource Summary“ verwendet. Ressourcen werden dabei als ein allgemeines in einer Vielzahl von Anwendungen einsetzbares Konzept zur Deliberation gesehen. Sie sind aber auch spezifisch genug, um konkrete Entscheidungsmuster herausbilden zu können. Weiterhin operiert das Verfahren auf der Planebene, was bisher in der vorliegenden Implementation der „Easy Deliberation“ Strategie nicht möglich war. Somit stellt dieses Verfahren eine interessante Alternative zum vorhandenen Verfahren in Jadex dar.

Kapitel 5

Ein Deliberationsverfahren für Jadex

Nach der Beurteilung der in Kapitel 4 behandelten Deliberationsverfahren, hat sich das Verfahren von [TWPF2002b] als das Verfahren der Wahl herausgestellt. In diesem Kapitel soll gezeigt werden, wie es in Jadex integriert wurde. Dazu wird folgendermaßen vorgegangen.

Zunächst wird eine Konzeption des zu entwickelnden Verfahrens vorgestellt. Darin wird beschrieben, wie die in [TWPF2002b] aufgeführten Begriffe und Verfahren umgesetzt werden. Im Speziellen wird darauf eingegangen, an welchen Stellen des Lebenszyklus eines Ziels das Verfahren ansetzt, was die verschiedenen Status bedeuten und wie die Ressourceninformationen gestaltet werden. Anschließend werden in einem zweiten Abschnitt die implementationstechnischen Details behandelt. Dazu gehört die Erläuterung der vorgenommenen Änderungen in der die ADF beschreibende XML Sprache und die genaue Umsetzung und Integration der im Abschnitt Konzeption vorgeschlagenen Meta-Aktionen. Abschließend wird das so implementierte Verfahren in einem dritten Abschnitt einer Bewertung unterzogen. Zum einen soll hier ein Vergleich mit dem in [TWPF2002b] beschriebenen Verfahren und dessen Implementation „X-JACK“ [TP2005] geführt werden. Zum anderen soll das neue Verfahren in den Kontext des bestehenden eingeordnet werden. Hierbei soll zum Einen aufgezeigt werden, welche Vorteile die hier vorgenommene Implementation gegenüber der in [TP2005] dargestellten aufweist. Und zum Anderen welche Vorteile und Nachteile sich gegenüber dem bestehenden Verfahren herausbilden. Es soll ebenso diskutiert werden, ob sich bestehendes Verfahren und das neu entwickelte ergänzen.

5.1 Konzeption

In diesem Abschnitt soll eine Konzeption des zu implementierenden System erstellt werden. Hierbei wird folgendermaßen vorgegangen. Zuerst werden die in

[TWPF2002b] verwendeten Begriffe wie „Resource“ oder „Resource Summary“ im Hinblick auf eine Implementierung konkretisiert. Anschließend sollen Überlegungen angestellt werden, wie eine Spezifikation der benötigten Informationen im „Agent Definition File“ aussehen könnten. Danach wird für die Entscheidungsmöglichkeiten die dieses Verfahren liefert, spezifiziert wie auf diese reagiert werden soll. Zum Beispiel sollen sich hier solche Fragen klären, wie das Verfahren auf einen Aktivierungsstatus „Schedulable“ reagiert. Schlussendlich sollen die Ansatzpunkte für dieses Deliberationsverfahren in der Attitüde und dem dynamischen Zustand eines Ziels lokalisiert werden.

5.1.1 Ressourceninformation

Als Grundlage des hier konzipierten Verfahrens werden die für Ziele und Pläne zu berechneten Ressourceninformationen (Resource Summary) angesehen. Die Ressourceninformation wird aufgeteilt in eine Auflistung der notwendigerweise und möglicherweise benötigten Ressourcen. Die notwendigerweise benötigten Ressourcen werden in einer Liste N, die möglicherweise benötigten Ressourcen in einer Liste P gespeichert.

Zur Darstellung und Verwendung von Ressourcen in Jadex, muss ein Ressourcenkonzept entwickelt werden, denn Jadex kennt in diesem Sinne keine Ressourcen. Eine Ressource an sich wird als eine Kombination aus einem Ressourcentyp, einem Bedarf (requirements) und einer Art betrachtet. Der Ressourcentyp ist ein Bezeichner für ein „Belief“ das im Besitz des Agenten ist. Es können somit nur Ressourcen verwendet werden, von denen der Agent auch Kenntnis hat. Der Bedarf einer Ressource gibt an, in welchem Umfang das referenzierte „Belief“ benötigt wird. Die Art einer Ressource beschreibt, ob es sich bei der benötigten Ressource um eine verbrauchbare oder um eine wiederverwendbare handelt.

Die Berechnung der Ressourceninformation verläuft wie in [TWPF2002b] dargestellt. Für ein Ziel berechnet sich dessen Ressourceninformation aus denen mit dem Operator $\uplus$ verknüpften Ressourceninformationen der anwendbaren Pläne. Für jeweils einen einzelnen Plan berechnet sich dessen Ressourceninformation aus der Kombination der Ressourceninformation möglicher Subziele mit dem Operator $\otimes$ mit anschließender Kombination ($\oplus$) der Ressourceninformation für die elementaren Aktionen des Plans selbst.

5.1.2 Agent Definition File

Die vorhandene XML Sprache der ADF lässt einen Entwickler die Beziehungen zwischen Zielen, Plänen und „Beliefs“ für die Anforderungen des Verfahrens, nicht ausreichend detailliert darstellen. Denn in der „Agent Definition File“ ist momentan nur ein „unvollständiger“ „Goal-plan Tree“ darstellbar. So ist zur Modellierungszeit durch die Spezifikation von `<trigger>` nur die Beziehung zwischen einem Ziel und den anwendbaren Plänen, nicht aber die Beziehung zwischen Plänen und erzeugten Subzielen darstellbar. Der Bezug zwischen Plänen und Subzielen ist aber für eine Vorherbestimmung der Ressourceninforma-

tion eines Ziels notwendig. Genauso mangelhaft ist die Möglichkeit Ressourcen darzustellen die durch die einzelnen Pläne benötigt werden. Durch eine Erweiterung der XML Sprache werden diese Nachteile behoben. Dem Entwickler soll dann die Möglichkeit gegeben werden, alle fehlenden Informationen darstellen zu können. Dabei wird er nur auf der Planebene modellierend tätig werden. Dort spezifiziert er für die jeweiligen Pläne welche Ressourcen und welche Subziele sie benötigen. Die Angabe der Ressourcen für Pläne bezieht sich nur auf die lokal benötigten. Erzeugt ein Plan Subziele, so werden mögliche Ressourcen dieser Subziele nicht in dem betreffenden Plan spezifiziert. Die Spezifikation dieser Ressourcen erfolgt in den Plänen zu diesen Subzielen. Die Spezifikation einer Ressource verbindet also einen Plan mit einem vorhandenen Belief und stellt die Anforderungen an das Belief dar. Die Spezifikation eines Subziels verbindet einen Plan mit vorhandenen Zielen.

5.1.3 Status

Die Verwendung der Ressourceninformation alleine ist nutzlos, solange sie nicht in Verbindung mit denen anderer Ziele und der vorhandenen Menge an Ressourcen gebracht wird. Diese Verbindung stellt der Algorithmus zur Berechnung des Status her. In [TWPF2002b] werden als Ergebnis der Berechnung des Status eines Ziels, sechs verschiedene Zustände verwendet. Diese werden für die eigene Implementation übernommen.

Das hier beschriebene Verfahren verwendet eine modifizierte Fassung des in [TP2005] beschriebenen Algorithmus. Ein wichtiger Unterschied ist, dass nur einzelne Ziele betrachtet werden. Zur Bestimmung des Status der Aktivierung eines Ziels werden drei Konfigurationen unterschieden.

- Das betreffende Ziel weist keine Einträge in der Menge der notwendigen und in der Menge der möglichen Ressourcen auf.
- Das betreffende Ziel weist nur Einträge in der Menge P auf.
- Das betreffende Ziel weist Einträge in beiden Mengen auf.

