Diplomarbeit

Konzeption und Realisierung einer Toolbox statischer Kontrollmethoden zur Steuerung eines Causal Link Planers

Vera Kettnaker

August 1995

Betreuung: Prof. Dr. Michael M. Richter
Dipl. Inform. Reinhard Praeger

Arbeitsgruppe Künstliche Intelligenz
–Prof. Dr. Michael M. Richter–
Fachbereich Informatik
Universität Kaiserslautern

Inhaltsverzeichnis

Kapitel 1

Einführung

1.1 Einführung

CAPlan[Web94] basiert auf dem SNLP-Algorithmus [BW92], einer bekannten Version des Originalalgorithmus von McAllester und Rosenblitt [MR91]. Seit dessen Veröffentlichung hat eine Flut von Papieren die Überlegenheit dieses Planers gegenüber herkömmlichen Planverfahren gezeigt. Diese Überlegenheit liegt einerseits in seinem nur partiell ordnenden least-commitment Ansatz begründet und anderseits in der Systematik der Suche, die einen minimalen Suchraum garantiert.

In letzter Zeit haben allerdings vermehrt Veröffentlichungen darauf hingewiesen, daß nicht so sehr die Größe des Suchraumes an sich, sondern vielmehr die Größe des *durchsuchten* Raumes ausschlaggebend für die Planungseffizienz sind, und Beispiele und Bedingungen angegeben, für die totalordnende, heuristische, oder unsystematische Planverfahren SNLP überlegen sein können. Die praktischen Erfahrungen mit SNLP in komplexeren Domänen zeigen ebenfalls, daß auch bei diesem Algorithmus effiziente Kontrollverfahren Grundvoraussetzung für den praktischen Einsatz sind.

1.1.1 Zielsetzung einer Basissteuerung

Thema dieser Arbeit ist eine Basissteuerung von SNLP durch statische Kontrollverfahren, also durch nicht-lernende Verfahren. Dazu zählen insbesondere

- Heuristiken
- Verfahren, die auf einer einmaligen Analyse basieren
- Verfahren, die Teilaspekte des Planungsproblems isolieren und separat lösen

Wie sinnvoll ist eine Basissteuerung?

Es wäre sicherlich möglich, lernende Kontrollverfahren auf den Basisplaner anzuwenden, ohne eine Basissteuerung dazwischenzuschalten. Gemeinsames Problem aller lernenden Kontrollverfahren ist aber das Utility-Problem[Min88], also die Schwierigkeit, nur solche Regeln zu lernen, die mehr Planarbeit sparen, als sie Kosten verursachen. Das Utility-Problem ist vor allem deshalb so kritisch, weil lernende Verfahren oft ganz spezifische Regeln lernen und von diesen nur ungenügend abstrahieren können. Häufig werden so sehr viele Instanzen einer allgemeinen Heuristik gelernt, ohne jemals das dahinterliegende Prinzip zu entdecken. Problematisch daran ist, daß dies die Berechnung von Kontrollinformation unnötig teuer macht, weil mehr Regeln überprüft werden müssen und die Tests durch die höhere Spezifität der Regeln auch teurer auszuwerten sind.

In ihrem Artikel *'Is there any Need for Domain-Dependent Control Information?'* [MLG91] sprechen sich Ginsberg und Geddis gegen solche domänenspezifische Kontrollregeln aus und plädieren

für domänenunabhängige Kontrollverfahren, die auf Domänenwissen und Modalwissen[1] operieren. Allerdings vernachlässigen sie in ihrem Artikel völlig den Effizienzaspekt, der ja ursprünglich zur Entwicklung von EBL-Verfahren geführt hat: Kontrollwissen soll durch Precompiling soweit instantiiert werden, daß es effizient anwendbar ist.
Es scheint daher ein guter Kompromiß zu sein, lernende Verfahren nicht dazu zu verwenden, überspezifische Instanzen allgemeiner Heuristiken zu generieren, sondern vielmehr dazu, die *Ausnahmen* dieser Grundstrategien zu erlernen. Solch eine Vorgehensweise kann durch eine Kombination einer Basissteuerung mit durch Fehlschlag getriggerten Lernverfahren realisiert werden.

Für den Einsatz einer Basissteuerung spricht auch die Tatsache, daß die meisten Lernverfahren fertige Pläne als Eingabe benötigen. Die praktische Erfahrung hat jedoch gezeigt, daß der Suchraum selbst für sehr kleine Probleme so groß wird, daß der Basisplaner nicht in der Lage ist, in akzeptabler Zeit eine Lösung zu finden. Wird der Planer daher mittels Basissteuerung in die Lage versetzt, eine größere Menge von Problemen selbständig zu lösen, dann stehen mehr und vor allem auch komplexere Beispiele zur Verfügung, von denen gelernt werden kann. Darüberhinaus gilt für alle lernenden Verfahren, daß sie bei der Konfrontation mit neuen Situationen auf den Basisplaner angewiesen sind. Insbesondere für einen Assistenzplaner wie CAPlan ist wichtig, daß der Planer auch in solchen Fällen 'vernünftig' vorgeht.

Zudem gilt, daß durch EBL-Verfahren nur die Kontrolle solcher Entscheidungen lernbar ist, die in SNLP als Backtrackingpunkte ausgezeichnet sind. Die Bearbeitungsreihenfolge von Zielen und Threats, die sowohl auf die Effizienz des Planungsprozeß als auch auf die Qualität der entstehenden Pläne starken Einfluß hat, muß daher durch alternative Kontrollverfahren, wie etwa den hier vorgestellten statischen Methoden, gesteuert werden.

Bei der Diskussion statischer Kontrollmethoden ist schließlich zu beachten, daß sich Heuristiken etwas zu Unrecht zu Stiefkindern der Forschung entwickelt haben. Im Sprachgebrauch der Planung sind Heuristiken billige, schwache Kontrollverfahren. Es wird aber häufig übersehen, daß plausible Heuristiken das Utility-Problem optimal lösen und daher überraschend stark sein können. Minton, Bresina und Drummond [MBD94] berichten etwa, daß durch die Verwendung einer einzigen Heuristik[2] die Effektivtät um 80% gesteigert werden konnte. Diese Heuristik genügte ebenfalls aus, um einen totalordnender Planer effizienter als sein partiell ordnendes Pendant zu machen. Blythe [Bly93] berichtet von einer Effektivitätssteigerung um 50% durch die Verwendung von Heuristiken. Trotz oder gerade wegen ihrer Einfachheit scheinen also Heuristiken in der Lage zu sein, das Kontrollproblem zu entschärfen.

1.1.2 Toolbox-Ansatz statt universeller Kontrolle

Obwohl diese Arbeit zunächst darauf ausgelegt war, eine universelle Steuerung für CAPlan zu finden, hat sich im Laufe der Zeit immer mehr die Einsicht durchgesetzt, daß es nicht möglich ist, eine universell einsetzbare und dennoch mächtige Basiskontrolle zu finden. Das Hauptproblem besteht darin, daß mächtige Kontrollverfahren meist teuer sind und auf die Behandlung einer ganz spezifischen Klasse von Schwierigkeiten ausgelegt ist. Domänenunabhängige Planer werden aber für ein breites Spektrum von Domänen eingesetzt, die völlig unterschiedliche Schwierigkeitsstrukturen aufweisen. Daraus folgt unmittelbar, daß es zwar möglich ist, den Lösungsvorgang für Probleme *einer* Domäne effizient zu unterstützen, daß es aber sehr schwierig ist, den unterschiedlichen Schwierigkeitsprofilen verschiedener Domänen gleichermaßen Rechnung zu tragen. Stone, Veloso und Blythe [SVB94] haben ganz ähnliche Erfahrungen gemacht und an Beispielen dokumentiert.

Diese Arbeit wird daher statt einer universellen Kontrollkomponente ein Arsenal unterschiedlichster Kontrollverfahren zur Verfügung stellen, die gut dokumentiert und einfach zu kombinieren sind und so das Zusammenstellen individueller Kontrollkomponenten ermöglichen.

---

[1] Ginsberg und Geddis verwenden diesen Begriff in der Bedeutung 'Metawissen'.
[2] Die Heuristik heißt 'min-goals' und wird in Abschnitt 4.3.3.4 beschrieben.

1.1.3 Ziel dieser Arbeit

Die vorliegende Arbeit verfolgt zwei Hauptziele: Zum einen sollen für CAPlan verschiedene Kontrollverfahren realisiert werden, aus denen eine effiziente und effektive Basiskontrolle zusammengestellt werden kann. Ein weiteres Hauptziel wird sein, auch eine theoretische Basis für die Steuerung von CAPlan zu legen, in dem der etwas diffuse Kontrollbegriff stärker erhellt und strukturiert wird.

Dazu zählen:

- ein umfassendes Verständnis des Planers und seiner Eigenschaften
- die Identifikation prinzipieller Kontrollmöglichkeiten
- die Spezifikation von Anforderungen und Zielen einer Kontrollkomponente
- die Klassifikation von Kontrollverfahren
- und schließlich die Bestimmung von Möglichkeiten, verschiedene Kontrollverfahren miteinander zu kombinieren.

1.2 Übersicht

Zunächst einige generelle Hinweise.

- Vom Leser wird die Grundkenntnis genereller Planungskonzepte vorausgesetzt. Eine gute Einführung in das partiellordnende Planen bietet [Wel94]; für eine ausführlichere Darstellung des CAPlan-Systems sei auf [Web94] verwiesen.
- Vor allem in den ersten zwei Kapiteln wird eine ganze Reihe von Fachbegriffen eingeführt werden, die Grundlage für das Verständnis der nachfolgenden Kapitel sind. Am Ende der Arbeit findet sich ein Index, der zu den verschiedenen Begriffen die Stellen angibt, an denen sie definiert worden sind, um so ein schnelles Nachschlagen zu ermöglichen.
- Diese Arbeit soll einen Grundstein für nachfolgende Arbeiten legen. An einigen Stellen wird deshalb auf Verfahren und Konzepte verwiesen werden, die außerhalb des eigentlichen Rahmens dieser Arbeit liegen. Um meinen Nachfolgern Sucharbeit zu ersparen, wurden auch für diese Gebiete Hinweise auf relevante Literatur gegeben.

Nun zur eigentlichen Übersicht: In **Kapitel 2** wird zunächst kurz der Algorithmus SNLP sowie die CAPlan spezifischen Erweiterungen wiederholt. In **Kapitel 3** wird das Profil des Kontrollproblems für diesen Planer herausgearbeitet. Dazu werden in einem ersten Schritt Anforderungen und Beschränkungen an eine Kontrollkomponente formuliert und in einem zweiten Schritt der Raum denkbarer Kontrollmöglichkeiten ausgelotet. Ferner wird der mit den Anforderungen verträgliche methodische Spielraum festgelegt. Aufbauend auf dieser Analyse werden in **Kapitel 4** für jeden der Kontrollpunkte eine Reihe bekannter und neuer Kontrollverfahren vorgestellt und dokumentiert. **Kapitel 5** beschäftigt sich mit der Kombinierbarkeit mehrerer Verfahren an einem Kontrollpunkt und den Interdependenzen von Verfahren an verschiedenen Kontrollpunkten. Dieses Kapitel enthält auch Hinweise für eine gezielte Konfiguration von Kontrollkomponenten und demonstriert deren Einsatz anhand von einigen Beispieldomänen. **Kapitel 6** beschließt die Arbeit mit einer Zusammenfassung und einem Ausblick.

Kapitel 2

CAPlan - ein Causal Link Planer

Das CAPlan-System[Web94] kombiniert den SNLP-Algoritmus mit der Abhängigkeitsverwaltung REDUX [Pet92, Pet91]. Der SNLP-Algorithmus operiert auf nur partiell geordneten Plänen und eröffnet dadurch einige Freiheitgrade bei der Modifikation von Plänen. Zudem wird durch diese Planrepräsentation der Einsatz mächtiger Backtrackingverfahren wesentlich erleichtert. Die explizite Verwaltung von Entscheidungsabhängigkeiten durch REDUX wurde gewählt, um CAPlan als Planungsassistenzsystem einsetzen zu können, das flexibel auf Änderungen und Vorschläge des Benutzers reagieren kann. Zusätzlich wurde der Grundalgorithmus um einige Konzepte erweitert, die die Domänenmodellierung vereinfachen oder die Effektivität erhöhen.

In Abschnitt 2.1 und 2.2 wird zunächst das CAPlan System vorgestellt, soweit dies für die nachfolgenden Kapitel benötigt wird. Abschnitt 2.3 ist eine Einführung in die Begriffswelt partiell ordnender Planer. Dabei wurde besonderer Wert darauf gelegt, die feinen Begriffsnuancen herauszuarbeiten und an Beispielen zu erläutern, weil dieser Wortschatz Grundlage für die Diskussion der Kontrollverfahren sein wird.

2.1 SNLP

In diesem Abschnitt wird der SNLP-Algorithmus vorgestellt. Wir beginnen zunächst mit der historischen Entwicklung, um die Zielsetzung dieses Algorithmus zu verstehen. Das ist im Kontext dieser Arbeit wichtig, weil vermieden werden sollte, durch Kontrolle die wesentlichen Errungenschaften dieses Algorithmus wieder preiszugeben.

2.1.1 Historische Entwicklung und Zielsetzung

Die ersten Planer [NSS60, Sus75, Tat74, War74] verwendeten totalgeordnete Pläne, weil diese Planrepräsentation einfach zu realisieren und zu handhaben ist. Totalordnende Planverfahren haben allerdings den Nachteil, daß sie sich frühzeitig auf die Reihenfolge der Teilziele festlegen müssen. Korrekturen an dieser Reihenfolge können nur durchgeführt werden, indem die gesamte Planungsarbeit zurückgenommen wird bis zu dem Punkt, an dem die Entscheidung für eine inkorrekte Bearbeitungreihenfolge getroffen worden ist.

Sehr früh entstanden deshalb schon Systeme [Sac73, Sac75] mit einer nur partiell geordneten Planrepräsentation[1], was grundsätzlich erlaubt, sich erst dann auf eine spezifische Ausführungsreihenfolge festzulegen, wenn unmittelbarer Handlungsbedarf besteht und sich die Entscheidung auf einer breiteren Informationsbasis treffen läßt. Bei diesen Planern ist also die Planungsreihenfolge von der Ausführungsreihenfolge getrennt. Diese frühen Systeme hatten allerdings keine theoretische Grundla-

[1] In der Literatur wird diese Eigenschaft oft auch als 'nichtlineares Planen' bezeichnet. Eine Abgrenzung der Begriffe findet sich in Abschnitt 2.2.1.

ge, die Aussagen über die Erfülltheit von Zielen zuließ und müssen eher als heuristische, unsystematische Verfahren verstanden werden.

Chapman [Cha87] führte mit dem Modal Truth Criterion (MTC) ein Paradigma ein, daß Aussagen über die Erfülltheit von Zielen erlaubte. Dies löste einen Forschungsboom im Bereich partiellordnender Planverfahren aus. Allerdings stellte sich bald als Hauptnachteil MTC-basierten Planens heraus, daß keinerlei Buchführung über den Lösungszustand schon bearbeiteter Ziele gemacht wird. Im Fall von negativen Interaktionen produziert dieser Planungsansatz daher meist stark redundante Pläne [Kam93a] und gelangt sehr leicht in Endlosschleifen.

McAllester und Rosenblitt [MR91] schließlich stellten mit FIND-COMPLETION einen neuen partiellordnenden Planer vor, der durch das Konzept der Causal Links buchführt über den Fortschritt des Planungsvorganges. Diese Buchführung bringt darüberhinaus zwei angenehme Seiteneffekte mit sich: der Algorithmus sucht systematisch und kommt ohne die kostspielige Berechnung von Wahrheitswerten aus. Bekannter geworden als das Original ist allerdings SNLP [BW92], die geliftete Version dieses Algorithmus.[2]

Die wesentlichen Errungenschaften von SNLP waren somit die partiell ordnende Planrepräsentation, die die Planungsreihenfolge von der Ausführungsreihenfolge trennt und die Vermeidung von Redundanz im Suchvorgang (Systematik) durch Buchführung über bereits geleistete Arbeit.

2.1.2 Der Algorithmus

Der Algorithmus soll hier nur in einer sehr abstrahierten Form dargestellt werden, um den zugrundeliegenden Kontrollfluß klar zu machen. Im nachfolgenden Abschnitt werden dann die Basiskonzepte von SNLP eingeführt und die einzelnen Schritte des Algorithmus etwas differenzierter vorgestellt.

```
GOALS ← { Set of Goals }.
while GOALS ≠ ∅ do:
        chooseAny g ∈ GOALS.
        OPS ← { Set of possible Operators }.
        if OPS = ∅, backtrack.
        else chooseAny o ∈ OPS.
        GOALS = GOALS ∪ { preconditions of o }.
        Make plan consistent. If impossible, backtrack.
```

Abbildung 2.1: Der Grundalgorithmus

Die elegante und einfache Struktur dieses Algorithmus ist häufig gelobt worden, täuscht aber auch über einige Interpretationsschwierigkeiten hinweg, die in Kapitel 2.3 noch ausführlich diskutiert werden.

2.1.3 Die Konzepte von SNLP

Wir werden nun zunächst die Konzepte von SNLP eingeführen und dann die einzelnen Schritte des Algorithmus nochmals etwas genauer untersuchen.

Schritt: In SNLP werden Operatorknoten als Schritte (Steps) bezeichnet.

Causal Link: Das zentrale Konzept von SNLP ist der Causal Link $P \xrightarrow{g} U$, [3] der die Entscheidung des Planers dokumentiert, für den Nutzer U das Ziel g durch den Erzeuger P einzuführen. Diese Entscheidung wird vom Algorithmus sowohl gegen die Zerstörung durch andere Schritte geschützt

[2] Der Begriff des Lifting stammt von [Rob65] und bedeutet in diesem Zusammenhang, daß der Algorithmus auch Operatorschemata verarbeiten kann, d.h. auch Ausdrücke mit Variablen.

[3] Die Bezeichnungen stehen for Producer,goal, User.

als auch dagegen, daß eventuell ein anderer Schritt ebenfalls g für U erzeugen könnte. Diese zwei 'Bedrohungen' der Planungsentscheidung $P \xrightarrow{g} U$ werden durch das Konzept des Threat beschrieben.