Trifft Situation eins zu, so ist es sicher dieses Ziel zu aktivieren. Dieses Ziel kann in Bezug auf Ressourcen keinerlei Probleme hervorrufen. In Situation zwei muss der Algorithmus nur einen Teilmenge der möglichen Ergebnisse prüfen. Er prüft ob Status „Schedulable“ oder „Safe“ zutrifft. Trifft keiner der beiden Status zu, so gilt für die betreffende benötigte Ressource der Status „Uncertain“. In Situation drei müssen alle möglichen Status untersucht werden. Der Gesamtstatus eines zu aktivierenden Ziels setzt sich der Berücksichtigung der Status der einzelnen Ressourcen zusammen. So ist der Gesamtstatus der Aktivierung eines Ziels „Safe“, wenn alle Ressourcen ein Status „Safe“ besitzen. Der Gesamtstatus ist hingegen „Conficiting“, sobald eine Ressource den Status „Conflicting“ besitzt. Die Mischformen „Schedulable“, „Schedule-dependent“ und „Scheduleable+Schedule-dependent“ werden nach ihrer Wichtigkeit geordnet. So gilt der Gesamtstatus „Scheduleable+Schedule-dependent“, wenn keiner

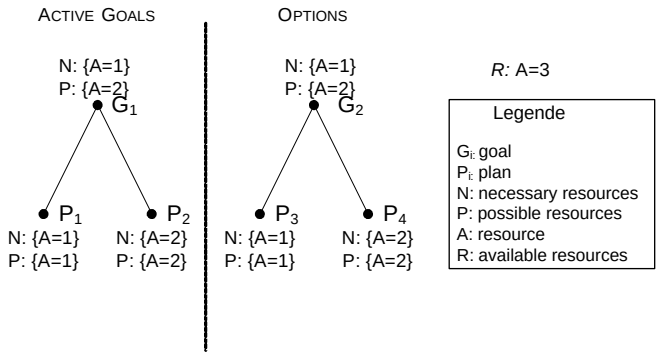

Abbildung 5.1: Ein Ziel mit Status Schedulable

der Ressourcen den Zustand „Conflicting“ bestitzt und mindestens eine Ressource den Zustand „Schedulable+Schedule-dependent“ besitzt. Für den Gesamstatus „Schedule-dependent“ gilt, dass keine Ressource den Zustand „Conflicting“ bestitzt, keine Ressourcen den Zustand „Scheduleable+Schedule-dependent“ besitzt und mindestens eine Ressource den Zustand „Schedulable“. Der Zustand „Schedulable“ kommt zustande, wenn alle anderen Zustände ausgeschlossen werden und mindestens ein Zustand „Schedulable“ ist. Gilt keiner der angesprochenen Zustände, so ist der Gesamtzustand mit „Uncertain“ festzulegen.

Nachdem der Aufbau des verwendeten Algorithmus erläutert wurden ist, soll jetzt beschrieben werden, wie mit dem berechneten Status umgegangen werden soll.

Safe

Ist eine Aktivierung eines Ziels, unter Berücksichtigung der Ressourceninformation sicher (Safe), so wird dieses Ziel ohne weitere Einschränkungen aktiviert.

Conflicting

Besteht hingegen ein Konflikt (Conflicting), wenn dieses Ziel aktiviert würde, so wird eine Aktivierung vermieden. Vielmehr wird das Ziel komplett verworfen und in den Zustand „Dropped“ überführt. Dies geschieht unabhängig davon, ob es sich bei den verwendeten Ressourcen um verbrauchbare oder wiederverwendbare handelt. In keinem Fall lohnt eine weitere Berücksichtigung dieses Ziels als Option.

Schedulable

Das Potential des Status „Schedulable“ soll an einer einfachen Graphik erläutert werden. In Abbildung 5.1 ist auf der linken Seite ein bereits aktives Ziel

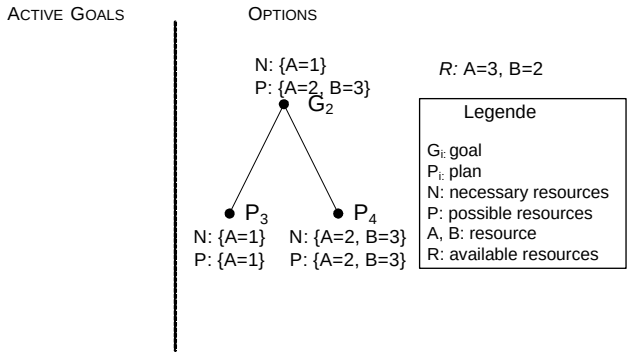

Abbildung 5.2: Ein Ziel mit Status Uncertain

mit seine Plänen dargestellt. Auf der rechten Seite befindet sich das zu aktivierende Ziel. Die Aktivierung hat das Ergebnis „Schedulable“ berechnet und es soll jetzt entschieden werden, was mit dem Ziel der rechten Seite geschehen soll. Wie man erkennen kann, könnte man Ziel zwei, unter der Bedingung das Plan drei zur Bearbeitung ausgeführt wird, sicher aktivieren. Würde Plan vier ausgewählt werden, so kommt man aufgrund dieser Tatsache zu keiner neuen Erkenntnis. Das Ziel zwei bleibt dadurch im Status „Schedulable“. Somit kann man Schlussfolgern, dass unter der Bedingung der Anwendung eines speziellen Plans Ziele deren Status „Schedulable“ ist, aktiviert werden können.

Schedule-dependent

Der Status „Schedule-dependent“ bietet nur eine Reaktionsmöglichkeit. Das betreffende Ziel muss solange warten bis die aktiven Ziele die benötigten Ressourcen wieder freigegeben haben. Erst dann besteht eine erfolgversprechende Aussicht auf eine sichere Aktivierung.

Uncertain

Der Status „Uncertain“ ähnelt im Reaktionsverhalten dem Status „Schedulable“. Ein einfaches Beispiel soll das Potential dieses Status verdeutlichen. In Abbildung 5.2 ist das Ziel eins mit den vorhandenen Plänen abgebildet. Der Plan zwei benötigt Ressource B in einem Umfang der ihm nicht zur Verfügung steht. Dadurch ergibt sich der Status „Uncertain“. Wie man erkennen kann, kommt es bei der gezielten Wahl des Plans eins zu keinerlei Komplikationen, während die Auswahl von Plan zwei einen Konflikt verursachen würde. Daraus schlussfolgert man, dass auch bei diesem Status die gezielte Auswahl eines Plans eine Aktivierung des betreffenden Ziels ermöglicht.

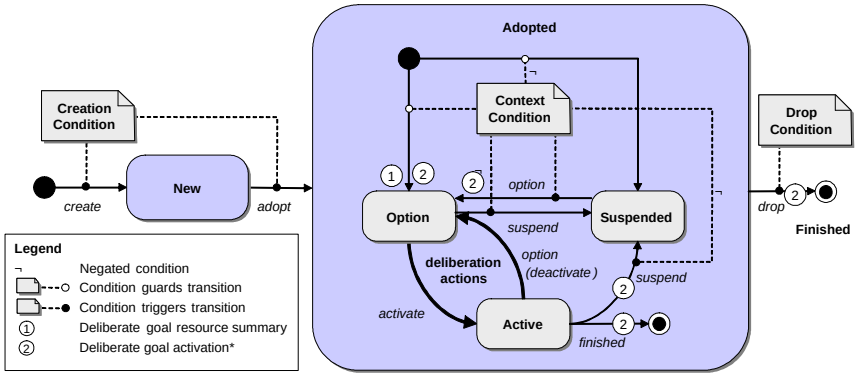

Abbildung 5.3: Ansatzstellen der Deliberation an der Attitüde eines Ziels, nach [PBL05a]

Schedule-dependent und Scheduleable

Bei diesem speziellen Status wird wie im Fall von „Schedule-dependent“ verfahren. Das Ziel muss demnach solange warten bis sich ein günstigerer Status ergibt bzw. die Ziele, die vorher bearbeitet werden müssen, beendet sind.

5.1.4 Deliberationsschnittstellen

Das hier konzipierte Verfahren soll ins Jadex System integriert werden. Wie bereits bekannt, bietet Jadex die Konzepte „variabler Interpreterzyklus“ und „erweitertes Zielkonzept“. Um bestehende oder neue Deliberationsverfahren in Jadex zu integrieren ist es notwendig für das jeweilige Verfahren entsprechende Meta-Aktionen zu spezifizieren. In dem hier vorliegenden Verfahren z.B. eine Meta-Aktion zur Bestimmung der Ressourceninformation eines Ziels. Dieser Abschnitt versucht sowohl die benötigten Meta-Aktionen zu konzipieren wie auch die Ansatzpunkte für diese Aktionen innerhalb des Zustandsraum des Ziels zu erkunden. Jadex implementiert vier verschiedene Zieltypen. Dabei bilden die Typen „Achieve“, „Perform“ und „Query“ eine Einheit. Im Normalfall ist eine Aktivierung dieser Zieltypen unmittelbar mit der Ausführung von Aktionen verbunden. Im Gegensatz dazu ist eine Aktivierung eines Ziels vom Typ „Maintain“ nicht notwendigerweise mit der Ausführung von Aktionen verbunden. Ziele vom Typ „Maintain“ werden erst bei einer Verletzung der durch sie überwachten „Maintaincondition“ bearbeitet. Bei einigen Meta-Aktionen wird daher zwischen Zielen vom Typ „Maintain“ und Zielen vom Typ der Menge „Achieve“, „Perform“ und „Query“ unterschieden.

Das hier beschrieben Verfahren erfordert wie die in 4.2 beschrieben „Easy Deliberation“ Strategie, dass der Agent Kontrollmöglichkeiten besitzen sollte. Diese Kontrollmöglichkeiten ermöglichen die Aktivierung von im Zustand „Option“ befindlichen Zielen zu steuern. Daraus folgt, dass der Agent durch dieses

Verfahren ebenfalls die Meta-Aktion Deliberate goal activation benötigt. Eine Voraussetzung um über die Aktivierung eines Ziels deliberieren zu können, ist im Fall des hier verwendeten Verfahrens jedoch zusätzlich dazu die Deliberation über die Ressourceninformation des Ziels. Ein Agent benötigt Informationen über die notwendig und über die möglich verwendeten Ressourcen um eine Entscheidung hinsichtlich der Aktivierung treffen zu können. So ergibt sich die Einführung der Meta-Aktion „Deliberate goal resource summary". Diese Aktion wird im Verbund mit der Aktion Deliberate goal activation eingesetzt. Zuerst wird durch sie die Ressourceninformation berechnet um eine Entscheidung bzgl. einer Aktivierung zu ermöglichen. Dabei gilt die Einschränkung, dass Zieltypen der Menge „Achieve", „Perform" und „Query" in Abhängigkeit von ihrem Status aktiviert werden während ein Ziel vom Typ Maintain einfach aktiviert wird. Eine Prüfung des Status eines Ziels vom Typ Maintain findet zu einem späteren Zeitpunkt statt.

Wird ein Ziel deaktiviert, z.B. aus Gründen einer erfolgreichen bzw. nicht erfolgreichen Bearbeitung oder der Verletzung der „Contextcondition", so besteht in dieser Situation ein weiterer Ansatzpunkt an dem das Deliberationsverfahren eingreifen sollte. Es muss die Ziele bestimmen, die aufgrund von Ressourcenmangel nicht parallel mit dem deaktivierten Ziel ausgeführt werden konnte und mit dem Vorliegen der aktuellen Situation eine berechtigte Chance auf eine Aktivierung hätten. Für jedes dieser Ziele entscheidet das Verfahren durch Anwendung der Meta-Aktion „Deliberate goal activation", ob das betreffende Ziel zum augenblicklichen Zeitpunkt aktiviert werden kann.

Die bis jetzt vorgestellten Ansatzpunkte für das Deliberationsverfahren beziehen sich nur auf die Attitüde des Agenten zu seinen Zielen. Ebenso von Interesse ist der dynamische Zustand eines Ziels. Hierbei sind folgende Situationen vorstellbar, die durch das Verfahren erkannt werden müssen:

- Ein Plan eines Ziels schlägt fehl. Die durch diesen Plan benötigten Ressourcen werden in der Gesamtauflistung der benötigten Ressourcen des betreffenden Ziels nicht mehr berücksichtigt und sind demnach freigegeben zur weiteren Verwendung durch andere Ziele.

- Ein Ziel ist nur unter der gezielten Auswahl eines Plans aktiviert worden. Dieser Plan schlug fehl. Mögliche andere, zur Ausführung bereitstehende Pläne sind in Bezug auf ihre benötigten Ressourcen nicht in jedem Fall anwendbar.

- Die „Maintaincondition" eines Ziels vom Typ „Maintain" ist verletzt. Das betreffende Ziel muss vom Ausführungszustand (processing state) „idle" nach „in process" wechseln.

Situation eins stellt eine Ursache für die Aktualisierung der Ressourceninformation dar. Dabei sollte das Deliberationsverfahren jedoch nicht in jedem Fall eine Aktualisierung vornehmen. Nur in dem Fall in dem der fehlgeschlagene Plan wiederverwendbare Ressourcen benutzte und diese in einer Menge benötigt wurden, die maximal im Vergleich zu den in anderen Plänen für dieses Ziel verwendeten

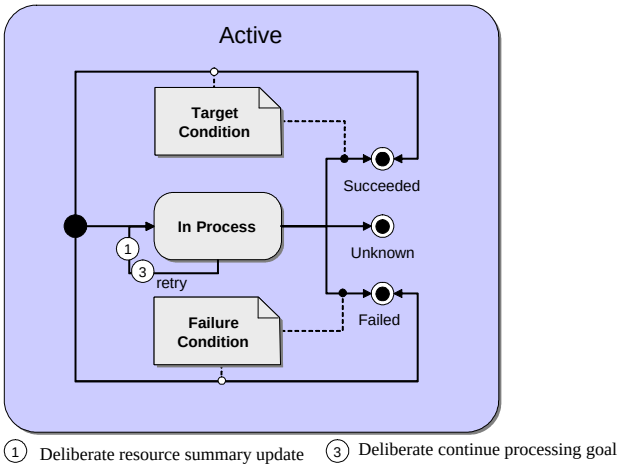

Abbildung 5.4: Ansatzpunkte für die Deliberation eines Ziels vom Typ Achieve

Werten war, ist ein Aktualisierung gerechtfertigt. Für verbrauchbare Ressourcen stellt eine Aktualisierung keinen Vorteil dar. Durch die Eigenschaft, dass dieser Ressourcentyp nach einer Verwendung nicht mehr verfügbar ist, reduziert sich die Anzahl der verfügbaren Ressourcen um den Anteil den der fehlgeschlagene Plan benötigt hat. Das Verhältnis von neuer Ressourceninformation zu vorhandener Menge an Ressourcen ist gleich geblieben. Die entsprechende Meta-Aktion Deliberate goal resource summary setzt diese Situation um. In Abbildung 5.4 ist der Ansatzpunkt des Deliberationsverfahren erkennbar. Das dort abgebildete Ziel vom Typ „Achieve“ steht dabei stellvertretend für alle anderen Zieltypen. Retry stellt dabei den Fall dar bei dem ein Plan fehlgeschlagen ist.

Wurde ein Ziel nur unter der Auflage aktiviert, weil es einen bestimmten Plan zu seiner Ausführung bieten konnte, so muss bei einem fehlschlagen dieses Plans die Auswahl weiterer Pläne einer Überwachung unterzogen werden. Das Verfahren prüft dabei, ob der als nächstes zur Bearbeitung anstehende Plan die weitere Bearbeitung des Ziels ermöglicht, indem über die Ressourceninformation des jeweiligen zugehörigen Ziels in Kombination mit dessen Aktivierung deliberiert wird. Ein Ziel vom Typ Maintain wurde in der oben beschriebenen Situation der Aktivierung eines Ziel, ohne Berücksichtigung des Status aktiviert. Ursache dieses Vorgehen war die Tatsache, dass Ziele vom Typ Maintain nicht unmittelbar in eine Bearbeitung führen müssen und somit die benötigten Ressourcen auf unbestimmte Zeit nicht in Anspruch genommen werden könnten. Das bedeutet zum Einen, dass ein Ziel vom Typ „Maintain“ erst ab dem Beginn der Bearbeitung, über den Status einer Bearbeitung deliberieren muss und zum Anderen, dass erst ein Ziel im Zustand „in process“ in einer Berechnung des Status für eine anderes Ziel berücksichtigt werden muss.