Threat: SNLP unterscheidet zwischen negativen und positiven Threats, die formell wie folgt definiert sind:

Definition 1 (positiver (negativer) Threat)
Ein **positiver (negativer) Threat** ist ein Paar $(C, P \xrightarrow{g} U)$ für das gilt:[4]
1. C,P,U sind Schritte eines Planes für die gilt, daß Schritt C nicht notwendigerweise vor P oder nach U angeordnet ist.
2. $\exists q \in effects(C)$ so daß $q \approx g(\neg g)$.

Das Zeichen $\approx$ wurde von Chapman [Cha87] im Rahmen seines MTC in der Bedeutung 'unifizierbar mit' (codesignation) verwendet und hat sich seither in der Literatur etabliert.

Plan: In SNLP ist ein Plan eine Menge von Schritten, auf denen durch eine Menge von Causal Links eine partielle Ordnung definiert ist. Ziel von SNLP ist es, partielle, d.h. unvollständige Pläne durch Einfügen von zusätzlichen Schritten und Causal Links zu einem Plan zu erweitern, der eine Lösung zum gegebenen Problem darstellt. Dabei stellt ein Plan eine Lösung des Problems dar, wenn jede Linearisierung des Plans den Anfangszustand des Problems in einen Zustand transformiert, in dem die Zielbeschreibung gilt.

Der Algorithmus startet mit einem partiellen Plan, der nur zwei künstliche Schritte START und FINISH enthält, die den Anfangszustand bzw. die Zielsituation des gegebenen Problems repräsentieren.

Bindungs-Constraints: SNLP verwaltet auf der Menge der Variablen eine Menge von Bindungsconstraints, um nicht nur bei den Ordnungsrelationen, sondern auch bei der Entscheidung für Variablenbindungen einen least commitment Ansatz verfolgen zu können.

Ordnungs-Constraints: SNLP verwendet außerdem eine Menge von Ordnungs-Constraints. Diese werden einerseits durch die Causal Links eingeführt (weil der Erzeuger P vor dem Benutzer U angeordnet sein muß) und andererseits durch die Threat-Auflösungsstrategien.

Operatorauswahl: Den Begriff der Operatorauswahl muß bei SNLP etwas weiter gefaßt werden: Als Operator werden nicht nur die Operatoren der Domäne verstanden, sondern auch das Ausnutzen bereits vorhandener Operatoren. In Anlehnung an [Wäs93] werden letztere im folgenden als **Phantomisierungsoperatoren** bezeichnet. Die Menge der möglichen Operatoren umfaßt also außer der Menge der anwendbaren Domänenoperatoren auch die Menge der konsistenten Phantomisierungsoperatoren. Unter konsistenten Phantomisierungsoperatoren kann man sich die Menge aller Seiteneffekte bisheriger Planschritte und der initialen Situation vorstellen, die weder Ordnungs- noch Bindungsconstraints verletzen. Implizit stellen Phantomisierungsoperatoren auch den Basisfall des Suchalgorithmus' dar: sie sind die einzigen Operatoren, die keine neuen Ziele einführen.

Threat-Auflösung: Obwohl der Begriff des Threat relativ neu ist [MR91], ist das dahinterliegende Konzept so alt wie das Planen selbst. Schon seit den allerersten Planungssytemen ist eine Reihe von Threatauflösungsmechanismen bekannt, die sich allerdings alle in eine der folgenden vier Klassen einordnen lassen. Die Terminologie stammt von Chapman [Cha87].

1. Demotion: C $\prec$ P
   Demotion löst einen Threat auf, indem es den bedrohenden Schritt vor den Erzeuger P des Causal Links anordnet.

2. Promotion: U $\prec$ C
   Promotion löst einen Threat auf, indem es den bedrohenden Schritt hinter dem Benutzer U des Causal Links anordnet.

3. Separation: P $\prec$ C $\prec$ U und q $\not\approx$ g
   Es gibt Bedrohungen, die nur bei bestimmten Variablenbindungen akut werden. Solche potentiellen Bedrohungen können vermieden werden, indem man Constraints einführt, die diese Variablenbindungen verhindern.

[4] C steht für Clobberer.

4. White Knights
   Zerstört ein Schritt als Nebeneffekt ein schon erreichtes Ziel, so läßt sich das Problem auch dadurch beheben, daß man das zerstörte Ziel noch einmal erzeugt.

Mit dem systematischen Ansatz sind allerdings nur Demotion, Promotion und Separation verträglich, weil durch das White Knight Kriterium für ein zerstörtes Ziel ein zweiter Erzeuger eingeführt wird. Barrett und Weld [BW92] weisen darauf hin, daß zur Erhaltung der Systematik die verschiedenen Möglichkeiten, eine Separation durchzuführen, disjunkt sein müssen.

Beispiel:

Seien $g = (pred\ v_1\ v_2); q = (pred\ w_1\ w_2);$
Ferner gelte $v_1, v_2, w_1, w_2 \in (Const \cup Var)$
Möglichkeiten der Separation sind:

1. $v_1 \not\approx w_1$ und $v_1 \approx w_1$
2. $v_1 \approx w_1$ und $v_2 \not\approx w_2$
3. $v_1 \not\approx w_1$ und $v_2 \not\approx w_2$

Dieses Verfahren läßt sich ganz analog auf n-äre Prädikate übertragen; allgemein gibt es $2^n - 1$ Möglichkeiten, eine Separation auszuführen.

Aus Gründen der Systematik[5] müßte die Threatauflösung eigentlich untrennbar mit einem Planungsschritt verbunden sein (quasi als atomarer Schritt), damit jeder Planungsschritt einen konsistenten Teilplan in einen ebenfalls konsistenten Teilplan überführt. Da aber die Threatauflösungsstrategien Backtrackingpunkte darstellen, werden sie für gewöhnlich wie spezielle Operatoren behandelt, die zur Erfüllung spezieller Ziele (nämlich dem Schutz vor einem Threat) dienen. In CAPlan wird diese Ziel/Operator-Sicht der Threatauflösungs explizit gemacht in Form von sog. **Protektionszielen** bzw. **Protektionsoperatoren**. Die Threatauflösung wird also in konkreten Implementierungen von SNLP durch separate Planschritte realisiert. Die Auswirkungen dieser Vorgehensweise waren Gegenstand intensiver Forschung (vgl. 30).

2.2 CAPlan-spezifische Erweiterungen

Die für CAPlan spezifischen Erweiterungen von SNLP lassen sich in drei Gruppen einteilen:

1. Verwendung eines JTMS zur Verwaltung von Entscheidungsabhängigkeiten
2. Definition eines Zielrekursionskriteriums
3. Erweiterungen der Modellierungssprache, die unter anderem implizite Kontrollpräferenzen ausdrücken
4. Einsatz mächtigerer Backtrackingverfahren

Wir werden diese Erweiterungen im folgenden nur in soweit vorstellen, wie sie Auswirkungen auf die Kontrolle haben.

2.2.1 Verwaltung von Entscheidungsabhängigkeiten

In CAPlan werden alle Entscheidungen über das JTMS-System REDUX verwaltet. Die explizite Verwaltung der Abhängigkeiten ermöglicht ein robustes Verhalten des Planers: der Benutzer kann an beliebigen Stellen des Planes Entscheidungen revidieren, ohne daß davon unabhängige Teile des Planes ebenfalls zurückgezogen werden.

[5] Der Begriff der Systematik wird in Abschnitt 2.2.1 ausführlich diskutiert. Hier genügt die Vorstellung, daß kein Planungszustand mehrfach durchlaufen wird.

2.2.2 Definition eines Zielrekursionskriteriums

McAllester und Rosenblitt beschreiben ihren Algorithmus als beschränkte Tiefensuche. Dies hat jedoch den unerwünschten Nebeneffekt, daß der Algorithmus dann nur noch relativ zu dieser Tiefenschranke vollständig ist. Auch Barrett und Weld ([BW92], S. 43) und Kambhampati [KK94b] scheinen Tiefenschranken zu verwenden.[6] Analog zu total-ordnenden Planern wie PRODIGY [FV92, CtPRG92] hat Frank Weberskirch in [Web94] ein Zielrekursionskriterium (goal loop criterium) definiert, daß Schleifen in der Ziel-Teilziel Hierarchie erkennt und eine Tiefenschranke überflüssig macht.

Das Zielrekursionskriterium. Das Zielrekursionskriterium für totalordnende Planer besteht lediglich darin, zwei Ziele, die in einer Ziel-Teilziel Relation stehen, auf Gleichheit zu überprüfen.
Weil CAPlan aber als partiellordnender Planer kein Backtracking über die Zielauswahl macht, wird ein komplexerer Test benötigt. Er läßt sich wie folgt definieren:

Definition 2 (Zielrekursionskriterium)
Ein Zielknoten g muß nicht weiter expandiert werden, wenn
1. g einen seiner Vorfahren (bezüglich der Ziel-Teilziel-Hierarchie) vollständig matcht und
2. zur Erreichung von g derselbe Operator verwendet wird, der auch zur Erreichung des Vorfahren benutzt wurde.

Das Zielrekursionskriterium für partiellordnende Planer ist also wesentlich schwächer als dasjenige für totalordnende Planer.

Auswirkungen auf die Eigenschaften des Planalgorithmus. Sinn eines Zielrekursionskriteriums ist es, diejenigen Suchbaumzweige zu identifizieren und zu verwerfen, die –wenn überhaupt– nur solche Lösungen erzeugen, die redundante Schritte enthalten.[7] In [KKY94] wird dieser Plantyp als 'bloated plan' bezeichnet. Im selben Papier werden auch **minimale Pläne** definiert als solche Pläne, aus denen keine Schritte entfernt werden können, ohne die Korrektheit des Plans zu zerstören. Wichtig ist, daß in Algorithmen ohne Tiefenschranke ein Zielrekursionskriterium notwending ist, um die Termination der Suche garantieren zu können, weil damit verhindert wird, daß beliebig tiefe Suchzweige expandiert werden. Allerdings verliert der Planer damit auch die Fähigkeit, theoretisch *alle* Lösungen eines Problems finden zu können (Vollständigkeit) und kann dies nur noch für die Untermenge der minimalen Pläne garantieren. Um Mißverständnissen vorzubeugen: die Verwendung von Zielabbruchkriterien impliziert nicht, daß jeder vom Algorithmus gefundene Plan minimal ist, sondern nur, daß jeder minimale Plan durch geeignete Wahl von Operatoren erreicht werden kann .

2.2.3 Erweiterungen der Domänenmodellierungssprache

CAPlan bietet eine erweiterte Domänenmodellierungssprache, die die Modellierbarkeit bestimmter Phänomene vereinfacht. Die dadurch formulierbaren Zusatzinformationen lassen sich für eine gezieltere Kontrolle nutzen.

Unterscheidung der Vorbedingungen in Preconditions und Phantoms. In CAPlan können Vorbedingungen als 'Phantom' deklariert werden.[8] Dies ist ein spezieller Typ von Vorbedingung, der nicht durch Anwendung von Domänenoperatoren erfüllt werden darf, sondern nur durch Ausnützen von Seiteneffekten. Mit diesen Phantom-Vorbedingungen lassen sich Zulässigkeitsbedingungen (engl. admissability constraints) eines Operators formulieren.
Sie sind allerdings nur bedingt für solch eine Modellierung geeignet, weil bei der Entscheidung für einen Operator, für den eine Phantom-Vorbedingung im Moment nicht erfüllt werden kann, diese Entscheidung im Zuge des Backtracking erst dann zurückgezogen werden kann, wenn im gesamten

[6] In einem Papier, das erst im Sep. 95 erscheint [Kam95], widmet sich Kambhampati allerdings doch dieser Problematik. Er stellt wie wir fest, daß einfache Kriterien für CL-Planer nicht zulässig sind und stellt ein eigenes, teuer auszuwertendes Kriterium vor, daß im wesentlichen nur bei einer FIFO-Strategie der Zielbearbeitung einsetzbar ist.

[7] Redundant sind gerade die Schritte, die die zwei rekurrierenden Ziele verbinden.

[8] Man beachte, daß Phantomisierungen (= Ausnutzen von Seiteneffekten) und die hier vorgestellten Phantomziele/Phantomvorbedingungen separate Konzepte sind!

Suchraum zur Erfüllung der noch offenen Ziele keine Möglichkeit zur Phantomisierung dieser Phantom-Vorbedingung gefunden werden kann. Phantom-Vorbedingungen sollten bei der Domänendefinition folglich nur mit äußerster Vorsicht eingesetzt werden.

Unterscheidung der Effekte in Haupt- und Seiteneffekte. CAPlan erlaubt in Anlehnung an [Wil84] die Unterscheidung der Effekte eines Operators in Purposes (Haupteffekte) und Side Effects (Seiteneffekte). Diese Information wird von der Defaultkontrolle SNLP+ als Präferenzordnung auf der Menge der Operatoren interpretiert: zur Erfüllung eines Zieles werden zunächst diejenigen Operatoren ausgewählt, die dieses Ziel als Haupteffekt aufweisen.

Invarianten. Es ist möglich, bei der Definition eines Problems Literale als Invarianten zu kennzeichnen, also als Eigenschaften des Weltzustandes, die nicht durch Anwendung von Domänenoperatoren verändert werden können und daher während des gesamten Planungsvorganges unverändert bleiben.

Typconstraints. CAPlan untersützt ein hierarchisches Typkonzept, daß über die Definition von Typconstraints für die Argumente von Domänenoperatoren die Angabe typbezogener Zulässigkeitsbedingungen ermöglicht.

2.3 Begriffswelt partiell ordnender Planungsalgorithmen

In diesem Abschnitt wird schrittweise in die Begriffswelt partiellordnender Planer eingeführt, werden Begriffe gegeneinander abgegrenzt und häufige Mißverständnisse ausgeräumt. Auf dieser begrifflichen Grundlage wird dann die Analyse des Kontrollproblems und der verschiedenen Kontrollverfahren aufbauen.

2.3.1 Lineares und nichtlineares Planen

In [Vel92] werden diese Begriffe wie folgt definiert:

Linear sind Planer, die die Menge der offenen Ziele als Stack verwalten.

Nichtlinear sind Planer, die die Menge der offenen Ziele als Menge verwalten.

Nichtlineare Planer sind im Gegensatz zu linearen Planern in der Lage, interagierende Probleme zu lösen, indem sie die Lösungen miteinander verzahnen. Eine Reihe von Forschern hat jedoch dringend empfohlen, die Begriffe linear/nichtlinear nicht mehr zu verwenden, weil sie vor allem in der älteren Literatur in der Bedeutung totalordnend/partiellordnend gebraucht worden sind.

2.3.2 Totalordnendes und partiellordnendes Planen

Diese beiden Begriffe beziehen sich auf die Repräsentation der Pläne während des Planungsvorgangs: Bei totalordnenden Planern sind die einzelnen Planschritte total geordnet, bei partiellordnenden Planverfahren sind sie eventuell nur partiell geordnet. Es gilt:

1. Eine Totalordnung ist der Spezialfall einer Partialordnung, bei der für jedes Schrittpaar eine Ordnungsrelation angegeben ist.

2. Um aus einem partiell geordneten Plan eine ausführbare Sequenz zu machen, muß er linearisiert, also total geordnet werden. Umgekehrt kann ein totalgeordneter Plan durch ein einfaches Postprocessing in einen partiell geordneten Plan umgewandelt werden [Vel92][Bäc94].

3. Partiellgeordnete Pläne werden gerne als Stellvertreter für die Menge ihrer Linearisierungen interpretiert [Wel94, KKY94, MR91]. Abb. 3 zeigt, wie man sich solch eine Zuordnung vorstellen kann.

4. Veloso [Vel89] hat darauf hingewiesen, daß häufig angenommen wird, ein partiell geordneter Plan repräsentiere die Menge aller Lösungen eines Problems. Diese Annahme ist falsch, da ein partiell geordneter Plan zwar alle seine Linearisierungen repräsentiert, aber nicht in der Lage ist, Operatordisjunktionen auszudrücken.

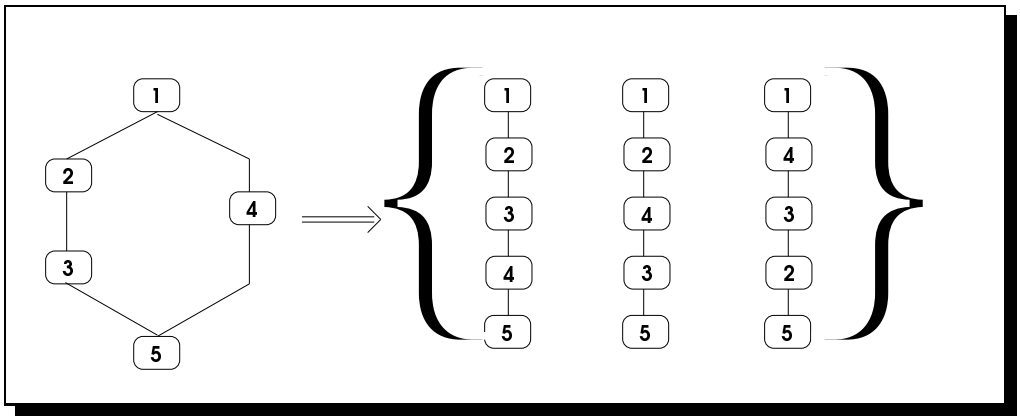

Abbildung 2.2: Abbildung eines partiell geordneten Plans auf die Menge seiner Linearisierungen

5. Linearität und Planrepräsentation sind völlig unabhängige Konzepte:
   Totalordnende Planer sind zwar häufig linear und partiell ordnende meist nichtlinear, doch sind auch einige Ausnahmen bekannt: Beispiele für nichtlineare, totalordnende Planer sind NoLimit [Vel89] und PRODIGY V4.0 [CtPRG92]. Ein Vertreter der linearen, partiellordnenden Planer hingegen ist der Vorläufer von CAPlan [Wäs93], der auf SIPE [Wil84] aufbaut.

2.3.3 Suche im Zustandsraum und Suche im Raum partieller Pläne

Bei der Suche im Zustandsraum stellt man sich den Suchraum als einen Graphen vor, indem die Knoten (Domänen-)zustände repräsentieren und die Kanten Operatoranwendungen darstellen. Suche im Raum partieller Pläne basiert hingegen auf der Vorstellung, daß ein Knoten einen partiellen[9] Plan symbolisiert, der selbst die Menge aller seiner Erweiterungen repräsentiert. Eine Kante in diesem Suchgraph repäsentiert eine Entscheidung, mit der die Menge der möglichen Erweiterungen eingeschränkt wird. In [KKY94] wird diese Sichtweise unter der Bezeichnung 'Refinement Search' ganz ausführlich beschrieben und formal definiert. Da Zustände durch Operatorfolgen ineinander überführt werden können, lassen sie sich auch als Anwendung einer Operatorfolge auf den Anfangszustand definieren. Über den Umweg der Linearisierungen lassen sich daher die zwei Suchräume aufeinander abbilden [MBD94]. Abb. 2.3 zeigt eine solche Zuordnung.

Wichtig ist wiederum eine Abgrenzung gegenüber den vorangegangenen Begriffen:

1. Bei linearen Planern ist die Menge der möglichen Erweiterungen jeweils kleiner als bei den nichtlinearen. Im Falle von linearen, totalordnenden Planern besteht eine eineindeutige Abbildung der beiden Suchräume aufeinander.

2. Normalerweise planen partiellordnende Planer im Suchraum partieller Pläne. Es gibt aber durchaus Beispiele partiellordnender Planer, die im Zustandsraum planen [GK91] und Beispiele totalordnender Planer, die im Raum partieller Pläne arbeiten [Wal75, Tat74, War74].

2.3.4 Planungsreihenfolge und Ausführungsreihenfolge

Es wird immer wieder betont, daß einer der großen Vorteile partiellordnenden Planens die Trennung von Planungsreihenfolge und Ausführungsreihenfolge ist. Das Beispiel PRODIGY [CtPRG92] zeigt aber, daß auch totalordnende Planer Planungsreihenfolge und Ausführungsreihenfolge trennen können. Die Trennung wird dadurch realisiert, daß jeder Operator durch zwei unabhängige Knoten repräsentiert wird: ein Knoten symbolisiert die Planungsentscheidung, der andere die Ausführung des Operators.

[9] Man beachte den Unterschied zwischen partiell und partiellordnend!

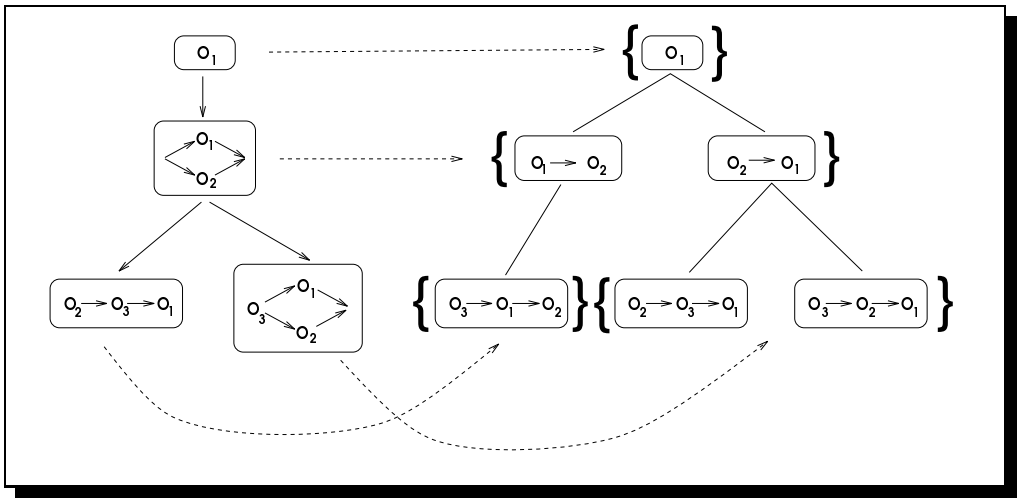

Abbildung 2.3: Abbildung des Suchraumes eines partiell ordnenden Planers (links) auf den Suchraum eines total ordnendes Planers

2.3.5 Vollständigkeit

Der Begriff der Vollständigkeit wird in der Literatur uneinheitlich verwendet. Veloso [Vel89] unterscheidet drei Vollständigkeitsbegriffe. Unter Vollständigkeit kann demnach die Fähigkeit eines Planers verstanden werden,

- alle Lösungen (insbesondere auch unendliche),
- alle minimalen Lösungen,
- oder zumindest eine Lösung

finden zu können.

In [KKY94] wird die erste Definition verwandt, die meisten anderen Quellen scheinen aber den Begriff der Vollständigkeit eher in der dritten Bedeutung zu verwenden. Diese letzte Version ist auch im Bereich der Reduktionssysteme gebräuchlich. Die meisten Planer garantieren jedoch Vollständigkeit bezüglich minimaler Pläne. Solche Suchverfahren heißen zulässig (engl. admissible) und dürfen diejenigen Pfade des Suchraums unerforscht lassen, die nachweißlich zu keiner Lösung führen oder nur zu nicht-minimalen Lösungen.

2.3.6 Systematik

McAllester, der diesen Begriff geprägt hat, definiert ihn als die Eigenschaft eines Planalgorithmus, keinen Plan (oder partiellen Plan) mehrfach zu untersuchen. Daraus folgt unmittelbar, daß der Suchraum kleiner ist als bei unsystematischen Verfahren, bei denen u.U. mehrere Wege zu ein und demselben Knoten führen. Diese ursprüngliche Definition ist jedoch von verschiedenen Forschern höchst unterschiedlich ausgelegt worden. Daher hat Kambhampati in [Kam93a] die verschiedenen Systematikbegriffe gegeneinander abgegrenzt:

1. (strenge) Systematik
   Je zwei vollständige oder partielle Pläne in verschiedenen Suchzweigen haben völlig disjunkte Mengen von durch sie repräsentierten Operatorfolgen. SNLP in seiner Grundform garantiert strenge Systematik.
2. Systematik bezüglich der Kausalstruktur
   Diese abgeschwächte Form bezieht den Begriff der Systematik nicht auf Operatorfolgen, sondern auf die Kausalstruktur des Plans.

3. Lösungssystematik
   Bei dieser Variante kann es während der Planung partielle Pläne geben, deren zugehörige Mengen von Operatorfolgen nicht disjunkt sind. Für alle *vollständigen Pläne* gilt allerdings, daß die Mengen der durch sie repräsentierten Operatorfolgen disjunkt sind. Diese Variante tritt auf, wenn bei SNLP die Auflösung von Threats verzögert wird.

2.3.7 Commitment

SNLP ist entsprechend der üblichen Terminologie ein Least Commitment Planer. Dieser Begriff ist aber sehr irreführend, weil SNLP mit zu der Gruppe von Planern gehört, die sich *am meisten* festlegen. Der Begriff des commitment bezog sich in der älteren Literatur hauptsächlich auf die Festlegung der Zielreihenfolge. Wie Abbildung 2.4 zeigt, legen sich partiellordende Planer in dieser Hinsicht tatsächlich weniger fest als totalordnende Planer. In der neueren Literatur wird dieser Begriff jedoch

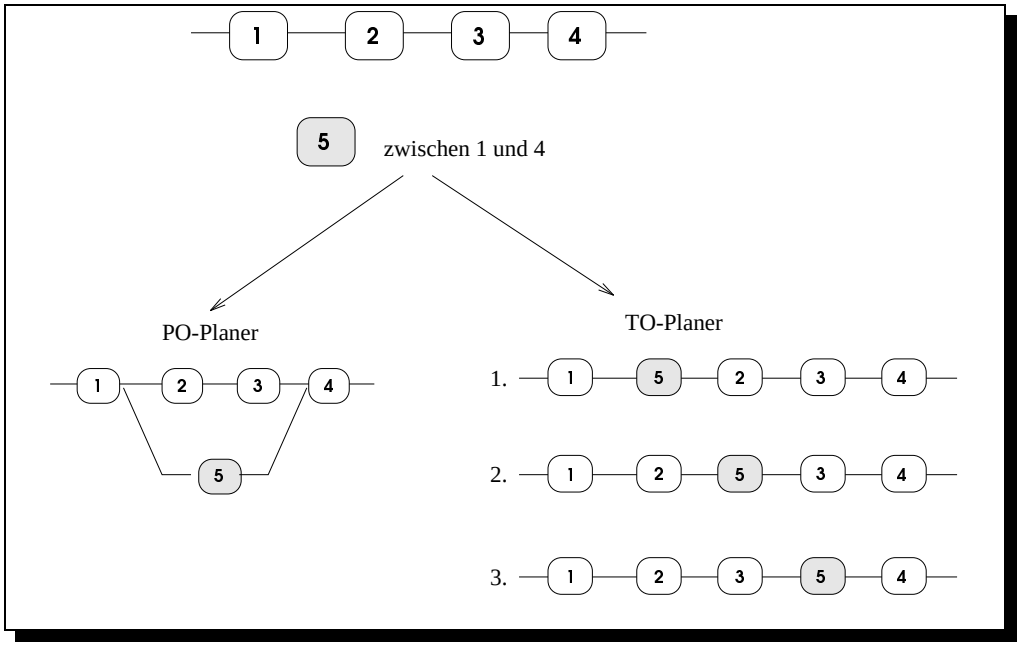

Abbildung 2.4: Unterschiedlicher Festlegungsgrad der Zielreihenfolge

für alle Arten von Entscheidungen verwendet. Die folgende Tabelle gibt Aufschluß darüber, welche commitment-Typen existieren und wie sie von unterschiedlichen Planertypen gehandhabt werden.

Festlegung von...	SNLP	TO-Planer
Zielreihenfolge	–	X
Schrittreihenfolge	partiell	total
Auflösung von Konflikten	positive u. negative	negative
Domänenoperator	X	Auswahl, aber keine Festlegung
Phantom.-Operator	X	–

- SNLP legt sich nicht auf eine Zielreihenfolge fest; diese ergibt sich vielmehr aus der Schrittreihenfolge. Die Schrittreihenfolge wird zum einen durch die Ziel-Teilziel Hierarchie und zum anderen durch die Auflösung von Konflikten bestimmt.

- Wichtig ist, daß sich SNLP durch die Causal Links ganz starr auf einen Erzeuger festlegt; diese Eigenschaft folgt direkt aus dem Konzept des Causal Link und ist unabhängig davon, ob positive Threats aufgelöst werden oder nicht.

Zustandsbasierte Planer legen sich hingegen nicht auf einen speziellen Erzeuger fest, sondern stellen nur sicher, daß zum Ausführungszeitpunkt des einführenden Operators alle benötigten Vorbedingungen durch den Zustand erfüllt werden.
Die striktere Festlegung auf nur einen Erzeuger wird SNLP dann zum Verhängnis, wenn Ziele häufig zerstört und erzeugt werden (sog. high frequency goals). In solchen Domänen steigt nämlich die Wahrscheinlichkeit, daß die erste Entscheidung revidiert werden muß. SNLP ist dann auf Backtracking angewiesen, während dies für totalordnende Planer meist nicht nötig ist, da sie nicht verlangen, daß der zunächst ausgewählte Erzeuger im entstehenden Plan auch tatsächlich derjenige Schritt ist, der das Ziel erzeugt. Veloso und Blythe [VB94] haben dieses Phänomen 'Linkability Problem' genannt und als erste beschrieben.[10]

- Der Begriff des 'Least Commitment' ist auch insofern mißverständlich, als er suggeriert, daß Entscheidungen erst dann getroffen werden, wenn genügend Informationen vorhanden sind, um sich für die *richtige* Alternative zu entscheiden. Dies trifft nicht zu; richtig ist zwar, daß SNLP unspezifischere Entscheidungen trifft als TO Planer, diese werden jedoch genauso blind getroffen.

- Bemerkenswert ist auch, daß einige Ordnungsbeziehungen redundant repräsentiert sind: Besteht zwischen zwei Schritten ein Threat, der durch Anordnung dieser zwei Schritte aufgelöst worden ist, und wird danach ein Phantomisierungslink zwischen diesen zwei Schritten eingeführt, so kann die Anordnung der Schritte erst dann rückgängig gemacht werden, wenn *beide* Entscheidungen zurückgenommen werden. Hätte man aber zuerst den Causal Link eingeführt, dann wäre der Threat ungültig geworden und kein weiterer Ordnungconstraint eingeführt worden. In Domänen, in denen fertige Pläne immer totalgeordnet sind (z.B. die Blockworlddomäne von Nilsson [Nil80]) kann man korrekte Pläne erzeugen, ohne ein einziges Protektionsziel zu bearbeiten.

- Kambhampati hat herausgefunden, daß ein Trade-Off zwischen der Redundanzvermeidung und dem Commitment besteht: Planer, die durch Systematik Redundanz im Suchraum vermeiden, müssen sich stärker festlegen; solche Planer, die einen echten least-commitment Ansatz verfolgen, nehmen Redundanz in Kauf. Zuletzt noch ein Diagram aus [Kam93a] (vereinfacht), daß

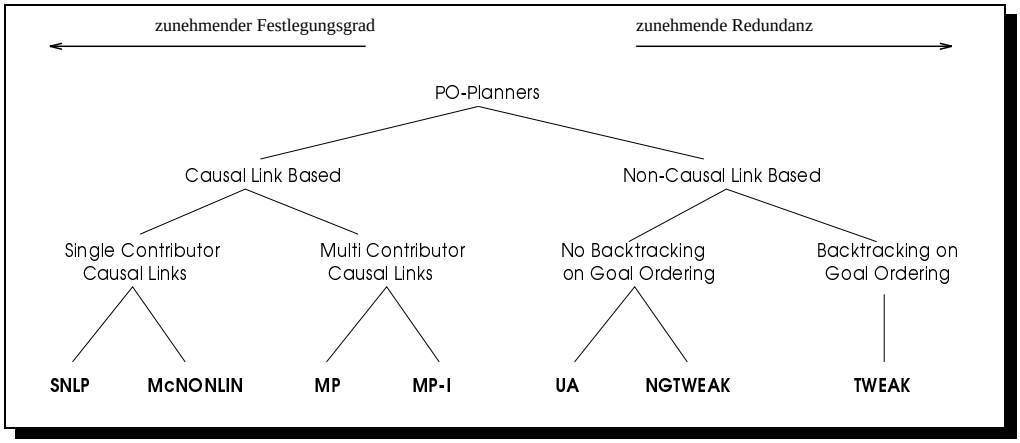

Abbildung 2.5: Die Bandbreite versch. Planungsalgorithmen bezüglich Commitment und Redundanzvermeidung

verschiedene Planer bezüglich dieser beiden Kriterien anordnet. Wie ersichtlich, ist SNLP der sich am stärksten festlegende partiellordnende Planer. Empirischen Untersuchungen Kambhampatis zufolge sind Planer in der Mitte des Spektrums am effizientesten.

[10] Veloso und Blythe beziehen diesen Begriff allerdings nur auf die Festlegung von Phantomisierungsoperatoren.

Kapitel 3

Analyse des Kontrollproblems für Causal Link Planer

In diesem Kapitel werden wir nun versuchen, die Struktur des Kontrollproblems genauer zu analysieren und zu einer schärfer umgrenzten Aufgabenstellung zu gelangen. Dazu werden wir zunächst die Schwierigkeit des Kontrollproblems darlegen. Daran anschließend werden wir die Anforderungen an eine Kontrollkomponente genau spezifizieren. Dies beinhaltet einerseits die Festlegung nicht zu verletzender Charakteristika des Planers (die sog. negativen Anforderungen), und andererseits die Formulierung zu erreichender Ziele und Qualitätsanforderungen. Schließlich werden wir den Raum denkbarer Kontrollmöglichkeiten ausloten und aufbauend auf den zuvor formulierten Anforderungen den methodischen Spielraum eingrenzen.

3.1 Das Kontrollproblem

In diesem Abschnitt wollen wir uns kurz vergegenwärtigen, wie schwierig eigentlich das allgemeine Planungsproblem ist. In einem zweiten Abschnitt sollen außerdem besondere Schwächen von SNLP und spezifische Schwierigkeiten des Kontrollproblems für CAPlan herausgestellt werden.

3.1.1 Was macht Planung schwer?

Betrachtet man Beispieldomänen wie die Blocksworld oder die Transportation-Domäne, die für Menschen keinerlei Schwierigkeiten aufweisen, solange nicht nach optimalen Lösungen gesucht wird, so ist zunächst einmal unverständlich und immer wieder überraschend, daß Planungssysteme so große Schwierigkeiten haben, selbst moderate Probleme zu lösen. Die Erfahrung, daß maschinelle Planung 'sehr schwierig' ist, wurde jedoch 1992 von Erol, Nau und Subramanian [ENS92] bestätigt, indem sie bewiesen haben, daß STRIPS-Planung mit einer festgelegten Menge von Datalog-Operatoren in PSPACE liegt[1], also in einer noch schwierigeren Komplexitätsklasse als NP.[2]

Einen weiteren Hinweis finden wir bei Drummond und Currie [DC89], die vorgerechnet haben, daß der Suchraumes eines Blocksworld-Problems mit 4 Zielen, jeweils 3 anwendbaren Operatoren und Tiefenschranke 7 bereits 12^7 partielle Pläne umfaßt. Selbst wenn wir eine realistischere Zahl von 2 Operatoren annehmen, führt dies zu über 2 Millionen Suchknoten. Für eine vollständige Durchsuchung solch eines Suchraumes würde CAPlan mindestens 116 Stunden[3] benötigen.

Wie bei 'intractable' Problemen üblich, bleiben uns bei der Planung dennoch mehrere Möglichkeiten, in praktikabler Zeit zu Lösungen zu gelangen:

[1] Dieses Ergebnis basiert zu einem großen Teil auf einem Beweis von Bylander [Byl91], der aussagenlogische STRIPS-Planung untersucht hat, bei der die Operatoren Teil der Eingabe eines Problemes sind.

[2] Wie üblich wird angenommen, daß PSPACE > NP > P.

[3] Für diese Abschätzung wurde angenommen, daß ein Suchschritt 200ms dauert.