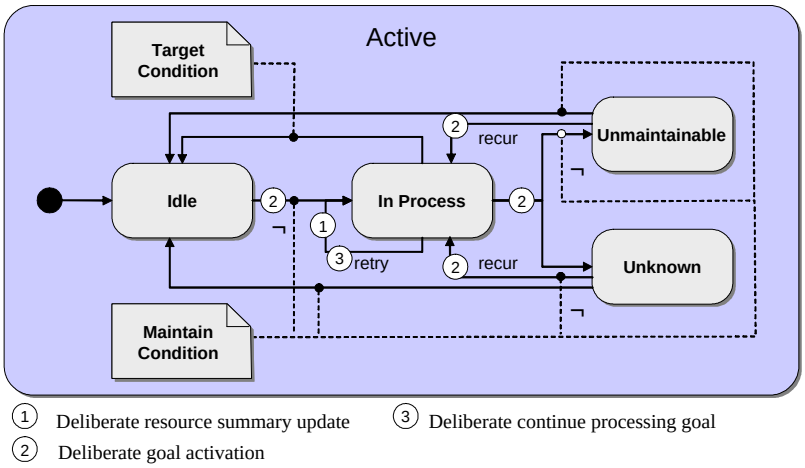

Abbildung 5.5: Ansatzpunkte für die Deliberation eines Ziels vom Typ Maintain

5.2 Implementation

5.2.1 Ressourceninformation

Die Klasse *RGoal* wird um die Attribute N und P ergänzt. Diese stellen die entsprechenden Mengen von Ressourcen jeweils in Form einer Liste dar. Diese Listen beinhaltetn die benötigten Ressourcen.

Eine Ressource wird durch die Klasse *Resource* repräsentiert. Objekte dieser Klasse besitzen einen Typ (type), einen Wert (value) und einen booleschen Eintrag für jede Art einer Ressource. Der Typ eines Ressourcenobjekt referenziert den Namen eines „Beliefs“ des Agenten. Den Wert für dieses Attribut bezieht der Agent aus dem im ADF spezifizierten Modell. Er steht im Attribut `ref` von `<resource>`. Der Wert (value) beschreibt den Anteil am referenzierten „Belief“ der durch diesen Plan benötigt wird. Generell sind hier Werte des Typs `java.lang.Number` möglich. Seinen Initialwert erhält das Attribut aus dem in `<resource>` spezifizierten Wert des Attributs `value`. Je nach Spezifizierung im ADF werden die booleschen Attribute für wiederverwendbare (reuseable) und verbrauchbare (consumable) auf die entsprechenden true oder false Werte gesetzt.

Für die Berechnungen zur Bestimmung des Status wurden statische Methoden in der Klasse *ResourceProcessor* definiert. Dabei handelt es sich hauptsächlich um Methoden wie die zur Bestimmung der sequentiellen unteren Grenze (calculateSequentialLowerBound), aber auch um zahlreiche Hilfsmethoden. Ebenfalls in dieser Klasse werden die Mengen N und P berechnet. Diese Methoden wurden in diesem Objekt konzentriert, weil sie in einer Vielzahl von Klassen benötigt werden. .

5.2.2 Status

Die Berechnung des Status der Aktivierung eines Ziels wurde in der Klasse *RGoalbase* verankert. Dort wird in der Methode `status` bei gegebenem Ziel für welches dieser Status berechnet werden soll (option) und für gegebene aktive Ziele ein Ergebnis berechnet. Dieses Ergebnis wird in dem Attribut status eines Objektes vom Typ *RGoal* gespeichert.

Bei der Berechnung des Status eines Ziels müssen verschiedene Bedingungen berücksichtigt werden. So werden keine aktiven Subziele sowie Ziele vom Typ Maintain im Ausführungszustand „Idle“ berücksichtigt. Das bedeutet, dass diese Ziele aus der Menge der aktiven Ziele entfernt werden. Subziele werden nicht berücksichtigt, weil die Ressourceninformation die sie besitzen, bereits implizit in der Ressourceninformation ihres übergeordneten Ziels enthalten ist. Würde die Ressourceninformation der Subziele berücksichtigt, so würde der Anteil der durch sie benötigt wird doppelt berechnet. Ziele vom Typ „Maintain“ im Zustand „Idle“ werden nicht berücksichtigt, weil sie wie bereits erwähnt nicht unmittelbar in eine Bearbeitung führen. Ein solches Ziel, würde es bei der Berechnung des Status berücksichtigt, bürgt ein zu hohes Risiko dass es benötigte Ressourcen beansprucht, die nicht unmittelbar benötigt werden. Somit werden nur Ziele vom Typ Maintain berücksichtigt die sich im Zustand „in process“ befinden.

5.2.3 Agent Definition File

Um dem Agenten die Möglichkeit zu eröffnen über seinen Ressourcenbedarf zu deliberieren, ist es notwendig, dass der Entwickler des Agenten verschiedene Informationen im ADF modellieren kann. Er muss Angaben zu den benötigten Ressourcen und zu denen von einem Plan erzeugten Subzielen tätigen können. In Abbildung 5.6 ist ein Ausschnitt aus einem ADF dargestellt. Dort werden die Möglichkeiten denen Entwicklern zur Verfügung stehen, um Ressourcen und Subziele für Pläne zu deklarieren, aufgezeigt. Durch die Deklaration von `<resources>` wird der Bereich gekennzeichnet, innerhalb dem die durch den betreffenden Plan benötigten Ressourcen aufgeführt werden können. Eine Ressource wird durch `<resource>` gekennzeichnet. Durch die drei notwendigen Attribute `value`, `ref` und `type` wird eine Ressource genauer spezifiziert. Dabei wird im Attribut `ref` dasjenige Belief referenziert, welches das Faktenwissen über die Menge einer vorhandenen Ressource beinhaltet. Mit dem Attribut `value` wird die durch den Plan benötigte Menge dieser Ressource angegeben.

Hier sind sowohl Zahlen wie auch Ausdrücke möglich, deren Typ sich an der Klasse des jeweiligen Beliefs orientieren sollten. Durch das Attribut `type` einer Ressource muss der Entwickler angeben, ob es sich bei dieser Ressource um eine wiederverwendbare (reuseable) Ressource oder um eine verbrauchbare (consumable) Ressource handelt. Zu beachten bei der Spezifikation von benötigten Ressourcen ist, dass für einen Plan nur diejenigen Ressourcen spezifiziert werden, die der Plan selbst durch die Ausführung seiner Aktionen benötigt. Ressourcen die von Plänen eventuell vorhandener Subziele benötigt werden, werden nicht innerhalb des die Subziele erzeugenden Plan aufgeführt. Die Angaben zu

```
<plans>
      <plan name="P1">
            <body>new Plan1()</body>
            <trigger>
                  <goal ref="g3"/>
            </trigger>
            <resources>
                  <resource value="2" ref="B" type="reuseable"/>
                  <resource value="1" ref="D" type="reuseable"/>
            </resources>
      </plan>
      <plan name="P2">
            <body>new Plan2()</body>
            <trigger>
                  <goal ref="g1"/>
            </trigger>
            <subgoals>
                  <subgoal ref="sg1"/>
            </subgoals>
      </plan>
</plans>
```

Abbildung 5.6: Spezifikationen am Plan, Auszug aus einer ADF

diesen Ressourcen werden für die Pläne der Subziele getätigt.

Damit ein Entwickler darstellen kann welche Subziele ein Plan theoretisch erzeugen kann, ist es möglich innerhalb eines `subgoals` tag auftretende Subziele zu definieren. Durch die Angabe eines `subgoal` tags spezifiziert man die Subziele. Mit dem notwendigen Attribut `ref` referenziert man den entsprechenden Zieltyp der durch diesen Plan erzeugt wird.

5.2.4 Struktur der Meta-Aktionen

Die Bestandteile des konstruierten Verfahrens sind in dem UML Klassendiagramm in Abbildung 5.7 dargestellt. Die gelb markierten Klassen stellen hierbei die Hauptkomponenten dar. Die in weiss belassenen Klassen sind vorhandene Klassen des Jadexsystems die durch für das Verfahren benötigte Methoden bzw. Attribute ergänzt wurden. Bereits erläutert wurden die Klassen *Resource* und *ResourceProcessor*, sowie die um Attribute bzw. Methoden ergänzten Klassen *RGoal* und *RGoalbase*. Nachfolgend sollen die Meta-Aktionen und ihre Umsetzung erläutert werden.