Einschränkung des Lösungsbegriffs. Vor allem bei Optimierungsproblemen wird in der Regel die Optimalität abgeschwächt und man beschränkt sich auf Approximationen einer vordefinierten Güte oder auf Algorithmen, die mit gewisser Wahrscheinlichkeit optimale Lösungen finden. Das Planungsproblem ist allerdings schon in der Grundform so schwierig, daß für gewöhnlich schon von vornherein darauf verzichtet wird, optimale Lösungen zu finden.[4] Bei dieser Aufgabenstellung läßt sich der Lösungsbegriff dahingehend abschwächen, daß man auf eine vollständige Lösung des Problem verzichtet. Dieser Weg wird von Anytime-Algorithmen [Her95, Zil93] eingeschlagen und ist in Realzeit-Anwendungen unumgänglich. Bei den Einsatzdomänen von CAPlan ist eine solche Einschränkung jedoch nicht sinnvoll.

Eine weitere Einschränkungsmöglichkeit ist das Relaxieren von Constraints, daß sich allerdings als sehr schwierig erwiesen hat, und das in CAPlan ebenfalls wenig sinnvoll ist, da bisher nur harte Constraints formulierbar sind. Das bisher von CAPlan ungenutzte REDUX-Konzept des 'Optimality Loss' könnte jedoch zur Beurteilung des Verletzungsgrades weicher Constraints genutzt werden.

Domänenspezifische Anpassung des Algorithmus. Bei komplexitätstheoretischen Betrachtungen ist für gewöhnlich nur die *allgemeine* Problemstellung schwierig, einzelne Probleminstanzen hingegen oft sehr einfach. Dies gilt auch für die Planung, wie Gupta und Nau am Beispiel der Blocksworld sehr schön gezeigt haben: Sie geben in [GN91] einen ganz offensichtlichen Algorithmus an, mit dem jedes Blocksworld-Problem in 2n Schritten gelöst werden kann (n ist die Anzahl der Klötze): Zunächst werden alle Klötzchen auf den Tisch gestellt und dann die gewünschten Türme von unten nach oben aufgebaut.

Wie das obige Beispiel demonstriert, läßt sich also durch geeignete Anpassung des Algorithmus auf eine Domäne ein effizientes, im Einzelfall sogar lineares Planverhalten erreichen. Für uns stellt sich nun die Frage, wie sich solch ein domänenspezifischer Zuschnitt erreichen läßt.

1. Ist ein solcher (Grund-)Algorithmus für eine Domäne schon bekannt, oder auch nur eine 'übliche' Vorgehensweise, so ist die einfachste Möglichkeit, dies dem Planer direkt mitzuteilen (das sogenannte 'learning by being told'). Die Bereitstellung einer solchen Schnittstelle ist wahrscheinlich ein Grund für den Erfolg der Hierarchical Task Network (HTN) Planer [CT91, Wil88] in realistischen Planungsdomänen.
   Ganz allgemein scheint auch der Erfolg menschlichen Planens darin zu liegen, daß im Gegensatz zu automatischen Planern nicht blind gesucht wird, sondern daß in großem Umfang allgemeines Weltwissen und zusätzlich zur Verfügung stehende Information (etwa die Position eine Klötzchens) dazu verwandt wird, den Suchraum sehr stark einzuschränken. Eine Schnittstelle zur Eingabe domänenspezifischer Kontrollstrategien wäre also auch eine Möglichkeit, dem Planer zusätzliche Informationsquellen zur Verfügung zu stellen.

2. Eine weitere Möglichkeit ist, dem Planer Lernverfahren mitzugeben, die den allgemeinen Algorithmus zu einem optimal angepaßten Suchverfahren konvergieren lassen. Eine Arbeit, die besonders diesen Konvergenzaspekt betont, ist [BV94].

3. Eine letzte Möglichkeit der Anpassung an individuelle Domänen ist, in einer Analysephase Metawissen über die Domäne zu extrahieren, und dieses domänenspezifische Wissen für die Steuerung des Planungsprozesses zu verwenden.

Heuristische Suche. Für viele der als intractable bekannten Domänen konnte inzwischen gezeigt werden, daß der worst-case relativ selten ist, und die Durchschnittskomplexität sehr viel niedriger ist. Als Beispiel mag etwa die Arbeit von Goldberg, Purdom und Brown [GPB82] dienen, die empirisch gezeigt haben, daß SAT-Probleme im Durchschnitt in $O(n^2)$ gelöst werden können.

Wir haben ähnliche Erfahrungen bei der Planung gemacht, als wir versucht haben, mit einem Problemgenerator größere Mengen von Testprobleme für die Blocksworld zu erzeugen: 'interessante' Probleme,

[4] Erol, Nau und Subrahmanian [ENS92] haben gezeigt, daß das Finden optimaler Pläne (mindestens) PSPACE ist, also komplexitätstheoretisch nicht schwieriger als das Finden irgendeiner Lösung. Insbesondere bei Lösungsräumen mit hoher Lösungsdichte ist das Finden einer Lösung aber wesentlich schneller als das Finden aller Lösungen.

bei denen Türme in andere Türme transformiert werden sollen, sind sehr schwer zu generieren. Dies wird auch klar, wenn man sich vor Augen hält, daß eine Situation, in der alle Klötzchen zu einem Turm gestapelt sind, eine ganz spezifische Ausnahmesituation im Planraum darstellt, in der als einzig mögliche Aktion das oberste Klötzchen wieder vom Turm genommen werden kann.

Wie kann nun die Einsicht, daß wirklich schwierige Probleme selten sind, zur Lösung unseres Planproblems beitragen? Eine mögliche Folgerung ist, daß wohl die meisten Probleme durch den Einsatz einiger grundlegender Heuristiken lösbar sein müßten. Die häufig gemachte Beschränkung der Problemmengen auf Probleme mit sich gegenseitig behindernden Zielen verzerrt die Bewertung von Kontrollmechanismen zugunsten analytischer Verfahren, die jedoch für den allgemeinen Fall viel zu aufwendig sein dürften. Bevor daher der Einsatz teurer, lernender Verfahren für eine Domäne erwogen wird, sollte sorgfältig geprüft werden, ob die intendierte Problemverteilung wirklich in die kleine Sparte der schwierigen Probleme fällt, und damit, ob der Einsatz einiger Basisheuristiken, evtl. unterstützt durch von Hand eingegebene domänenspezifische Regeln. nicht völlig ausreichend ist.

Diese These wird auch durch neuere Arbeiten im Gebiet der Constraint Satisfaction Probleme gestützt, in der sich Algorithmen mit ganz einfachen, heuristischen Repair-Methoden herkömmlichen Backtrackingverfahren als weit überlegen erwiesen haben. Zusammenfassend läßt sich also sagen, daß eine erfolgsversprechend Strategie darin besteht, eine statischen Basiskontrolle mit lernenden Kontrollverfahren zu kombinieren, die den allgemeinen Algorithmus an die jeweilige Domäne anpassen.

3.1.2 Ein allgemeines Modell der Planungskontrolle

Wie im vorangegangenen Abschnitt schon näher ausgeführt, treten auf dem Gebiet der Constraint Satisfaction Probleme sehr billige, zufallsbasierte Suchverfahren wie etwa GSAT in scharfe Konkurrenz zu intelligenten und aufwendigen. Daher stellt sich auch auf dem Gebiet der maschinellen Planung die Frage nach einem Erwartungsmodell wissensintensiver Kontrolle, wie sie in dieser Arbeit untersucht werden soll.

3.1.2.1 Modell der optimalen Kontrolle

In einem ersten, naiven Ansatz mag man sich unter optimaler Kontrolle einen Suchprozeß vorstellen,bei dem das Suchverfahren direkt zu einer Lösung geleitet wird, ohne jemals Suchschritte zurücknehmen zu müssen. Solch ein Modell wird in Abb. 3.1 veranschaulicht. Wie im letzten Abschnitt

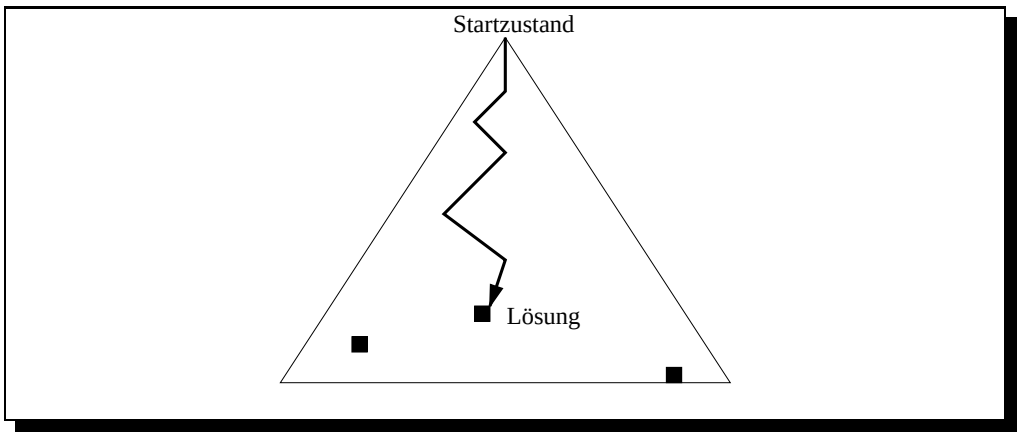

Abbildung 3.1: Veranschaulichung des Modells optimaler Kontrolle: Der Planer wird direkt zu einer Lösung gesteuert.

schon näher erläutert, ist dieses Modell aber nur bedingt von Nutzen, weil es aus komplexitätstheoretischen Gründen für das allgemeine Planungsproblem unerreichbar bleibt. Die Spezialfälle, für die eine optimale Kontrolle realisierbar ist - wie der oben angeführte Algorithmus zur Lösung von Blocksworldproblemen - sind für die Praxis irrelevant, weil durch den gänzlichen Wegfall der Sucharbeit

auch keine Planung im eigentlichen Sinne mehr geleistet wird. Wir benötigen also ein realistischeres, abgeschwächtes Modell.

3.1.2.2 Ein abgeschwächtes Modell optimaler Kontrolle

Wir schwächen das Modell optimaler Kontrolle dahingehend ab, daß wir Sucharbeit nicht mehr gänzlich ausschließen wollen. Um aber dennoch unsere Erwartungen an wissensintensive Kontrolle gegenüber einem blind suchenden, zufallsgesteuerten Suchprozeß abzugrenzen, fordern wir, daß einzelne Fehlentscheidungen nur begrenzte Auswirkungen auf die Effizienz des Suchverfahrens haben bzw. daß der Effizienzverlust mit der Anzahl der Fehlentscheidungen korreliert. Da der Backtrackingaufwand zur Behebung einer Fehlentscheidung mit exponentiellem Aufwand wächst, je früher diese Entscheidung im Suchprozeß getroffen worden ist, läßt sich daraus unmittelbar die Forderung ableiten, daß frühe Entscheidungen möglichst sicher sein sollten und daß wir erst bei zunehmender Suchtiefe Fehlentscheidungen der Kontrolle und damit verbundene Sucharbeit tolerieren wollen. Solch ein abgeschwächtes Modell läßt sich etwa wie folgt darstellen: Dieses Modell deckt sich recht gut mit

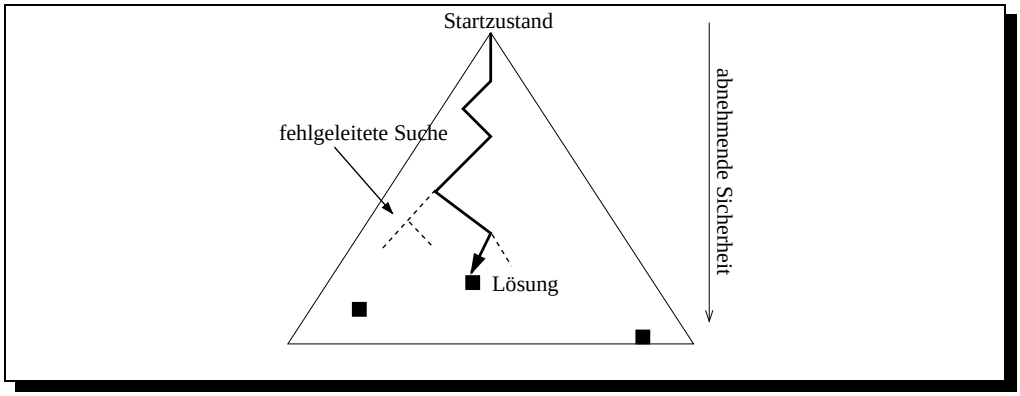

Abbildung 3.2: Ein abgeschwächtes, realistischeres Modell optimaler Kontrolle: Mit zunehmender Suchtiefe sind einige Fehlentscheidungen erlaubt.

der Erwartungshaltung, die wissensintensiver Kontrolle entgegengebracht wird und läßt sich durch konkrete Kontrollverfahren zumindest approximieren, etwa weil Zustandsinformationen, die von sehr vielen Kontrollverfahren ausgenutzt werden, zu Beginn des Planungsproblems noch relativ sicher sind.

3.1.3 Das Kontrollproblem für CAPlan

Nachdem wir uns nun mit dem allgemeinen Planungsproblem auseinandergesetzt haben, wollen wir uns hier kritische Punkte des SNLP - Algorithmus genau anschauen und auch auf einige Faktoren hinweisen, die die Kontrolle von SNLP besonders schwierig machen.

3.1.3.1 Problematik der Zielreihenfolge

Korf hat in [Kor87] Planungsprobleme in unabhängige, serialisierbare und nicht-serialisierbare Probleme eingeteilt, also in solche, bei denen die Lösungen der einzelnen Teilziele in jeder Reihenfolge, nur in einer bestimmten Reihenfolge oder nur miteinander verschränkt eine Lösung des Gesamtproblems bilden. Offensichtlich ist die Klasse der nicht-serialisierbaren Probleme durch einen Überreichtum an lokalen Minima die schwierigste Problemklasse. Als Beispiel dieser Problemklasse führt Korf den Rubic's Cube an, bei dem eine Gesamtlösung nur gefunden werden kann, indem schon erreichte Ziele kurzfristig wieder zerstört werden müssen. Barrett und Weld haben in [BW92] diese Taxonomie wieder aufgegriffen und um die Klasse der schwer serialisierbaren Probleme erweitert. Mit demselben Papier begann wohl auch der Siegeszug von SNLP, weil mit einer Reihe von Experimenten gezeigt wurde, daß SNLP mit schwer oder gar nicht serialisierbaren Problemen weit weniger Schwierigkeiten hat als

totalordnende Planer. Mit diesem und ähnlichen Untersuchungen entstand also der Eindruck, daß die Reihenfolge, in der die Ziele bearbeitet werden, für Planer wie SNLP unkritisch ist; ein Eindruck, der auch dadurch verstärkt wird, daß die Zielauswahl keinen Backtrackingpunkt darstellt.

In Abschnitt 3.3.5.1 wird an Beispielen verdeutlicht, daß die Zielreihenfolge entgegen diesem Eindruck sowohl die Planungseffizienz als auch die Planqualität stark beeinflußt. Hier soll nun analytisch dargelegt werden, warum die Zielauswahl kritischer ist, als zunächst angenommen werden kann.

Indirekte Festlegung der Reihenfolge durch Phantomisierungen. Um eine Terminierung zu gewährleisten, bevorzugt jede Kontrollstrategie – wenn auch in unterschiedlichem Maße – Phantomisierungs-Links gegenüber der Einführung neuer Operatoren. Diese Phantomisierungs-Links stellen nun einerseits selbst eine Ordnungsrelation dar, verhindern aber vor allem die Anordnung zwischen Erzeuger und Benutzer aller der Schritte, die das durch den Causal Link geschützte Literal bedrohen. Dies ist vor allem in solchen Domänen kritisch, in denen viele Interaktionen auftreten, und auch schon im Anfangszustand Ziele erfüllt sind: Eine falsche Reihenfolge der Zielbearbeitung führt dann häufig dazu, daß diese falsche Reihenfolge durch Phantomisierungs-Links verankert wird, was bei Auftreten von Konflikten eine Anordnung der interagierenden Planstücke unmöglich macht. Das bedeutet also, daß die prinzipielle Möglichkeit partiellordnender Planer, Planstücke unabhängig von ihrer Planungsreihenfolge anzuordnen, in der Praxis häufig stark eingeschränkt ist.

Indirekte Planung. CAPlan erlaubt nur einen einzigen Erzeuger für ein Ziel. Diese Einschränkung führt in Verbindung mit einer falschen Zielreihenfolge zu ganz skurrilen Plansequenzen. Als Beispiel

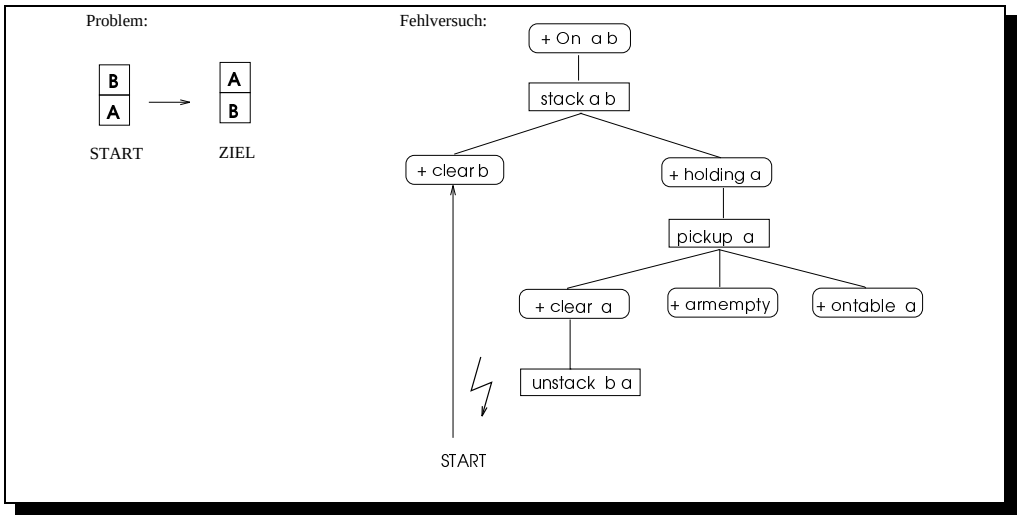

Abbildung 3.3: Ein Beispielproblem. Es sei angenommen, daß die Ziele von links nach rechts bearbeitet werden.

möge die in Abb.3.3 gezeigte Situation dienen, bei der fälschlicherweise mit der Planung von (+ clear b) begonnen wurde. Um die Zeichnung nicht unnötig komplex werden zu lassen, wurde angenommen, daß dieses Ziel schon im Anfangszustand gilt. Dasselbe Problem tritt auf, wenn etwa stattdessen zunächst ein Turm von Klötzchen oberhalb von B abgebaut werden müsste. Die Problematik der obigen Situation besteht nun darin, daß CAPlan nicht einfach den jetzigen Erzeuger 'START' von (+ clear b) durch einen anderen, der veränderten Situation angepaßten Erzeuger ersetzen kann, sondern daß durch Backtracking zunächst alle Möglichkeiten zur Erplanung von (+ holding a) ausgetestet werden, bis schließlich die in Abb. 3.4 gezeigte Möglichkeit gefunden wird. Auffallend ist, daß im Unterplan für (+ clear b) ein Teil der Planungsarbeit für das Ziel (+ holding a) geleistet werden muß, und daß in diesem Unterplan (+ clear b) mehrmals erplant wird. Dieser Effekt tritt in größeren Problemen in noch stärkerem Umfange auf.

Da diese seltsamen Unterpläne jedoch nur nach erheblichen Backtracking gefunden werden können, ist

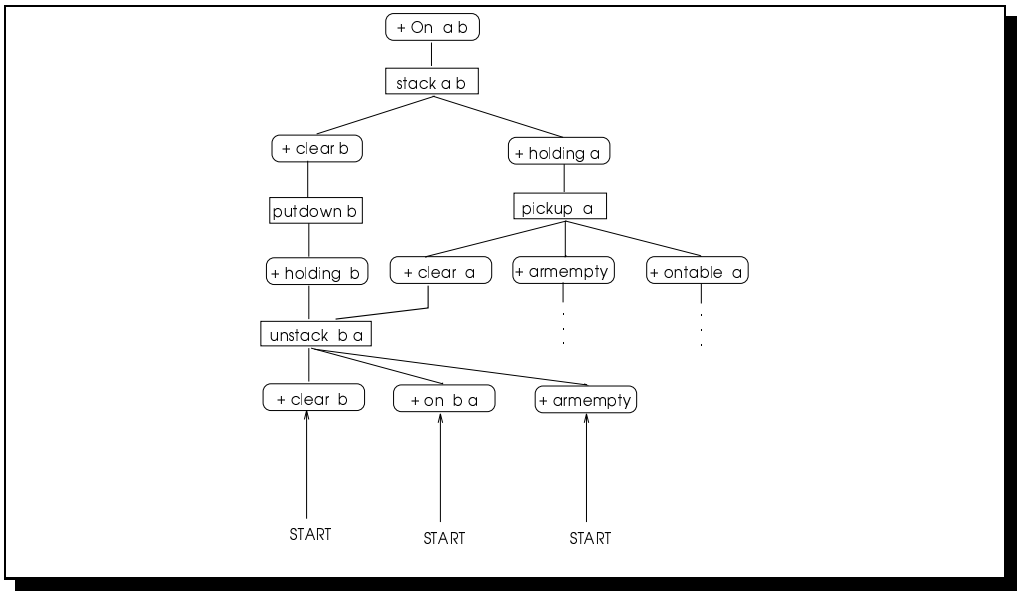

Abbildung 3.4: Lösung durch indirekte Planung

nicht klar, ob partiellordnende Planer in solchen Fällen überhaupt einen Vorteil bieten. Man beachte hierbei, daß bei Planern, für die die Zielauswahl ein Backtrackingpunkt ist, das herkömmliche Goal-Loop-Kriterium eingesetzt werden kann, und daß dort die Zielreihenfolge geändert würde, anstatt die Planung der beiden interagierenden Ziele in einen einzigen Unterplan zu verlagern.

3.1.3.2 Problematik der Operatorauswahl

Zunächst einmal ist zu bemerken, daß durch die explizite Repräsentation von Phantomisierungsoperatoren die Operatormenge ganz erheblich größer wird, was direkt zu einer Erhöhung des Branching-Faktors führt. Die Auswahl eines bestimmten Seiteneffektes stellt eine wesentlich stärkere Festlegung dar, als dies in unsystematischen Planern gemacht wird, die noch nicht einmal zwischen noch nicht bearbeiteten und phantomisierten Zielen unterscheiden.

Ein weiterer wichtiger Punkt ist, daß durch den systematischen Ansatz von SNLP an einer einmaligen Festlegung strikt festgehalten wird, und auch in Beispielen wie in Abb. 3.3, Erzeuger nicht einfach ersetzt werden können. Die einzige Möglichkeit, einen einmal gewählten Operator zu ersetzen, ist das Backtracking. Damit wird die Korrektheit der Operatorwahl kritischer als bei nicht-systematischen Planern, wo in einigen Fällen Backtracking noch durch Nachplanen umgangen werden kann.

Besonders kritisch wird die Operatorauswahl auch durch Phantomvorbedingungen: Ist eine Phantomvorbedingung nicht phantomisierbar, so wird zunächst der gesamte Suchraum des restlichen Problems expandiert, bevor feststeht, daß diese Phantomvorbedingung auch später nicht phantomisiert werden kann. Die Auwahl von Operatoren mit Phantomvorbedingungen sollte daher nur sehr sorgfältig erfolgen, falls nicht durch die Domänenmodellierung feststeht, daß solche Phantomisierungsvorbedingungen tatsächlich immer phantomisiert werden können, falls also diese Vorbedingungen als Operatorauswahlkriterium verwandt werden. Collins und Pryor [CP92] haben die Verwendung solcher Filter-Vorbedingungen für SNLP untersucht und kommen zu dem Ergebnis, daß solche Vorbedingungen nur zu einer Verbesserung des Planverhaltens führen, wenn sie von der Auswahlkontrolle auch tatsächlich als Filter eingesetzt werden.[5] Für partiell ordnende Planer wie SNLP kann dies allerdings nur in Form einer Präferenz auf den Alternativen geschehen, da außer am Ende der Planung die Erfülltheit der Filterbedingungen nicht entschieden werden kann.

[5] Dies wird durch die in Abschnitt 4.3.3.4 beschriebene gewichtete Minimum Conspiracy Strategie realisiert.

3.1.3.3 Nachteil backtracking-basierter Verfahren

Ganz wichtig ist jedoch, daß für CAPlan wie für alle backtracking-basierten Planer gilt, daß frühe Fehler – ganz unabhängig vom Entscheidungstyp – ein Problem praktisch unlösbar machen. Man muß sich dabei vor Augen halten, daß bei frühen Fehlern die Konflikte meist erst viel später erkennbar werden. Beim Backtracking muß dann der gesamte Suchraum für den falschen partiellen Plan untersucht werden. Da kleine partielle Pläne jedoch sehr viele Erweiterungsmöglichkeiten bieten, kann dieser Suchraum schon zu groß sein, um in akzeptabler Zeit durchsucht werden zu können. Intelligentere Backtrackingverfahren wie etwa das für CAPlan implementierte Backjumping lindern das Problem sicherlich etwas, sind aber im allgemeinen immer noch zu schwach.

Es gilt daher, daß *alle* Entscheidungen wichtig sind und jeder Entscheidungspunkt durch geeignete Verfahren unterstützt werden sollte. Ist die Berechnung korrekter Entscheidungen nicht möglich, so muß zumindest versucht werden, unsichere Entscheidungen so weit wie möglich hinauszuzögern, weil der Suchraum stark expandierter Pläne sehr viel kleiner ist. Für 'späte' Fehler bleibt daher das Backtracking oft sehr lokal.

3.1.3.4 Fehlen von Zustandsinformation

Zuletzt noch ein Problem der Kontrolle partiell ordnender Planer an sich: Viele der bekannten Kontrollverfahren für totalordnende Planer und auch die meisten domänenspezifischen Heuristiken beziehen sich auf den Zustand. Ein Beispiel wäre etwa: 'Falls das Klötzchen, an dem Du interessiert bist, auf einem anderen steht, dann nimm den Operator UNSTACK, um es in die Hand zu nehmen.'. Bei partiell ordnenden Planern besteht nun das Problem, daß sich ein Zustand gar nicht bestimmen läßt, daß also im obigen Beispiel nicht überprüft werden kann, ob sich das Klötzchen in der beschriebenen Position befindet oder nicht. Im Rahmen dieser Arbeit wurde dieses Problem meist dadurch gelöst, daß statt dem Zustand eine Obermenge als dessen Approximation verwandt wurde.

Katukam und Kambhampati [KK94b] haben hingegen versucht, den EBL Ansatz so zu modifizieren, daß von den Regeln statt Zustandsbedingungen Bedingungen an die Planstruktur gestellt werden. Sie haben allerdings die Erfahrung gemacht, daß Bedingungen an Planstrukturen zu schwach sind, um darauf eine wirkungsvolle Kontrolle aufbauen zu können. Wir haben diese Erfahrung ebenfalls gemacht, als wir (erfolglos) versucht haben, domänenspezifische Regeln zu finden, die sich nur auf Planstrukturen beziehen.

3.2 Spezifikation der Randbedingungen

Wir werden uns nun mit den Anforderungen an Kontrollmethoden beschäftigen. Für diese Anforderungen ergibt sich unmittelbar eine Dreiteilung:

1. Ziel der Kontrollmethoden sollte sein, die Qualität des Planungsvorganges zu erhöhen.

2. Ziel von Kontrolle ist auch, die Qualität des entstehenden Planes zu verbessern.

3. Schließlich werden auch außer den zuvor genannten Hauptzielen noch weitere nicht-funktionale Anforderungen an eine Kontrollkomponente gestellt.

Die Definition eines Qualitätsbegriffes ist sehr umstritten und nicht zuletzt auch vom Einsatzgebiet des Planers und der Pläne abhängig. Die Diskussion über die Bewertung von Planungsvorgang und erzeugten Plänen ist auch Gegenstand eines neuen, kontroversen Forschungstrends [Wil94][Pol95]. Die nachfolgend aufgeführten Punkte sind deshalb keineswegs als erschöpfend anzusehen, sondern eher als konsensfähige Mindestanforderungen.[6]

[6] Arbeiten, die sich direkt mit dem Begriff der Planqualität und der Planungsqualität auseinandersetzen, sind [PC93][GL94].

3.2.1 Qualität des Planverfahrens

3.2.1.1 Qualitätsaspekte

Die Bewertung des Planungsvorgangs läßt sich in die folgenden Aspekte aufgliedern:

1. Korrektheit
   Unter Korrektheit eines Planverfahrens ist zu verstehen, daß nur korrekte Lösungen (s.u.) erzeugt werden.
2. Effizienz
   Ein Planungsverfahren ist effizient, wenn es sein Ziel bei sparsamem Einsatz von Ressourcen erreicht. Da für uns Speicherprobleme keine Rolle spielen, reduziert sich hier die Effizienz auf die Geschwindigkeit des Planungsvorganges.
3. Effektivität
   Unter Effektivität sei die Fähigkeit verstanden, gute, d.h. qualitativ hochwertige (s.u.) Lösungen zu finden.
4. Mächtigkeit/Vollständigkeit
   Unter Mächtigkeit soll die Größe der Menge der grundätzlich vom Planer erzeugbaren Lösungen verstanden werden.
5. Transparenz und Natürlichkeit
   CAPlan ist als Assistenzplaner konzipiert worden. Damit der Benutzer wirklich sinnvoll auf einzelne Planungsentscheidungen Einfluß nehmen kann, muß ein Mitverfolgen des Planungsvorganges möglich sein. Dies ist nur gewährleistet, wenn der Planer sehr transparent und natürlich vorgeht.
6. Robustheit und Beeinflußbarkeit
   Jedes Assistenzsystem muß beeinflußbar sein und auf Änderungen und Benutzerfehler flexibel reagieren können. Für CAPlan bedeutet dies vor allem, daß auch bei Einsatz von Kontrollverfahren die volle Backtrackingfähigkeit erhalten werden muß. Ganz konkret heißt das, daß entweder der Algorithmus nur verfeinert wird oder für eventuelle Änderungen REDUX-Beweise gefunden werden müssen.

Da diese Qualitätsanforderungen sich zum Teil widersprechen, muß nun festgelegt werden, wo wir Abstriche akzeptieren können und welche Kriterien nicht eingeschränkt werden dürfen.

3.2.1.2 Trade-Off der Qualitätsanforderungen

Um zu bestimmen, wie die verschiedenen Qualitätsmerkmale gegeneinander abgewogen werden sollen, müssen wir uns am gewünschten Einsatzgebiet und dem gegebenen Basisplaner orientieren. Die Ausrichtung am intendierten Einsatzgebiet dient dazu, die Benutzerakzeptanz zu maximieren; die Berücksichtigung des Basisplaners bewirkt, daß die Stärken dieses Algorithmus' durch Kontrolle nicht verwässert werden.

- Wie in Abschnitt 2.2.1 näher erläutert, ist CAPlan ein Assistenzplaner, der insbesondere auf Benutzerinteraktion ausgelegt ist und auf Änderungen durch den Benutzer flexibel und robust reagiert.
- Die hervorstechendsten Merkmale von SNLP sind die Korrektheit und die Systematik der Suche sowie der Least Commitment Ansatz.

Damit ergibt sich für die Gewichtung der einzelnen Qualitätsmerkmale des Planungsvorganges:

1. Bei der Korrektheit und der Robustheit des Planungsverfahrens sollten keine Abstriche gemacht werden. Wie schon erläutert, muß daher bei allen Verfahren die Backtracking-Verträglichkeit sorgfältig überprüft werden.

2. Auch eine Einschränkung des least-commitment-Ansatzes würde einen Kernpunkt des SNLP-Algorithmus verletzen und ist daher unakzeptabel. Es bietet sich allerdings an, den least-commitment-Ansatz noch konsequenter zu verfolgen.

3. Obwohl auch die Systematik eine ganz zentrale Errungenschaft von SNLP ist, sollte hier vielleicht doch ein Abrücken von der *strengen* Systematik erlaubt sein, weil Analysen [SP93, Kam93a] gezeigt haben, daß sich zu strenges Festhalten an diesem Kriterium z.T. sehr nachteilig auf das Laufzeitverhalten auswirkt. Man muß sich hier vor Augen halten, daß Systematik keinen *inherenten* Wert hat, sondern nur ein theoretisches Konstrukt ist, daß als indirektes Maß für die Redundanzvermeidung eingeführt worden ist. Diese Sichtweise macht auch klar, warum es keinen Sinn macht, die Systematik zu maximieren, wenn dadurch der eigentlich interessante Effizienzaspekt beeinträchtigt wird.

4. Zum Aspekt der Mächtigkeit läßt sich anmerken, daß CAPlan schon in der Grundversion die Vollständigkeit des Planers eingeschränkt hat auf die Menge der minimalen Pläne (vgl. 2.2.2). Dies ist sinnvoll, weil dadurch Effizienz und Effektivität gesteigert werden konnten. Unter dem Aspekt der Benutzerakzeptanz ist also durchaus wünschenswert, die Mächtigkeit des Verfahrens noch weiter auf die Menge der 'interessanten' Pläne einzuschränken, wenn dies mit einer Effizienz- oder Effektivitätssteigerung einhergeht.

5. Anders als bei autonomen Planern ist bei interaktiv konzipierten Planern die Transparenz und Natürlichkeit des Planungsvorgangs von zentraler Bedeutung. Jede Kontrolle sollte sich daher unbedingt an diesen Begriffen orientieren, allerdings nicht unter Einschränkung der Korrektheit und der Vollständigkeit bezüglich minimaler Pläne.

Das folgende Schema faßt die obigen Überlegungen noch einmal zusammen.

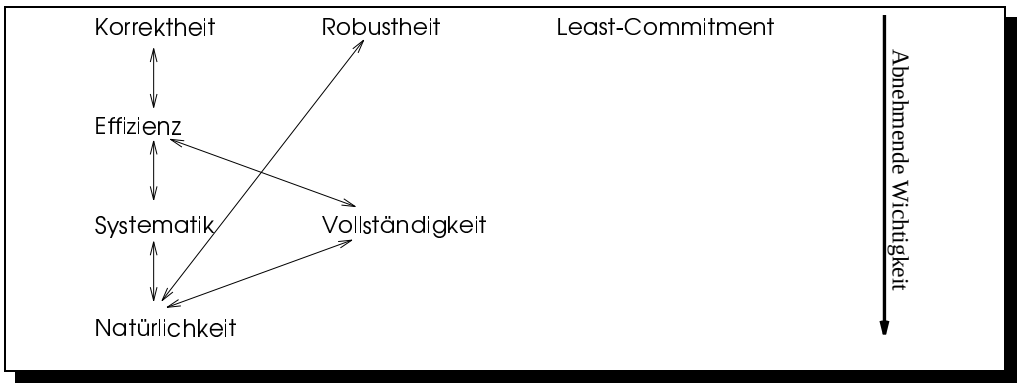

Abbildung 3.5: Trade-Off der Qualitätsmerkmale des Planungsprozesses. In y-Richtung ist die Dominanz abgetragen; Pfeile deuten Konflikte zwischen den Merkmalen an

3.2.1.3 Diskussion der Trade-Offs

Die meisten der in der im obigen Schema angegebenen Trade-Offs dürften unmittelbar einsichtig sein; zwei dieser Konflikte bedürfen allerdings noch einiger Erläuterungen.

Korrektheit contra Effizienz. Die Klasse der Anytime-Planer [Her95, Zil93] beschäftigt sich explizit mit diesem Trade-Off, der vor allem bei zeitkritischen, reaktiven Planungssytemen praxisrelevant ist. Da CAPlan aber nicht für den Einsatz unter Echtzeitbedingungen ausgelegt ist, wurde der Korrektheit des Verfahrens uneingeschränkte Priorität gegenüber der Effizienz eingeräumt.