DeliberateGoalResourceSummaryAction Diese Meta-Aktion kapselt die Berechnung der Ressourceninformation eines Ziels. Dabei werden Ressourceninformationen nur für das als Attribut aufgeführte Ziel und die für dieses Ziel anwendbaren Pläne im Attribut plans berechnet. Die Berechnung der Informationen erfolgt so wie in 5.2.2 beschrieben. Um die Ressourceninformation eines Ziels zu berechnen, wurden Erweiterungen in den Klassen *MGoal* und *MPlan* notwendig. M-Klassen sind die Klassen die von

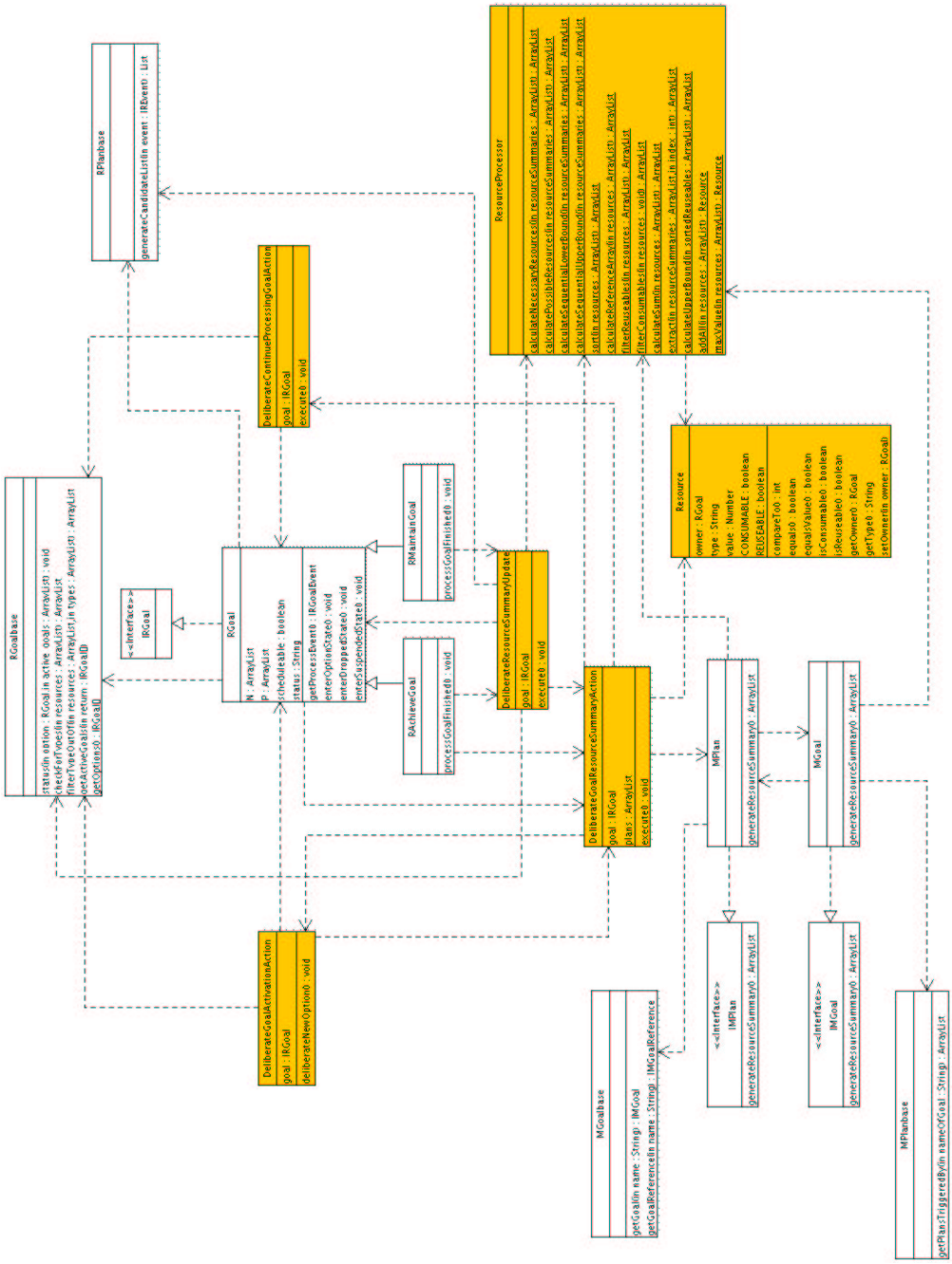

Abbildung 5.7: UML Klassendiagramm

dem „XML Data Binding Framwork“ jBind[1] erzeugt wurden und somit das Modell aus dem ADF in eine Klassenstruktur umsetzen. Die Erweiterung bestand in der Bereitstellung der Methode `generateResourceSummary`. Mithilfe dieser Methode kann ein Ziel bzw. ein Plan seine Ressourceninformation bestimmen. Für die Erweiterung wurden die M-Klassen für Ziele und Pläne ausgewählt, da ausser dem Ziel, für das die Ressourceninformation eigentlich bestimmt werden soll, kein anderes Konstrukt eine Klasse zur Laufzeit (R-Klasse) besitzt. Die Bestimmung läuft sozusagen auf dem Modell, welches im ADF definiert wurde.

Die berechneten Ressourceninformationen werden in den Attributen N für „necessary“ Ressourcen und P für „possible“ Ressourcen gespeichert.

DeliberateGoalActivationAction Innerhalb dieser Meta-Aktion greift das Verfahren auf die berechneten Ressourceninformationen in N und P des als Attribut spezifizierten Ziels zurück und berechnet für dieses den Status einer Aktivierung. Für Subziele eines aktiven Ziels und für Ziel vom Typ Maintain im Zustand „Idle“ wird kein Status berechnet. Beide werden ohne weitere Überprüfung in den Zustand „Active“ überführt.

DeliberateContinueProcessingGoalAction Diese Meta-Aktion ähnelt im Aufbau der Meta-Aktion DeliberateGoalActivationAction. Der Unterschied besteht jedoch im Kontext in dem diese Meta-Aktion angewendet wird. Denn diese Meta-Aktion ist nur für Ziele vorgesehen, deren Aktivierung mit dem Status „Scheduleable“ oder „Uncertain“ beurteilt wurde. Für diese Ziele bestand die Möglichkeit durch Auswahl eines bestimmten Plans (siehe Status „Uncertain“ oder „Schedulable“ in 5.1.3) sicher aktiviert zu werden. Sollte der so ausgewählte Plan aber fehlschlagen, so kann der Agent nicht einfach dahin übergehen einen neuen Plan zu suchen und zu bearbeiten. Er muss vielmehr darüber deliberieren, ob der nächste zur Verfügung stehende Plan auch noch eine sichere Bearbeitung des betreffenden Ziels zulässt. Das geschieht, indem der Agent den nächsten zu bearbeitenden Plan in die Bestimmung der Ressourceninformation des Ziels einbezieht. Ist das Ziel mit dieser neuen Ressourceninformation sicher, so wird es weiter bearbeitet. In den anderen Fällen außer dem Fall „conflicting“ werden die betreffenden Pläne übergangen und ein neuer Plan ausgewählt. Für den Fall, dass der betreffende Plan einen Konflikt auslösen würde, so wird dieser als nicht mehr anwendbar deklariert. Es wird dann ein weiterer Plan gesucht. Sollte der Agent keine anwendbaren Pläne mehr besitzen oder hat er bereits alle anwendbaren Pläne erfolglos geprüft, so wird das betreffende Ziel im ersten Fall deaktiviert und in den Zustand „Dropped“ überführt und im zweiten Fall auch deaktiviert, aber in den Zustand „Option“ überführt.

DeliberateGoalResourceSummaryUpdateAction Hier wird für ein angegebenes Ziel im Attribut goal geprüft, ob der gerade fehlgeschlagene Plan

[1] www.jBind.org

eine, im Vergleich zu den in den anderen für dieses Ziel zur Verfügung stehenden Plänen, maximale Ressource benötigt hat. Trifft dies zu, so aktualisiert diese Meta-Aktion zugleich die gesamte Ressourceninformation für das betreffende Ziel. Anschließend wird über eine Aktivierung von im Zustand „option“ befindlichen Zielen deliberiert. Diese Ziele könnten durch eine Aktualisierung der Ressourceninformation profitieren.

5.3 Bewertung

Das implementierte Verfahren muss natürlich in den Kontext bestehender Arbeiten eingeordnet werden. Dazu wird es mit X-JACK, einer Implementation welche ebenfalls die Idee aus [TWPF2002b] aufgegriffen hat und mit der „Easy Deliberation“ Strategie verglichen. Dadurch soll aufgezeigt werden, welche Vorteile diese Implementation gegenüber einer ähnlichen Implementation aufweist und welche Vorteile bzw. Nachteile dieses Verfahren gegenüber einem konzeptionell anderen Verfahren zeigt.

5.3.1 Vergleich mit „X-JACK“

X-JACK [TP2005] ist eine Erweiterung der Agentenplattform JACK. Die Erweiterung besteht in der Möglichkeit deliberativ auf Zielen eines Agenten tätig zu werden. Deliberiert wird nach der Idee aus [TWPF2002b] auf Ressourcen, die von den Plänen der Ziele benötigt werden. JACK besitzt kein erweitertes Zielkonzept wie Jadex und es ist deshalb umso interessanter festzustellen, wie sich zwei ähnliche Verfahren in Agentenplattformen implementieren ließen, die sich so eklatant in diesem wichtigen Punkt unterscheiden. Soweit dies möglich war wird das in dieser Arbeit implementierte Verfahren mit dem Verfahren welches in [TP2005] beschrieben wird verglichen. Es konnten keine konkreten Implementierungsdetails eingesehen werden, so dass der hier beschriebene Vergleich eher knapp ausfallen wird.

Grundlegende Datenstrukturen

Im Unterschied zu der Implementation dieser Arbeit, benötigte X-JACK eine Erweiterung der Repräsentation seiner Ziele. X-JACK spezifiziert deshalb für ein jedes Ziel eines Agenten einen sogenannte „Goal-plan Tree“. Weiterhin wurde in X-JACK ein „Resource Manager“ eingeführt, der die Zugriffe auf die Ressourcen simulierte. Beide Konstrukte wurden in dem hier implementierten Verfahren nicht aufgegriffen. Zum einen besitzt Jadex schon ein erweitertes Zielkonzept um die notwendigen Vorbedingungen für das Deliberationsverfahren zu erfüllen, andererseits wurde absichtlich von der Verwendung eines „Resource Manager“ abgesehen. Das hier implementierte Verfahren verwendete ausschließlich interne Referenzen auf Beliefs um benötigte Ressourcen zu deklarieren. Die Notwendigkeit für die Spezifikation eines eigenen Prozesses dafür wurde nicht gesehen. Ebenso nicht implementiert wurde das für X-JACK konstruierte „Goal

Management System“. Dieses Management System ist kurz gesagt für die Erzeugung der „Goal-plan Trees“ für jeden Typ Ziel verantwortlich. Das war für Jadex ebenfalls nicht notwendig.

Details des Verfahrens

Beide implementierten Verfahren verwenden die in [TWPF2002b] angesprochenen Zustände des Status der Aktivierung eines Ziels. X-JACK berücksichtigt in der vorliegenden Ausführung nur die Zustände „Safe“, „Conflicting“, „Schedulable“ und „Uncertain“. Für die beiden erstgenannten Zustände sind die Reaktionen beider Implementation identisch. Im dem einen Fall werden Ziele aktiviert im anderen nicht. Für den Status „Schedulable“ schlägt X-JACK einen einfachen Weg ein. Es erlaubt Zielen mit solch einem Status erst aktiviert zu werden wenn sie definitiv „Safe“ sind. In dem in dieser Arbeit implementierten System, wird ein anderer Weg beschritten. Wie bereits im Kapitel 5.1.3 beschrieben, werden Ziele im Zustand „Schedulable“ nur, unter der Auflage der Auswahl eines bestimmten Plans, ausgewählt. Das bedeutet für solche Ziele, dass sie unter Vorbehalt aktiviert werden und ihre Bearbeitung überwacht wird. Ebenso gilt das für den Zustand „Uncertain“. Hier werden Ziele nur unter Vorbehalt der spezifischen Planauswahl aktiviert. X-JACK aktiviert Ziele eines solchen Status nicht. Ein weiterer wichtiger Unterschied ist, dass X-JACK mit hoher Wahrscheinlichkeit[2] dieses Verfahren nur für Ziele vom Typ Achieve implementiert. Das in dieser Arbeit implementierte Verfahren hingegen verwendet die Deliberation auch für Ziele vom Typ Maintain. Welche besonderen Bedingungen dabei gelten ist in Kapitel 5.1.4 nachzulesen.

Zusammengefasst kann gesagt werden, dass das in dieser Arbeit implementierte Verfahren, zu einem grossen Anteil von den Konstrukten Ziel und Datenstrukturen wie dem variablen Interpreter profitiert hat.

5.3.2 Vergleich mit der Easy Deliberation Strategie

Spezifikation

Im Gegensatz zur „Easy Deliberation“ Strategie bindet das hier implementierte Verfahren explizit die Berücksichtigung von Interaktionen zwischen Plänen eines Ziels in den Deliberationsprozess ein. Diese Interaktionen werden dazu verwendet um auf der Ebene der Ziele negative Interaktionen erkennen zu können. Im Gegensatz zur „Easy Deliberation“ Strategie, ist während der Spezifikation der Information die notwendig für das Verfahren ist, nicht erkennbar, welche Konflikte sich auf der Ebene der Ziele ergeben können. Das Verfahren unterstützt also keine direkte Ausdrucksmöglichkeit für Interaktionen, sondern berechnet diese erst.

[2] Der Autor dieser Diplomarbeit kann sich nur an die ihm zur Verfügung stehende Literatur halten. Somit interpretiert er aufgrund der Ausführungen in [TWPF2002b] das es sich bei den dort angesprochenen Zielen um Ziele vom Typ Achieve handelt.

Das Verfahren besitzt ein erhöhte Fehleranfälligkeit gegenüber der „Easy Deliberation“ Strategie. Dadurch das ein Entwickler eines Agenten nicht auf Anhieb nachvollziehen kann, ob Ziele die bestimmte Ressourcen benötigen, parallel oder sequentiell ausgeführt werden können, besteht die Gefahr, dass Fehler die während der Spezifikation der Ressourcen entstehen, unbemerkt bleiben. Dies wird auch in [DvdT2004] bestätigt. Dort wird zudem noch ausgeführt, dass die Spezifikation solcher Ressourceninformation aufwendig ist. Wenn man z.B. berücksichtigt, dass für ein Ziel mindestens zwei Pläne zur Verfügung stehen und diese Pläne Ressourcen benötigen. So ist um einen Konflikt zwischen zwei solchen Zielen erkennen zu können, ein erheblicher Mehraufwand an Spezifikation zu leisten, als für die vergleichbare Leistung mit der „Easy Deliberation“ Strategie.

Das hier implementierte Verfahren unterstützt nicht das Capability Konzept. Es besteht also nicht die Möglichkeit Plankonflikte von Zielen verschiedener Wirkungsbereiche zu Erkennen.

Entscheidungsmöglichkeiten

Das hier implementierte Verfahren gibt im Unterschied zur „Easy Deliberation“ Strategie keine lokale Ordnung zwischen den Zielen an. Das bedeutet, dass dieses Verfahren nur eine Empfehlung zur parallelen oder sequentiellen Abarbeitung der Ziele ausspricht, nicht aber, im Fall der sequentiellen Bearbeitung, eine Reihenfolge der Elemente der Sequenz festlegt. Vielmehr ist das hier beschriebene Verfahren ein emergentes Verfahren dem während der Spezifikation der verwendeten Information nicht unbedingt angesehen werden kann, welche Ziele in Konflikt stehen könnten.

Jedoch kann dieses Verfahren im Unterschied zur „Easy Deliberation“ Strategie nicht nur die binären Entscheidungen parallele oder sequentielle Bearbeitung treffen, sondern auch Mischformen, hier bezeichnet als „0,5 Entscheidungen“. Bei einer solchen Entscheidung gibt das Verfahren keine einhundertprozentige Entscheidung für eine Seite an, es sagt damit vielmehr aus, dass es unter Umständen möglich sein kann dieses Ziel parallel mit anderen zu bearbeiten oder nur sequentiell.