Natürlichkeit. Um das Kollidieren des Natürlichkeitsaspektes mit beinahe allen anderen Qualitätsansprüchen zu erklären, soll hier kurz ausgeführt werden, was den menschlichen Planungsprozess charakterisiert:

- Das menschliche Kurzzeitgedächnis hat bekannterweise eine Kapazität von etwa 3-7 Einheiten. Um dennoch mit der Komplexität einer Planungsaufgabe zu Rande zu kommen, werden zwei Grundstrategien eingesetzt:
  - Ein hierarchisches Vorgehen, bei dem zunächst einmal ein Grobplan erstellt wird, der dann schrittweise verfeinert wird. Sobald die Planungsaufgabe 'überschaubar' scheint, wird aber lokal geplant. Dies bedeutet auch, daß dann Hin- und Herspringen zwischen Teilaufgaben vermieden wird.
  - Ein divide-and-conquer Ansatz, bei dem die einzelnen Teilprobleme separat gelöst und dann unter Verwendung heuristischer Konfliktauflösungsstrategien zu einer Gesamtlösung zusammengesetzt werden.
- Aus der Beschränkung des Kurzzeitgedächnisses folgt zudem, daß der Mensch nicht systematisch (im technischen Sinne) planen kann; dies ist nur möglich, wenn er unter Verwendung von Hilfsmitteln buchführt über die schon untersuchten Alternativen.
- Alternativ zur Buchhaltung wird häufig die Simulation eingesetzt: Dabei wird die Ausführung eines Teilplanes simuliert (z.B. auf Papier) und das Ergebnis als Ausgangspunkt für die weitere Planung verwendet. Dies bedeutet auch, daß Backward-Chaining meist nicht in Reinkultur, sondern in Kombination mit simulativem Vorwärtsschließen zum Einsatz kommt.
- Statt der systematischen Suche setzt der Mensch auch häufig heuristische Repair-Methoden ein, d.h. er setzt eine Hill-Climbing-Suche ein, bei der inkorrekte Lösungen mit kleinstmöglichem Aufwand solange lokal korrigiert werden, bis eine global konsistente Lösung entsteht.
- Für die menschliche Vorgehensweise ist ebenfalls charakteristisch, vorausschauend zu planen d.h. insbesondere konfliktvermeidend.
- Zuletzt ist menschliches Planen auch durch Opportunismus [HRHR79] und Chaos gekennzeichnet.

Die Nachahmung natürlicher Verhaltensweisen wollen wir natürlich nur auf die Aspekte beschränken, die mit Ansatz von SNLP vereinbar sind. Obwohl die Nachahmung der heuristischen Repair-Strategie bei CSP-Systemen überzeugende Ergebnisse geliefert hat, sollte bei einem SNLP-basierten Planer nicht auf die Vollständigkeit und die Systematik verzichtet werden.

Was bleibt also ?

1. Auch ohne explizite Verwendung eines hierarchischen Planers sollte versucht werden, zuerst einen Grobplan für die wichtigen und/oder kritischen Ziele zu erstellen
2. Konfliktvermeidung
3. Kombination von Backward-Chaining und Simulation
4. Kein unmotiviertes Hin- und Herspringen, d.h. lokal fokussierende Planung

3.2.2 Qualität des entstehenden Planes

Wir wollen uns nun mit der Qualität der entstehenden Pläne beschäftigen.

3.2.2.1 Qualitätsaspekte

Für die entstehenden Pläne lassen sich ganz ähnliche Qualitätsaspekte wie für den Planungsvorgang selbst stellen.

1. Effektivität
   Unter Effektivität eines Planes kann man den Erfüllungsgrad der vorgegebenen Ziele verstehen.

2. Effizienz
   Die Erreichung der Aufgaben mit minimaler Schrittzahl, bzw. (in Systemen mit explizitem Kostenmodell) die Erreichung der Aufgaben unter Minimierung der Kosten.

3. Robustheit und Flexibilität
   Weil in reellen Anwendungen die closed world assumption nicht haltbar ist, müssen die Pläne oft während der Ausführung an eine veränderte Umgebung oder veränderte Zielsetzungen angepaßt werden. Robustheit und Flexibilität mißt den Replanning-Aufwand in solchen Fällen.

4. Natürlichkeit
   Wie von Verfechtern der HTN-Planer [Wil88, CT91] immer gerne betont wird, ist für die Benutzerakzeptanz nicht nur die Korrektheit eines Planes entscheidend, sondern oft auch, inwieweit der Plan den Gepflogenheiten der jeweiligen Domäne entspricht.

3.2.2.2 Diskussion der Planqualitätsaspekte

Auch hier ist ein Abwägen der verschiedenen Qualitätsanforderungen nötig.

Effizienz. Die Effizienz eines Planes hängt sehr stark vom zugrundeliegenden Kostenmodell ab, an sich sogar vom –meist impliziten– Nützlichkeitsmodell [WH94]. CAPlan besitzt bisher keinen Mechanismus zur Integration von Kosten- oder Nützlichkeitsabwägungen und sieht auch nicht vor, nach dem Finden der ersten Lösung nach weiteren Lösungen zu suchen. Obwohl die dazu nötigen Änderungen gering sind, sprengt eine *ganzheitliche* Lösung der Planoptimierung den Rahmen dieser Arbeit. Benötigt würden unter anderem eine komfortable Wissensakquisitions-Schnittstelle, über die unterschiedliche Optimierungskriterien eingegeben werden könnten [DGT94] und eventuell auch Postprocessing-Komponenten, die erzeugte Pläne auf zusätzliche Ausnutzung synergistischer Effekte hin überprüfen [FLY91, YNH92]. Die Optimierung der Planeffizienz wird daher in dieser Arbeit kein explizites Ziel sein, sondern sich höchstens als Seiteneffekt der Planungsoptimierung ergeben.

Natürlichkeit. Die Natürlichkeit von Plänen ist noch schwerer zu definieren als die Natürlichkeit des Planungsvorganges und ist auch in den meisten Fällen domänenabhängig. Wir werden daher die Natürlichkeit der Pläne im Rahmen dieser Arbeit nicht weiter verfolgen, weil dies nur (?) durch einen Apprenticeship-Ansatz möglich ist.

Robustheit und Flexibilität. Zur Robustheit ist schon gesagt worden, daß der SNLP-Ansatz redundante Kausalstrukturen methodisch ausschließt. SNLP-Pläne können also nicht in Situationen robust sein, in denen der Erzeuger eines Zieles ersetzt werden muß. Die Flexibilität der Pläne hingegen ergibt sich als Nebenprodukt des least-commitment Ansatzes; durch weitere Vermeidung voreiliger Constrainteinführungen läßt sich die Menge der Ordnungen sogar noch verkleinern.

3.2.3 Weitere Anforderungen an eine Kontrollkomponente

Nachdem wir jetzt die Zielsetzung einer Kontrollkomponente untersucht haben, formulieren wir nun noch zusätzliche Kriterien, denen eine Realisation genügen sollte:

Modularität. Wie schon in der Einleitung bemerkt wurde, weisen die verschiedenen Planungsdomänen ganz unterschiedliche Schwierigkeitsprofile auf, denen eine Kontrollkomponente Rechnung tragen muß. Die dadurch notwendig gewordene individuelle Anpassung der Kontrolle an die jeweilige Domäne wird am besten durch einen Toolbox-Ansatz unterstützt. Voraussetzung dafür ist eine modulare Realisierung der einzelnen Kontrollmethoden.

Minimale Modifikationen. Aus diesem Toolboxansatz folgt weiterhin, daß die einzelnen Kontrollmethoden die bestehende Kontrollstruktur in CAPlan nur verfeinern bzw. minimal modifizieren sollten. Zu starke Verletzung dieser Lokalitätsbedingung würde zu Einschränkungen der Flexibilität führen. Die Möglichkeit, ganz flexibel maßgeschneiderte Kontrollkomponenten zusammenstellen zu können ist aber unverzichtbar für die Behandlung verschieden strukturierter Kontrollprobleme.

Benutzbarkeit. Schließlich sollte die Integration verschiedener Kontrollansätze benutzerfreundlich gestaltet und durch eine geeignete Oberfäche unterstützt werden.

Dokumentation. Um dem Benutzer oder dem Wissensingenieur die Zusammenstellung einer sinnvollen Kontrollkomponente zu ermöglichen, müssen die einzelnen Methoden bezüglich ihrer Einsatzbedingungen, ihrer Kosten, ihres Nutzens und ihrer Kompatibilität mit anderen Methoden dokumentiert sein.

3.3 Identifizierung und Auswahl grundlegender Kontrollmöglichkeiten

In diesem Abschnitt werden ganz systematisch verschiedene Kontrollebenen und Eingriffsmöglichkeiten identifiziert und festgehalten, welche den zuvor definierten Anforderungen genügen und weiter verfolgt werden sollen.

3.3.1 Die verschiedenen Kontrollebenen

Es lassen sich vier Kontrollebenen ausmachen:

1. Veränderungen der Grobarchitektur von SNLP
   Als Änderungen der Grobarchitektur werden solche Modifikationen bezeichnet, die den Charakter des Algorithmus völlig verändern.

2. Veränderungen der Feinarchitektur
   Veränderungen der Feinarchitektur erhalten zwar den grundsätzlichen Charakter des Algorithmus, hinterfragen aber einzelne Entwurfsentscheidungen und Definitionen des Basisalgorithmus.

3. Kontrollpunktübergreifende Kontrollstrategien
   Kontrollpunktübergreifende Kontrollstrategien beschreiben ganz grundlegende globale Suchstrategien, in die die lokalen Kontrollmethoden eingebettet werden.

4. Auswahlsteuerung an den vorgesehenen Kontrollpunkten.
   Unter Kontrollpunkten werden hier die Stellen im SNLP-Algorithmus verstanden, an denen im Originalalgorithmus eine indeterministische Auswahl [MR91] getroffen wird.Dies sind all diejenigen Stellen im Algorithmus auf Seite 8, an denen die 'Funktion' `chooseAny` verwendet wird.

Im folgenden soll skizziert werden, was auf den einzelnen Ebenen denkbar ist.

3.3.2 Veränderungen der Grobarchitektur

Veränderungen an der Grobarchitektur sind nur dann gerechtfertigt, wenn sie einen inherenten Mangel des Algorithmus beheben. Betrachten wir deshalb kurz die Hauptkritikpunkte [Wil94, Pol95]:

- SNLP ist ein nicht-hierarchischer Planer
  Die Lösung großer Planungsaufgaben läßt sich von Planern, die ausschließlich auf der Ebene der atomaren Schritte planen, nicht bewältigen.

- Die Einflußnahme des Benutzers auf die Art der entstehenden Pläne ist gering.
  Der Benutzer kann zwar interaktiv jede einzelne Entscheidung beeinflussen; gemeint ist hier aber, daß der Benutzer keine Möglichkeit hat, allgemeine Lösungsmuster anzugeben, die die Struktur akzeptabler Pläne beschreiben. HTN-Planer [Wil88, CT91, EHN94] bieten z.B. die Möglichkeit, regelrecht Grammatiken anzugeben, die gewünschte Lösungsverfahren und -typen beschreiben.

- SNLP erscheint oft zu restriktiv in seiner Festlegung auf genau einen Erzeuger. Neben Effizienzproblemen durch häufiges Backtracking hat diese Beschränkung auf einen Erzeuger den Nachteil, daß die entstehenden Pläne wenig robust sind. Damit ist SNLP für den Einsatz in unsicheren, realitätsnahen Domänen ungeeignet.

Welche Lösungsansätze sind denkbar ?

Den ersten Kritikpunkt kann man etwas entschärfen, wenn man einen Abstraktionsmechanismus [YT90, Ber92] auf SNLP aufsetzt, mit Hilfe dessen sich ein hierarchisches Vorgehen simulieren läßt. Solch ein Eingriff in die Grobarchitektur ließe sich dadurch vermeiden, daß man sich auf ein schwächeres Abstraktionskonzept beschränkt und durch Vorbedingungs-Relaxation einmalig eine Abstraktionshierarchie der Ziele berechnet [Kno90]. Diese Variante wurde jedoch nicht weiter verfolgt, sondern lediglich versucht, die verschiedenen Hierarchieebenen der Domäne in die Domänendefinition hineinzukodieren.

Barrett und Weld [BW94] haben vorgeschlagen, HTN-ähnliche Kontrollschemata in PO-Planern als lokale Auswahlkriterien zu verwenden. An den einzelnen Kontrollpunkten sollen dann nur solche Alternativen ausgewählt werden, die sich durch eines der Kontrollschemata parsen lassen. Obwohl dadurch immer noch keine hierarchische Formulierung der Kontrollkonstrukte möglich ist, wäre diese Variante wohl relativ problemlos in CAPlan zu integrieren. Denkbar wäre auch, CAPlan um eine zentrale Steuerungskomponente zu erweitern, die solche Kontrollschemata interpretiert. Beide Möglichkeiten wurden hier jedoch nicht weiter in Betracht gezogen, da dies im Rahmen eines ganzheitlichen (lernenden) apprenticeship-Ansatzes viel sinnvoller ist.

Es bleibt noch zu bemerken, daß CAPlan durch seine Erweiterungen der Modellierungssprache (siehe Abschnitt 2.1.2) bereits ausdrucksstärker als SNLP ist. Beispielsweise stellt die Differenzierung der Vorbedingungen in CAPlan eine einfache Form der typed conditions [TDD94] dar, wie sie auch in HTN-Planern verwendet werden.

Es gibt sog. Multi-Contributor Planner [Kam92, Kam94], die mehrere Erzeuger zulassen. Für diese Änderung müssen allerdings fast alle der Konzepte von SNLP erweitert werden, insbesondere das Konzept der Threats. Zudem verliert der Algorithmus seine Systematik. Einer empirischen Auswertung Kambhampatis zu Folge [Kam93a] sind diese Planern aber am effizientesten, weil sie Commitment und Systematik am ausgewogensten verfolgen.[7] Um die Modifikationen möglichst gering zu halten, wird hier auch dieser Ansatz nicht weiter verfolgt; dies geschieht auch mit Hinblick darauf, daß CAPlan durch das REDUX-Systems zumindest eine effektive Unterstützung des Replanning bietet.

3.3.3 Veränderung der Feinarchitektur

Welche Punkte der Feinarchitektur sind diskussionswürdig?

[7] Vergleiche dazu die Graphik 2.5 auf Seite 16: MP und MP-I sind Multi-Contributor Planner.

Spezialisierung des Threat-Begriffes. Es ist vorgeschlagen worden, diesen Begriff auf die *notwendigen* Threats einzuschränken, also auf solche Bedrohungen, die nicht vom Eintreten bestimmter Variablenbindungen abhängig sind. Durch diese naheliegende Spezialisierung ließe sich die Menge der Protektionsziele stark einschränken. Kambhampati [Kam93b] hat darauf hingewiesen, daß dadurch die Systematik erhalten bleibt, weil der Systematikbegriff immer nur relativ zu einem Threatbegriff definiert ist. Eine zweite Spezialisierungsmöglichkeit ist, nur negative Threats zu bearbeiten. Knoblock und Yang [KY95] haben diese Möglichkeit für eine künstliche Domäne ausgetestet, für die die Menge positiver und negativer Threats variiert werden konnte. Die Analyse ergab, daß die Mitbetrachtung positiver Threats das Planverfahren kaum ineffizienter macht und in den meisten Fällen deutliche Vorteile bringt. Kambhampati [Kam93a] liefert dafür auch eine theoretische Erklärung und gibt Beispiele dafür an, daß das Nichtbeachten positiver Threats SNLP sogar ineffizienter machen kann als Planer, die keine Causal Links verwenden. Diese Spezialisierung wird deshalb nicht weiter aufgegriffen. Zudem existiert in CAPlan schon die Option, die Behandlung positiver Threats abzustellen.

Verallgemeinerung des Threat-Begriffes. Wenn man zur Bestimmung eines Threats nicht nur simples Matchen zuläßt, sondern auch Inferenzen mit Hilfe der Axiome, erhält man ein 'vorausschauendes', stärkeres Threat-Konzept.

Präferenz der Threat-Auflösung. Die Frage, inwieweit es notwendig ist, Threats sofort aufzulösen, wurde in der Literatur ausgiebig untersucht: Harvey [HGS93] hat bewiesen, daß die Systematik des Suchverfahrens unabhängig vom Zeitpunkt der Threat-Auflösung ist. Kambhampati hat angemerkt, daß Harvey den Begriff der Systematik nur in der Bedeutung 'Lösungssystematik' (vgl. Abschnitt 2.3.6) verwendet und daß die strenge Systematik bei Änderung des Threat-Auflösungszeitpunktes nicht mehr garantiert werden kann. Zudem zeigt er an einem Beispiel, daß der Suchraum eines nur lösungssystematischen Planers *größer* sein kann als der eines unsystematischen Planers. Smith und Peot [SP93] schließlich betonen, daß die *strenge* Systematik erhalten bleibt, solange man nachweisen kann, daß alle noch unaufgelösten Threats auflösbar sind, d.h. solange man keine Pläne weiter expandiert, die eigentlich inkonsistent sind. Wegen des ohnehin zweifelhaften Nutzens der Systematik für die Effizienz des Planungsvorgangs sind Änderungen des Threat-Auflösungszeitpunktes eine nähere Untersuchung wert.

Änderung der zugelassenen Threat-Auflösungsmöglichkeiten.
Konkrete Änderungsvorschläge sind:

Zusätzliches Verwenden des White-Knight-Kriteriums: Die zusätzliche Verwendung des White-Knight-Kriteriums ist zwar in vielen Fällen äußerst nützlich, verwandelt aber SNLP wieder in einen unsystematischen Planer und macht letzlich auch die Buchführung mittels Causal Links sinnlos.

Verzicht auf Separationsoperatoren: Wie in Abschnitt 2.1.3 schon näher erläutert, gibt es $2^n - 1$ Separationsoperatoren für ein Prädikat mit n Argumentvariablen. Dies führt offensichtlich zu einem stark erhöhten Verzweigungsgrad. Der Verzicht auf Separationsoperatoren ist äquivalent zur oben angesprochenen Spezialisierung des Threatbegriffes auf notwendige Threats.

Gemeinsame Auflösung aller nach einem Schritt aufgetretenen Threats: Es scheint sinnvoll, die Menge der Threats zuerst zu analysieren und dann durch geeignete Maßnahmen zusammen zu lösen, anstatt die einzelnen Threats separat aufzulösen. Da CAPlan aber im Gegensatz zu SNLP darauf ausgelegt ist, Threats einzeln aufzulösen, kann dieser Ansatz nur unter Zuhilfenahme einer Art Cache realisiert werden, der die Analyse-Ergebnisse enthält und dann schrittweise abgearbeitet wird. Diese Erweiterung wäre aber in sich abgeschlossen und ist daher mit der Lokalitätsbedingung für Modifikationen verträglich.

Generalisierung des Inkonsistenzbegriffes. Durch Verwenden von mächtigeren Inkonsistenzbegriffen (z.B. mit Hilfe der Axiome) können inkonsistente Domänenoperatoren, Phantomisierungsoperatoren und auch unauflösbare Threatmengen früher erkannt werden und somit die Expansion fruchtloser Teile des Suchraums vermindert werden.

3.3.4 Kontrollstrategien

Basissuchstrategien

CAPlan verwendet Tiefensuche. Dies liegt zum einen daran, daß SNLP als Tiefensuche-Algorithmus vorgestellt worden ist, und zum anderen daran, daß REDUX als JTMS besonders zur Unterstützung dieser Suchstrategie geeignet ist. Beides schließt jedoch den Einsatz anderer klassischer Suchverfahren, wie etwa Best-First, Branch-And-Bound usw. nicht grundsätzlich aus.

Im Gegensatz zur Tiefensuche betrachten jedoch alle diese alternativen Suchstrategien eine *Menge* möglicher Pläne. Diese Option wird von der derzeitigen Systemarchitektur nicht unterstützt. Eine Realisationsmöglichkeit wäre aber zum Beispiel, alle aktuell nicht bearbeiteten Pläne durch künstlich eingeführte Rückzugsbedingungen 'auszuschalten' und nach jedem Planschritt den jeweils besten Teilplan durch Entfernung dieser künstlichen Rückzugsbedingungen wieder anzuschalten.[8] Benötigt würde außerdem eine Buchhaltung, die die Bewertungen und die 'Schaltkombinationen' der alternativen Teilpläne verwalten würde.

Da Frank Weberskirch sich in näherer Zukunft mit diesen Ansätzen beschäftigen will, werden wir uns hier auf Varianten der Tiefensuche beschränken. Ein weiterer Grund ist, daß Breitensuche nicht von Heuristiken profitieren kann, weil immer alle Alternativen gleichzeitig betrachtet werden[MBD94].

Suchstrategien im Rahmen der Tiefensuche

Auch innerhalb der Tiefensuche sind verschiedene Strategien denkbar:

- LIFO bzw. FIFO-Strategien
- Planung in Richtung oder entgegengesetzt zur Ausführungsreihenfolge
- Least-Commitment im ursprünglichen Sinne

Ausschlaggebend bei der Entscheidung für eine Basissuchstrategie ist die Frage, ob einige der gewünschten lokalen Kontrollmethoden Zustandsinformationen benötigen. Werden die Ziele nämlich in LIFO Manier bearbeitet, so bedeutet dies für die Zustandsberechnung, daß angenommen wird, die Ziele blieben ungeordnet. Dies trifft für den Großteil der Domänen nicht zu.

Bei der Planung entgegengesetzt zur Ausführungsreihenfolge (d.h. bei striktem Backward-Chaining) gilt ebenfalls, daß für die Zustandsberechnung implizit angenommen wird, daß Schritte, die nicht zur Erplanung des aktuell verfolgten Zieles dienen, nicht zwischen den Anfangszustand und die aktuelle Position treten. Diese Annahme ist ebenfalls unzulässig.

Ein echter Least-Commitment Ansatz unterstützt hingegen Verfahren, die Zustandsinformation verwenden, weil durch diesen Ansatz die Zustandsinformation sicherer und damit verläßlicher wird. Im Rahmen dieser Arbeit soll daher nur eine Kombination von FIFO, Planung in Ausführungsreihenfolge und Least Commitment weiterverfolgt werden. Ein separater Abschnitt in Kapitel 4 für Kontrollstrategien entfällt daher.

3.3.5 Auswahlsteuerung an den einzelnen Kontrollpunkten

Die Kontrolle durch Auswahlsteuerung an den von SNLP vorgesehenen Punkten soll so gestaltet sein, daß die Vollständigkeit unbeeinträchtigt bleibt. Wir legen deshalb fest, daß die lokalen Kontrollmethoden die Alternativen nur bewerten, nicht aber endgültig ausschließen dürfen. Grund hierfür ist auch, daß es zum Teil sehr schwierig ist, Beweise zu konstruieren, die sich in das REDUX System einbinden lassen. Durch die ausschließliche Verwendung als Präferenzordnung können Veränderungen und Erweiterungen des REDUX Systems vermieden werden. Somit bleibt die volle Backtracking-Fähigkeit und damit die Robustheit des Systems garantiert.

[8] Dieser Vorschlag stammt von Jürgen Paulokat.

Im folgenden sollen die vier prinzipiellen Entscheidungsgruppen daraufhin untersucht werden, ob eine durch Kontrolle veränderte Auswahl überhaupt Einfluß auf die Planungseffizienz und/oder die Art der entstehenden Pläne hat. Ferner soll untersucht werden, ob es prinzipiell Kriterien und Informationen gibt, die solch eine Auswahl steuern können.

3.3.5.1 Zielauswahl

Bedeutung der Zielauswahl. Bei Einführung des ersten partiellordnenden Planers NOAH schien durch die weniger restriktive Planrepräsentation das Problem der richtigen Zielreihenfolge endgültig gelöst zu sein. Drummond und Currie [DC89] merken jedoch an, daß NOAH vor allem aufgrund seiner ausgefeilten Zielauswahlkriterien in der Lage war, Probleme wie die Sussman Anomaly zu lösen und weniger aufgrund der neuen Planrepräsentation.

Kambhampati zufolge reduziert vor allem der Einsatz starker Schutzmechanismen die Problematik der Planungsreihenfolge [Kam94]. Man sollte sich jedoch bezüglich des Einflusses starker Schutzmechanismen keine allzugroßen Hoffnungen machen: selbst bei moderat großen Plänen führt eine falsche Entscheidung oft zu einer ganzen Flut von Konflikten, die nicht zusammen konsistent auflösbar sind. SNLP merkt dies zwar, nachdem er alle Auflösungsmöglichkeiten ausprobiert hat; besser ist jedoch allemal, diese Konfliktlawine von vornherein zu vermeiden.

Die folgenden Beispiele sollen verdeutlichen, wie die Planungsreihenfolge der Ziele Einfluß auf Planungseffizienz und Planqualität nimmt:

Planungseffizienz
Entscheidet sich der Planer dafür, ein Ziel zu bearbeiten, für das sich die Entscheidungsgrundlage zur Auswahl des Operators im Laufe der nachfolgenden Planung noch sehr stark ändert, so steigt die Wahrscheinlichkeit, eine falsche Operatorauswahl zu treffen.

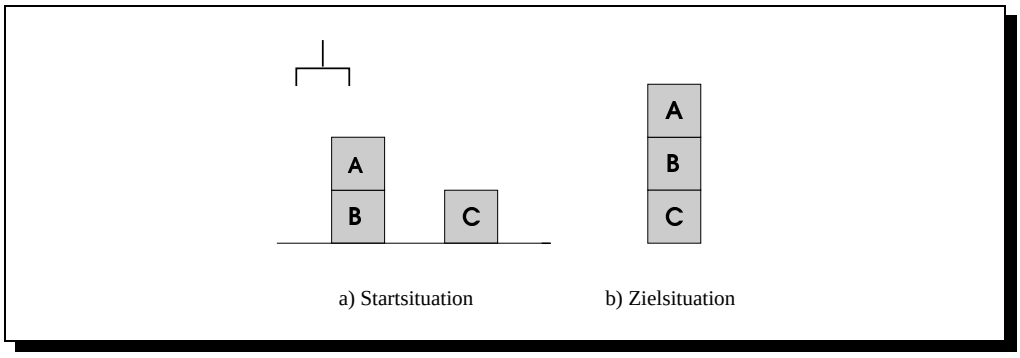

Abbildung 3.6: Ein Beispiel aus der Blocksworld

Wenn in obigem Beispiel der Planer zuerst das Ziel (+ On a b) bearbeitet, so wird es mit der Startsituation phantomisiert werden. Da bei der Planung des zweiten Zieles (+ On b c) der Block a vom Block b aber heruntergenommen werden muß, ist diese Phantomisierung falsch. Weil diese Entscheidung jedoch ganz unten im Entscheidungsstack liegt und der Konflikt erst sehr spät auftritt, expandiert der Planer einen riesigen Suchraum, bevor er die falsche Phantomisierung zurücknimmt. Plant er jedoch zuerst (+ On b c), dann liegt die falsche Entscheidung bei Auftreten des Konfliktes ganz oben auf dem Stack und kann ohne Aufwand durch den richtigen Operator ersetzt werden.

Planqualität
Man stelle sich ein Szenario vor, in dem g1 und g2 zu erreichende Ziele sind. Ferner stehen zwei Operatoren zur Erplanung von g2 zur Verfügung, op1 mit einem einzigen Effekt g2 und op2 mit einem Haupteffekt g3 und dem Seiteneffekt g2. Weiterhin wollen wir annehmen, daß g3 eine Vorbedingung von g1 ist, und daß zur Erplanung von g3 nur der Operator op2 zur Verfügung steht.

Beginnt man nun mit der Planung von g2, so wird der Planer op1 als Erzeuger von g2 auswählen. Plant er dann g1, so muß er zur Planung von g3 op2 auswählen, der als Nebeneffekt g2 erzeugt. Der

Planer erkennt die Redundanz zwar, 'löst' den Konflikt aber durch eine Anordnung. Das Ergebnis ist ein suboptimaler Plan (siehe Abb. 3.7, a)). Beginnt man stattdessen mit der Planung von g1,

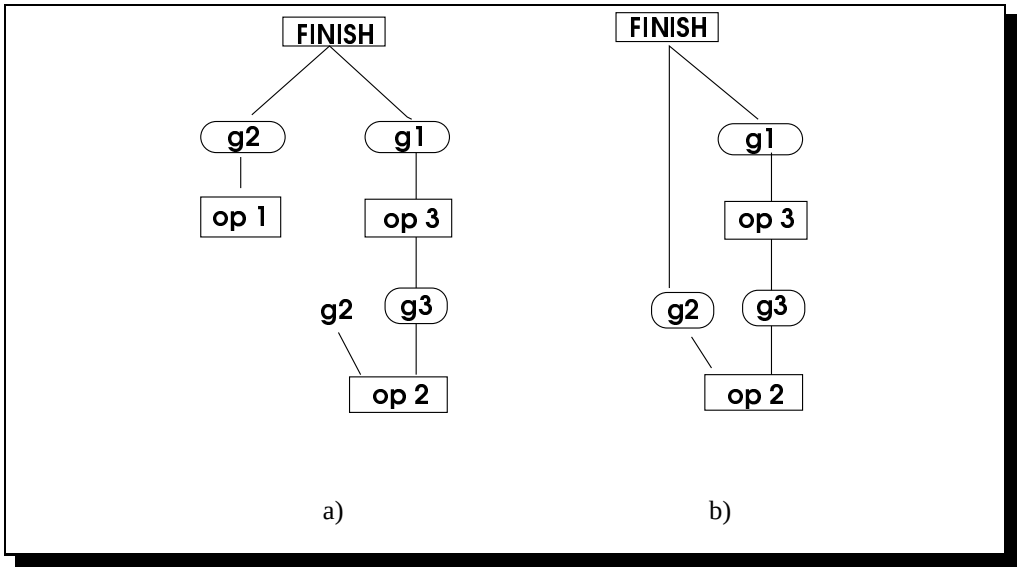

Abbildung 3.7: Auswirkung der Zielreihenfolge auf die Planqualität: links wurde mit der Planung von g2 begonnen, rechts mit g1.

so erkennt der Planer bei Bearbeitung des Zieles g2, daß eine Phantomisierung mit op2 möglich ist, und führt keinen zusätzlichen Schritt ein(siehe Abb. 3.7,b)). Obwohl das Szenario etwas konstruiert erscheint, begegnet man diesen Effekten während der Planung sehr häufig.

Steuerungsfähigkeit der Zielauswahl. Da sich das Problem der richtigen Zielauswahl bei totalgeordneten Planern noch viel stärker stellt, existiert ein breites Spektrum an Methoden und Heuristiken für die Zielauswahl. Viele dieser Verfahren sind jedoch nur eingeschränkt oder gar nicht für Causal Link Planer einsetzbar.

3.3.5.2 Reihenfolge der Threat-Auflösung

Bedeutung der Threat-Auflösungsreihenfolge. Die Reihenfolge der Threat-Auflösung hat nur insofern Bedeutung, als auch diese Entscheidungspunkte beim Backtracking invers zu ihrer Einführungsreihenfolge bearbeitet werden. Deshalb sollte versucht werden, Threats für die die Auflösungsart unsicher ist, möglichst spät zu bearbeiten.

Steuerungsfähigkeit der Threatauflösungsreihenfolge. Man kann die Wahrscheinlichkeit, daß eine Threatauflösungsmöglichkeit richtig ist, reziprok zur Anzahl der möglichen Alternativen definieren. Durch genauere Bestimmung der Menge konsistenter und damit tatsächlich verwendbarer Auflösungsmöglichkeiten läßt sich also noch genauer abschätzen, welche Threats am eindeutigsten auflösbar sind und daher zuerst bearbeitet werden sollen.

3.3.5.3 Operatorauswahl

Bedeutung der Operatorauswahl. In den meisten Domänen hat die Auswahl der Operatoren Einfluß auf Planungseffizienz und Planqualität. Bei Causal Link Planern verschärft sich dieses Problem noch dahingehend, daß gleich die erste Entscheidung richtig sein muß, da die Systematik der Suche ein Nachplanen verbietet und eine einmal getroffene Entscheidung nur durch Backtracking revidiert werden kann.

Steuerungsfähigkeit der Operatorauswahl. Zur Steuerung der Auswahl sind aus der Literatur etliche Verfahren bekannt, die jedoch – ganz analog zu den Zielauswahlmechanismen – nur bedingt übertragbar sind.

3.3.5.4 Protektionsoperatorauswahl

Bedeutung der Threatauflösungsart. Das Auflösen von Threats geschieht immer durch Einführen von Constraints. Daher werden Planungseffizienz und Planqualität direkt beeinflußt:

Planungseffizienz
Ein Constraint kann - probleminherent - eine Lösung völlig ausschließen. Treten die daraus resultierenden Inkonsistenzen aber erst sehr viel tiefer im Suchbaum auf, so wird dieser falsche Threatauflösungsconstraint erst nach großem Backtrackingaufwand wieder zurückgenommen.

Planoptimalität
Die Auswirkungen einer falschen Threat-Auflösung auf die Planoptimalität werden schnell offensichtlich, wenn man sich vorstellt, daß ein Constraint alle minimalen Lösungen ausschließt, aber nicht zum Ausschluß aller suboptimalen Lösungen führt. Der Planer ist dann gezwungen, eine suboptimale Lösung auszugeben.

Steuerungsfähigkeit der Threat-Auflösung. Aus der Literatur ist recht wenig zur Auswahl von Threat-Auflösungsmöglichkeiten bekannt. Es wurden zwar sehr häufig trickreiche Konfliktauflösungsheuristiken fest in die Planer eingebaut; die meisten dieser Konfliktauflösungsschemata fallen allerdings in die Kategorie der Repair-Methoden und sind daher nicht für systematische Planer einsetzbar. Im Prinzip läßt sich jedoch die Frage nach der Steuerungsfähigkeit darauf reduzieren, ob die Wirkung eines Constraints auf den nachfolgenden Planungsprozeß abzuschätzen ist. Da aus dem Bereich der CSP Systeme durchaus solche Kriterien und Heuristiken bekannt sind, scheint eine –wenn auch grobe– Abschätzung nicht völlig ausgeschlossen.

Kapitel 4

Lösungsansätze

In diesem Kapitel werden wir nun ganz konkrete Kontrollverfahren präsentieren. Zunächst werden aber allgemeine, abstrakte Heuristiken eingeführt, denen wir dann die vorgestellten Methoden zuordnen können. Ferner werden im Abschnitt 4.1.2 Hilfsmittel vorgestellt, die von mehreren Kontrollverfahren verwendet werden. Schließlich wird in Abschnitt 4.1.3 das Schema erläutert, mit dem die Verfahren klassifiziert werden. In den nächsten Abschnitten werden dann Kontrollverfahren der einzelnen Kontrollebenen vorgestellt. Für die Kontrollebenen der Grobarchitekturänderungen und der Kontrollstrategien gibt es keinen eigenen Abschnitt, weil Kontrollverfahren auf diesen Ebenen schon im letzten Kapitel ausgeschlossen worden sind.

4.1 Grundlagen

4.1.1 Grundlegende Heuristiken

In diesem Abschnitt soll es darum gehen, ganz universelle Problemlösungsheuristiken zu umreißen. Dies wird uns im folgenden erlauben, die vorgestellten Kontrollverfahren als Instanzen dieser allgemeinen Heuristiken zu erkennen. Letztlich sind alle Kontrollverfahren Operationalisierungen allgemeinerer Heuristiken und können hierarchisch angeordnet werden [Len83]. Ginsberg und Geddis empfehlen, Kontrollverfahren in eine domänenunabhängige Heuristik und die Domänenspezifische (Meta-) Information aufzuspalten. Während ich das aus Effizienzgründen nicht für vorteilhaft halte, fördert die Kenntnis der zugehörigen domänenunabhängigen Prinzipien aber zweifellos das Verständnis der Kontrollverfahren.

Look Ahead Das Prinzip des Look Ahead besteht darin, vorausschauend zu planen. Im Bereich der Constraint Satisfaction Probleme versteht man unter Look Ahead eine Strategie, bei der einer Variable kein Wert zugewiesen wird, wenn dadurch der Wertebereich einer anderen Variable leer wird. Veloso und Blythe [BV92] haben dies auf die Planung in der Weise übertragen, daß Operatoren, für die keine konsistente Bindung gefunden werden kann, nicht ausgewählt werden. In dieser Arbeit soll der Begriff des Look Ahead allerdings wieder in seiner ursprünglichen Bedeutung verwandt werden: Sackgassen werden durch zusätzliche Berechnungen früher als solche erkannt und gar nicht erst betreten.