Kapitel 6

Fazit und Ausblick

Diese Arbeit beschritt das sehr spezielles Gebiet der Goal Deliberation, innerhalb der Thematik der Multiagentensysteme. Dieses Gebiet ist noch ein reines Forschungsthema und entsprechend übersichtlich waren die Veröffentlichungen dazu. So wurde in [PBL05a] dazu vermerkt, dass diesem Thema noch nicht viel Aufmerksamkeit zugeteilt worden ist. Als Indiz dafür, wird dort das Fehlen eines explizit vorhandenen Zielkonzeptes, wie es in Jadex vorliegt, angeführt. Dies kann durch die Erkenntnisse die in dieser Arbeit erlangt wurden, bestätigt werden. Vielfach begegneten während der Literaturrecherche Systeme, die einen sehr einfachen Ansatz entwickelten um zwischen den Zielen eines Agenten entscheiden zu können. Dies beruhte immer auf dem Vorhandensein eines sehr einfachen Zielkonstruktes. Nur in wenigen Fällen, siehe Kapitel 4.3, wurden komplexere Ansätze vorgeschlagen. Diese gingen immer einher, mit einer Erweiterung des bestehenden Zielkonzeptes. Auch zu beobachten war, dass der Deliberation auf Zielen die Notwendigkeit eines eigenständiges Dasein nicht attestiert wurde. Deliberation auf Zielen wurde insbesondere im Bereich des Planens, welcher einen erheblichen Anteil in der künstlichen Intelligenz ausmacht, in den Prozess der Konstruktion von Plänen integriert. Somit war die Notwendigkeit einer Deliberation auf dem Konstrukt Ziel nicht gegeben.

Diese Arbeit hat aufgezeigt, wie auf der Grundlage eines erweiterten Zielkonzeptes, ein Deliberationsverfahren aufsetzen kann, dass komplexe Interaktionen zwischen Zielen erkennen und entsprechend behandeln kann. Dabei wurde ausführlich die Grundlage für solch ein Deliberationsverfahren, das Konzept Ziel, beschrieben. Es wurden Aspekte des Konzeptes erläutert, die zum Teil prädestinierender für einen deliberativen Einsatz waren (siehe deklarativer Aspekt), Eigenschaften von Zielen aufgeführt die Seitens der Theorie an das Konzept gestellt wurden und verschieden Attribute aufgezählt, die das Konzept Ziel zu einem vollwertigen Konstrukt werden ließen, welches eine Deliberation erheblich begünstigt.

Im Anschluß daran wurden verschiedene Deliberationsverfahren vorgestellt. Hier handelt es sich sowohl um aktuelle Trends der Forschung wie aber auch um

ältere Ansätze. Für die Verwendung des erweiterten Zielkonzeptes und aufgrund der gesteigerten Entscheidungsmöglichkeiten wurden die Ansätze basierend auf der „Summary“ Information in die engere Wahl genommen. Dabei wurde das Verfahren der „Resource Summary“ favorisiert, weil es ein Verfahren darstellte welches eine vielseitige Anwendbarkeit versprach und ergänzend zu dem bestehenden Verfahren in Jadex sein könnte.

Die Implementation zeigte schließlich, dass der Einsatz des „Resource Summary“ Verfahren auch praktisch möglich ist. Es wurden die Konzepte aus dem beschriebenen Verfahren umgesetzt und einige eigenständige Erweiterungen eingeführt. So musste das „Agent Definition File“ an die veränderten Spezifikationsanforderungen angepasst werden und Klassen des Jadexsystems dahingehend modifiziert werden das eine Repräsentation und Berechnung von „Resource Summaries“ möglich wurde. An kleinen Beispielen die nicht in dieser Arbeit erwähnt wurden, ist das Verfahren erprobt wurden.

Abschließend wurde das implementierte Verfahren mit einem ähnlichen, auf „Resource Summary“ basierenden, Verfahren und mit dem bereits in Jadex implementierten „Easy Deliberation“ Verfahren verglichen. Dabei stellte sich, speziell im Fall des Vergleichs mit der „Easy Deliberation“ Strategie, heraus, dass das „Resource Summary“ Verfahren ein potentielles weiteres mögliches Verfahren zur Deliberation auf Zielen darstellt.

6.1 Ausblick

Die in Jadex implementierten Deliberationsverfahren decken ein breites Spektrum an Anwendungsdomänen ab. Für die Zukunft muss gezeigt werden, dass die implementierten Verfahren, insbesondere das hier in dieser Arbeit vorgestellte, relevante Verbesserung im Umgang mit Zielen eines Agenten in komplexen und umfangreichen Anwendungen bieten können. Ebenso muss das volle Potential eines Goal Deliberationsverfahren erforscht werden. Dazu ist es notwendig das der Schritt von der Deliberation innerhalb eines Agenten hinzur Deliberation zwischen Agenten vollzogen wird. Dort muss dann evaluiert werden inwieweit die innerhalb eines Agenten verwendeten Konzepte zur Deliberation, positive Auswirkungen auf die Interaktionen zwischen Agenten haben können. Insbesondere die Idee der Ressourcen bietet hier eine geeignete Grundlage für möglichen Informationsaustausch zwischen Agenten.

Abbildungsverzeichnis

Literaturverzeichnis

[Bea94] L. Beaudoin: *Goal processing in autonomous agents.* PhD Dissertation, University of Birmingham, School of Computer Science, 1994.

[BS93] L. Beaudoin, A. Sloman: *A study of motive processing and attention.* In: A. Sloman, D. Hogg, G. Humphreys, A. Ramsay, D. Partridge (editors): In Prospects for Artificial Intelligence: Proceedings of AISB-93. IOS Press, Amsterdam, 1993, pages 229–238.

[Bra87] M. Bratman: *Intentions, Plans, and Practical Reasoning.* Cambridge, Massachusetts, and London, England: Harvard University Press, 1987.

[BIP88] M. Bratman, D. Israel, and M. Pollack: *Plans and resource-bounded practical reasoning.* In: Computational Intelligence. Vol. 4: 1988, 349–355.

[BPML04] L. Braubach, A. Pokahr, W. Lamersdorf, D. Moldt: *Goal Representation for BDI Agent Systems.* In: Rafael H. Bordini, Mehdi Dastani, Jürgen Dix, Amal El Fallah-Seghrouchni (editors): Second International Workshop on Programming Multi-Agent Systems: Languages and Tools (ProMAS 2004), New York, NY, USA, July 2004.

[BP2005] L. Braubach, A. Pokahr: *Jadex-Project* Internet `http://vsis-www.informatik.uni-hamburg.de/projects/jadex`

[Bro86] R. A. Brooks: *A robust layered control system for a mobile robot.* In: IEEE Journal of Robotics and Automation,2(1), pages 14–23.

[BRHL99] P. Busetta, R. Rönnquist, A. Hodgson, and A. Lucas: *JACK Intelligent Agents – Components for Intelligent Agents in Java.* In: AgentLink News Letter, January, 1999.

[CD99] B. J. Clement and E. H. Durfee: *Theory for coordinating concurrent hierarchical planning agents using summary information.* In: Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence (AAAI '99/IAAI '99), Orlando, Florida, United States, American Association for Artificial Intelligence, Menlo Park, CA, USA, 1999, pages 495–502.

[CL93] P. R. Cohen, H. J. Levesque: *Intention is choice with commitment.* In: Aritficial Intelligence 42. Elsevier Science Publishers Ltd., Essex, UK, 1990, pages 213–261.

[DvLF93] A. Dardenne, A. van Lamsweerde and S. Fickas: *Goal-directed Requirements Acquisition.* In: 6IWSSD: Selected Papers of the Sixth International Workshop on Software Specification and Design. Elsevier Science Publishers B. V., Amsterdam, The Netherlands, 1993, pages 3–50.

[DdBDM2003a] M. Dastani, F. de Boer, F. Dignum, J.-J. Meyer: *Programming Agent Deliberation: An Approach Illustrated Using the 3APL Language.* In: Proceedings of the Second International Conference on Autonomous Agents and Multiagent Systems (AAMAS'03), Melbourne, July 2003, ACM Press, 2003, pages 97–104.

[DvRDM2003b] M. Dastani, B. van Riemsdijk, F. Dignum, J.-J. Ch. Meyer: *A Programming Language for Cognitive Agents – Goal Directed 3APL* In: Proceedings of the First Workshop on Programming Multiagent Systems: Languages, frameworks, techniques, and tools (ProMAS03) to be held at AAMAS'03, Melbourne, July 2003, 2003.

[DvdT2004] M. Dastani and L. van der Torre: *Programming BOID-Plan Agents: Deliberating about Conflicts among Defeasible Mental Attitudes and Plans.* In: AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems. IEEE Computer Society, Washington, DC, USA, 2004, pages 706–713.

[Fer92] I. A. Ferguson: *TouringMachines: An Architecture for Dynamic, Rational, Mobile Agents.* PhD Dissertation, University of Cambridge, 1992.

[FN71] R. E. Fikes, N. Nilsson: *STRIPS: A new approach to the application of theorem proving to problem solving.* In: Artificial Intelligence, 5(2), pages 189–208.

[GMP2002] F. Giunchiglia, J. Mylopoulos, and A. Perini: *The Tropos Software Development Methodology: Processes, Models and Diagrams.* In: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'02). ACM Press, July 2002.

[Hub99] M. J. Huber: *JAM: A BDI-theoretic Mobile Agent Architecture.* In: Proceedings of the third annual conference on Autonomous Agents, Seattle, Washington, United States. 1999, pages 236–243.

[Hub2001] M. J. Huber: JAM-Manual. Internet `http://www.marcush.net/IRS/irs_ downloads.html`, 2001.

[HdBvdHM98] K. V. Hindriks, F. S. de Boer, W. van der Hoek, and J.-J. Ch. Meyer: *Failure, Monitoring and Recovery in the Agent Language 3APL.* In: Fall Symposium on Cognitive Robotics (AAAI), 1998, pages 68–75.

[HdBvdHM99] K. V. Hindriks, F. S. de Boer, W. van der Hoek, and J.-J. Ch. Meyer: *Agent Programming in 3APL.* In: Autonomous Agents and Multi-Agent Systems, 2(4):357-401, 1999.

[Jen99] N. R. Jennings: *Agent-based Computing: Promise and Perils.* In: Proc. 16th Int. Joint Conf. on Artificial Intelligence (IJCAI-99). Stockholm , Sweden: 1999, pages 1429–1436.

[Kir91] D. Kirsh: *Today the earwig, tomorrow man.* In: Artificial Intelligence 47. Elsevier Science Publishers Ltd., Essex, UK, 1991, pages 161–184.

[Loy97] A. B. Loyall: *Believable Agents: Building Interactive Personalities.* PhD thesis, Carnegie Mellon University Pittsburgh, School of Computer Science, 1997.

[MP93] J. P. Müller, M. Pischel: *The Agent Architecture INTERRAP: Concept and Application.* German Research Center for Artificial Intelligence (DFKI), Saarbrücken, 1993.

[NL96] T. J. Norman, D. Long: *Alarms: An implementation of motivated agency.* In: M. Wooldridge, J.P. Muller, and M. Tambe (editors): Intelligent Agents: Theories, Architectures, and Languages, LNAI 1037 Springer, 1996, pages 219-234.

[Nor97] T. J. Norman: *Motivation-based direction of planning attention in agents with goal autonomy.* PhD thesis, Department of Computer Science, University College London, 1997.

[PL2000] L. Padgham and P. Lambrix: *Agent Capabilities: Extending BDI Theory.* Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence. AAAI Press / The MIT Press, 2000, pages 68–73.

[PBL05a] A. Pokahr, L. Braubach, W. Lamersdorf: *A BDI Architecture for Goal Deliberation*, In: Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'05), Poster, to appear.

[PBL05b] A. Pokahr, L. Braubach, W. Lamersdorf: *Jadex: A BDI Reasoning Engine*, Chapter of Multi-Agent Programming, Kluwer Book, Editors: R. Bordini, M. Dastani, J. Dix and A. Seghrouchni, to appear.

[Pol92] M. E. Pollack: *The uses of plans.* In: Artificial Intelligence. Vol.57. Elsevier Science Publishers Ltd, 1992, S. 43–68

[RG91] A. S. Rao, M. P. Georgeff: *Modeling Rational Agents within a BDI-Architecture.* In: J. Allen, R. Fikes, and E. Sandewall (editors): Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR'91). Morgan Kaufmann Puplishers, San Mateo, 1991, pages 473-484.

[Sho92] Y. Shoham: *Agent-oriented programming.* In: Artificial Intelligence Vol. 60, Elsevier Science Publishers Ltd., Essex, UK, 1993, pages 51–92

[SRG99] M. P. Singh, A. S. Rao, M. P. Georgeff: *Formal Methods in DAI: Logic-Based Representation and Reasoning.* In: Gerhard Weiss (Editor): Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. Cambridge, Massachusetts, London, England, The MIT Press, S. 331–376, 1999.

[SBW94] A. Sloman, L. P. Beaudoin, & I. P. Wright: *Computational modelling of motive-management processes.* Cognitive Science Research Report CSRP-94-14, School of Computer Science and Cognitive Science Research Centre, University of Birmingham, 1994.

[TPH2002a] J. Thangarajah, L. Padgham, and J. Harland: *Representation and reasoning for goals in BDI agents.* In: Michael Oudshoorn (editor): Proceedings of the Twenty-Fifth Australasian Computer Science Conference (ACSC 2002), Melbourne, Australia, 2002.

[TPW2003] J. Thangarajah, L. Padgham, M. Winikoff: *Dectecting & Avoiding Interference Between Goals in Intelligent Agents.* In: Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003), Acapulco, Mexico, 2003.

[TPW2003b] J. Thangarajah, L. Padgham, M. Winikoff: *Detecting & Exploiting Positive Goal Interaction in Intelligent Agents.* In: Proceedings of the second international joint conference on Autonomous agents and multiagent systems (AAMAS 2003), Melbourne, Australia, 2003. New York, NY, USA: ACM Press, 2003, S. 401–408.

[TP2005] J. Thangarajah, L. Padgham: *An empirical evaluation of reasoning about resource conflicts in Intelligent Agents.* In: Third International Joint Conference on Autonomous Agents and Multiagents Systems (AAMAS 2004), to appear.

[TWPF2002b] J. Thangarajah, M. Winikoff, L. Padgham, and K. Fischer: *Avoiding resource conflicts in intelligent agents.* In: Proceedings of the 15th European Conference on Artificial Intelligence 2002 (ECAI 2002), Lyon, France, 2002.

[Til2005] Tilab: JADE Hompage. Internet `http://jade.cselt.it/`

[vLam2001] A. van Lamsweerde: *Goal-Oriented Requirements Engineering: A Guided Tour.* In: Proceedings of the 5 th International Symposium on Requirements Engineering (RE'01), Toronto, Canada, August, 2001. IEEE Computer Society Press, 2001, pp. 249-263.

[vRvdHM2003] M.B. van Riemsdijk, W. van der Hoek, J-J.Ch. Meyer: *Agent programming in Dribble: from beliefs to goals with plans.* In: Proceedings of

the second international joint conference on autonomous agents and multiagent systems (AAMAS'03). Melbourne, pp. 393–400.

[Wei2000] G. Weiss: *Prologue.* In: Gerhard Weiss (Editor): Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. Cambridge, Massachusetts, London, England, The MIT Press, S. 1–9.

[Wil83] R. Wilensky: *Planning and Understanding: A Computational Approach to Human Reasoning.* Reading, Massachusetts: Addison-Wesley Publishing Company, 1983.

[WPH2001] M. Winikoff, L. Padgham, J. Harland: *Simplifying the Development of Intelligent Agents.* In: AI '01: Proceedings of the 14th Australian Joint Conference on Artificial Intelligence, Springer-Verlag, London, UK, 2001, pages 557–568.

[WPHT2002] M. Winikoff, L. Padgham, J. Harland, J. Thangarajah: *Declarative & Procedural Goals in Intelligent Agent Systems.* In: Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR2002), Toulouse, France, April, 2002.

[Woo2002] M. Wooldridge: *An introduction to multiagent systems.* Chichester, England: John Wiley & Sons, 2002.

[WJ95] M. Wooldridge, N. R. Jennings: *Intelligent Agents: Theory and Practice.* In: The Knowledge Engineering Review. Vol. 10: 1995, S. 115–152

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. Ich bin mit einer Einstellung in den Bestand der Bibliothek des Fachbereiches einverstanden.

Hamburg, den 22. Juli 2005

Tobias Blasche