Small-Is-Quick Dieser Begriff stammt von Pearl [Pea84] und besagt, daß kleine Pläne auch schnell geplant werden können, weil jeder neue Schritt nicht nur ein Ziel löst, sondern auch mehrere neue erzeugt. Schon allein aus Effizienzgründen sollte daher versucht werden, ein Problem mit möglichst wenig Operatoranwendungen zu lösen.

Opportunismus Das Opportunismus-Prinzip ist eine Variante der Small-Is-Quick Heuristik und beschreibt das Bestreben, die Ausnutzung von Seiteneffekten zu maximieren.

Maximale Informiertheit Maximale Informiertheit besagt, daß sichere Entscheidungen bevorzugt werden, wobei 'sicher' hier für 'mit hoher Wahrscheinlichkeit richtig' steht. Die Sicherheit einer Entscheidung hängt von der Bewertung der ausgewählten bzw. der zurückgewiesenen Möglichkeiten ab und damit von Kontrollverfahren, die Alternativen bevorzugen oder zurückstellen. Das Prinzip der maximalen Informiertheit bewirkt, daß unsichere Entscheidungen hinausgezögert werden. Dadurch wird eine baumartige Struktur des Suchraumes begünstigt und damit die Wahrscheinlichkeit, daß Backtracking lokal bleibt.
Beispiel: In Abb. 4.1 besteht für Ziel 1 nur eine Lösungsmöglichkeit . Für Ziel 2 sei nur die letzte Lösungsmöglichkeit mit der Lösung von Ziel 1 verträglich. Beginnt man nun wie in der Zeichnung links dargestellt, so bleibt das Backtracking zum Finden der mit der Lösung von Ziel 1 verträglichen Lösung lokal. Rechts hingegen beschränkt sich das Backtracking nicht nur auf den Suchraum von Ziel 2, sondern auch Ziel 1 wird mehrmals geplant.

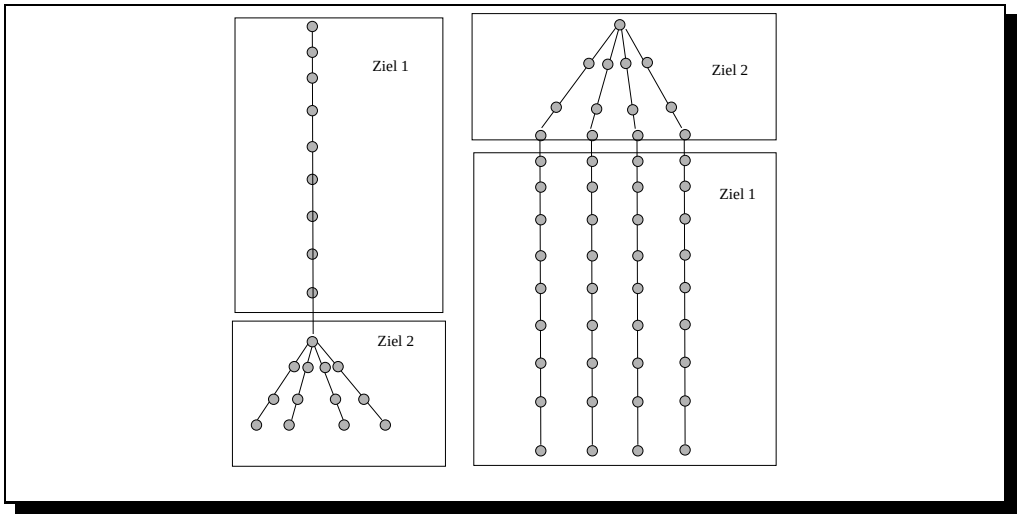

Abbildung 4.1: Einfluß des Prinzips der maximalen Informiertheit auf die Suchraumstruktur

Bottleneck Planung / most constrained first Bottleneck Planung ist eine Spielart der 'Maximalen Informiertheit', die die Sicherheit der Entscheidung daran festmacht, daß die Menge der Alternativen für diese Entscheidung gering ist [Mus94]. Bottleneck Planung ist spezieller als die Heuristik der maximalen Informiertheit, weil hier keine Präferenzinformation mitberücksichtigt wird.

LeastCommitment Least-Commitment ist ein überfrachteter Begriff, der in der Planung häufig als Synonym für 'partiell geordnete Planrepräsentation' benutzt wird. Hier ist aber die wortwörtlichen Bedeutung gemeint, also das Prinzip, sich so wenig wie möglich festzulegen bzw. sich nur festzulegen, wenn eine Entscheidung unumgänglich ist. Least-Commitment wird meist mit dem Prinzip der maximalen Informiertheit verwechselt. Der Unterschied ist, daß maximale Informiertheit nur das Treffen *falscher* Entscheidungen, least commitment aber *jede* Art von Festlegung vermeidet. Least Commitment wird auch häufig mit dem folgenden Prinzip verwechselt:

Least Constraining Least Constraining besagt, daß solche Entscheidungen bevorzugt werden, die die partielle Lösung möglichst wenig beschränken, weil damit die Menge der möglichen Vervollständigungen der partiellen Lösung groß bleibt und damit die Wahrscheinlichkeit, daß unter diesen Vervollständigungen auch eine Lösung ist.

Monotoner Fortschritt Monotoner Fortschritt ist auch als Hill Climbing bekannt: Beim Erfüllen eines Zieles wird vermieden, bereits erfüllte Ziele wieder zu zerstören.

4.1.2 Hilfsstrukturen

Für die Berechnungen der verschiedenen Kontrollmethoden wird eine Reihe von Hilfskonstrukten benötigt, die hier kurz vorgestellt werden sollen.

4.1.2.1 Axiome

Domänenaxiome sind mit die mächtigsten Hilfsmittel, die Planern intelligentere Entscheidungen ermöglichen. Durch Domänenaxiome kann der Planer Inferenzen aus der Plansituation ableiten, die weit über die Aussagen hinausgehen, die sonst möglich sind. Der Begriff des Axioms wird in der Planung allerdings höchst mehrdeutig gebraucht. Meistens wird mit Axiomen aus der Gültigkeit eines Situationsmerkmales auf die Ungültigkeit eines anderen geschlossen [KK94a]. Axiome werden aber z.B. auch eingesetzt, um die Vollständigkeit von Zuständen zu beschreiben oder um Typinformationen von Objekten zu inferieren [Etz90].

In CAPlan wird durch Axiome die gegenseitige Ausschließlichkeit bestimmter Zustände modelliert, ähnlich wie dies auch Etzioni [Etz90] tut. Allerdings benötigt er dazu bis zu vier verschiedene Klassen von Axiomen, während wir mit einer auskommen.

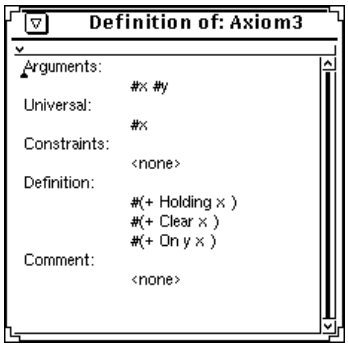

Abbildung 4.2: Ein Axiom aus der Blocksworld

Syntax der Axiome. Wie in Abb.4.2 ersichtlich besteht ein Axiom aus einer Liste von Literalen[1], die die Menge der sich ausschließenden Zustände beschreibt. Zusätzlich können Typ- und Gleichheitsconstraints angegeben werden. Maximal ein Argument kann allquantifiziert werden.

Semantik der Axiome. Die oben angegebene Syntax wird wie folgt interpretiert:

- nicht allquantifizierte Variablen sind existentiell quantifiziert.
- Es gibt nur *eindeutige* Existenzquantoren (i.Z. $\exists!$), d.h. für nicht allquantifizierte Variablen wird angenommen, daß genau eine Belegung existiert.
- Die Literale der Liste sind implizit durch XOR (i.Z. $\oplus$) verknüpft
- Die Constraints verstehen sich als Anwendbarkeitsbedingung des Axioms

Das obige Beispiel bedeutet also

$$\forall x \exists! y((+ \text{ Holding x}) \oplus (+ \text{ Clear x}) \oplus (+ \text{ On y x}))$$

[1] Die Liste steht im Slot 'Definition'

Deduktionsmöglichkeiten. Die Axiome können auf zwei Arten eingesetzt werden:

1. Schlußfolgerungen aus der Gültigkeit eines Literals
   Ist bekannt, daß ein Literal der Liste gültig ist[2], so kann daraus auf die Ungültigkeit der anderen Literale geschlossen werden.

2. Schlußfolgerungen aus der Ungültigkeit eines Literals
   Ist bekannt, daß ein Literal der Liste nicht gültig ist, so kann daraus geschlossen werden, daß eines der anderen Literale gültig ist. Ist bekannt, daß keines der Literale gültig ist, so kann die Unmöglichkeit dieser Situation hergeleitet werden.

Beispiel: Mit obigem 'Axiom 3' können folgende Schlüsse gezogen werden:

$$(+ \text{ Clear x}) \longrightarrow ((- \text{ Holding x}) \wedge (- \text{ On y x}))$$
$$(- \text{ Clear x}) \longrightarrow ((+ \text{ Holding x}) \oplus (+ \text{ On y x}))$$

Modifikationen, die sich durch die Verwendung von Variablen ergeben. Die Schlußfolgerungen sind zunächst einmal für die Aussagenlogik vorgestellt worden. Die Axiome sind aber in der Regel für variable Argumente definiert, so daß geringfügige Anpassungen nötig sind.

Zunächst wird eine Substitution σ bestimmt, die das Anfrageliteral mit einem der Disjunktionsliterale unifiziert. Diese Substitution wird dann auf die Schlußfolgerung angewandt. Nicht durch die Substitution gebundene Axiomvariable werden dabei als allquantifizierte (im Falle gültiger Anfrageliterale) bzw. existenzquantifizierte Variablen interpretiert.

Wird das Anfrageliteral mit einem Disjunktionsliteral unifiziert, das eine existentiell quantifizierte Variable enthält, so muß dem Ergebnis eine Kopie dieses Disjunktionsliterals hinzugefügt werden, bei der die existentiell quantifizierte Variable nicht substituiert, sondern durch eine neue Variable ersetzt wird. Für diese neue Variable wird ein Constraint eingeführt, der eine Bindung an den Wert verbietet, durch den die existentiell quantifizierte Axiomvariable substituiert worden war. Interessanterweise hat Etzioni diesen Spezialfall übersehen und in solchen Fällen schlichtweg falsche Schlüsse gezogen.

Beispiel:

$$(+ \text{ On a b}) \longrightarrow ((- \text{ Holding b}) \wedge (- \text{ Clear b}) \wedge (\forall y(y \neq a) : (\text{- On y b}))$$
$$(- \text{ On a b}) \longrightarrow ((+ \text{ Holding b}) \oplus (+ \text{ Clear b}) \oplus (\exists y(y \neq a) : (+ \text{ On y b}))$$

4.1.2.2 PSGs – Problem Space Graphen

Problem Space Graphen, kurz PSGs, sind von Etzioni [Etz90, Etz93] für den totalordnenden Planer PRODIGY eingeführt worden. PSGs sind eine kompakte Darstellung des Problemraumes: sie stellen alle Möglichkeiten dar, ein einzelnes Ziel zu erplanen.

Technisch gesehen sind PSGs AND-OR-Graphen, die für jedes in der Domäne erplanbare Literal berechnet werden und auf vielfältige Weise zur statischen Domänenanalyse eingesetzt werden können. Die disjunktiv verknüpften Nachfolgerknoten eines Zielknoten sind dabei alle für dieses Ziel relevanten Operatoren; die konjunktiv verknüpften Nachfolgerknoten eines Operatorknoten sind dessen Vorbedingungen. Die rekursive Expansion bleibt durch Anwendung mehrerer Abbruchbedingungen endlich. Abbildung 4.3 zeigt einen PSG aus der Blocksworld-Domäne.

Die Abbruchbedingungen sind:

1. Gibt es für ein Ziel keine anwendbaren Operatoren, so wird dies durch die Blattmarkierung 'unachievable' dokumentiert.[3]

2. Teilzielrekursion
   Die PSGs beinhalten auch die Teilzielzerlegung. Deshalb können Teilzielrekursionen völlig analog

[2] Die Argumente der Axiome sind natürlich Variablen. Der Einsatz der Axiome läßt sich aber in der Aussagenlogik leichter erklären.

[3] In der CAPlan-Version der PSGs werden auch alle Phantomvorbedingungen so markiert, da auf sie ebenfalls keine Operatoren angewendet werden dürfen.

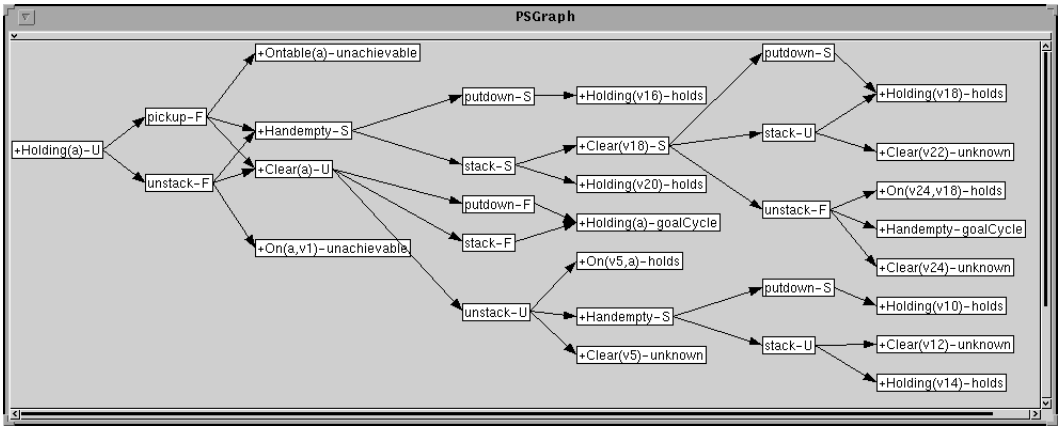

Abbildung 4.3: Ein PSG der Blocksworld

zum Planungsvorgang erkannt werden. Solche Blattknoten werden mit 'goalCycle' markiert. Um die PSGs kompakt zu halten wird allerdings das herkömmliche, starke Rekursionskriterium benutzt.

3. Potentielle Teilzielrekursion
   Die PSGs enthalten viele Variablen. Tritt eine Teilzielrekursion daher nur unter bestimmten Variablenbindungen auf, so wird die Expansion vorsichtshalber abgebrochen und der Knoten mit 'unknown' markiert.

4. Für einige Zielknoten läßt sich mit Hilfe der Axiome herleiten, daß sie gültig sein müssen, wenn die Expansion des Suchraumes bis zu dieser Stelle vorangetrieben worden ist.[4] Da diese Knoten dann vom Planer nicht weiter expandiert würden, beendet der PSG-Mechanismus ebenfalls die Expansion und markiert den Knoten mit 'holds' (Beispiel s. 4.4

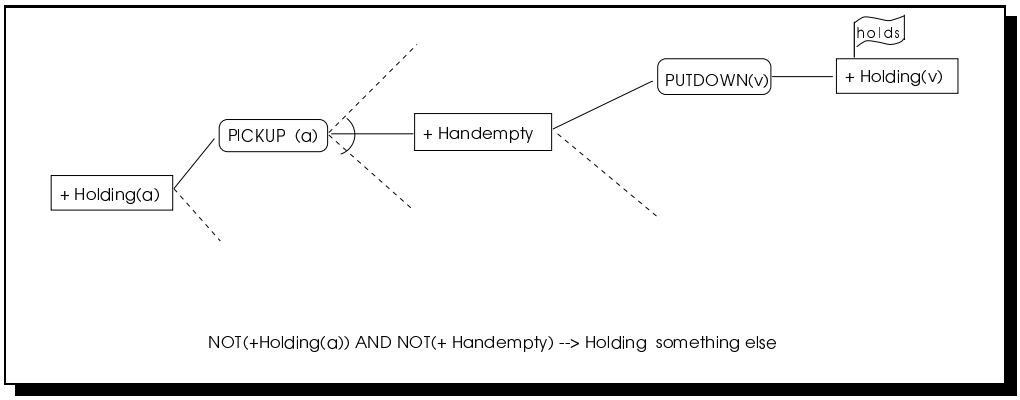

Abbildung 4.4: Ein Ausschnitt aus obigem PSG, der die Anwendung der Axiome demonstriert

Bernd Völker [Völ94] hat die PSGs in der oben beschriebenen Form für CAPlan implementiert und mit einer Graphikoberfläche versehen. Später hat sich gezeigt, daß eine Integration von Constraints in die PSGs unumgänglich wurde. Dies liegt zum einen daran, daß die Operator-Constraints in CAPlan meist wichtige Bedeutungsträger sind und bei der Formulierung von Fehlschlagsbedingungen (vgl. Abschnitt 4.3.3.1) nicht vernachlässigbar sind. Zudem mußte auch der Erweiterung der Axiome um Anwendbarkeits-Constraints Rechnung getragen werden.

[4] Solch ein Knoten wird nur erreicht, wenn alle Knoten auf dem Pfad von der Wurzel zu diesem Knoten nicht im Zustand gültig waren. Ist die Menge der Literale auf diesem Pfad eine Obermenge einer Axiomdisjunktion, so folgt, daß nicht alle diese Literale ungültig sein können– also muß das letzte wahr sein.

Propagierung von Operator-Constraints. Die Propagierung von Operator-Constraints wurde so realisiert, daß jeder PSG Knoten um einen Eintrag erweitert wurde, der die Menge aller eigenen und ererbten Constraints umfaßt, die die eigenen Argumente betreffen. Operatoren, deren Constraints nicht mit den ererbten Constraints zu vereinbaren sind, werden als Fehlschlag markiert und nicht weiter expandiert.

Verwendung von Axiomen mit Constraints. Axiome werden u.a. dazu verwendet, die Expansion der PSGs zu beschränken. Bei der Planung kann die Anwendbarkeitsbedingung eines Axiomes direkt abgetestet werden, weil z.B. der Typ eines Objektes bekannt ist. Will man aber bedingte Axiome auf PSGs anwenden, so steht man vor dem Problem, daß die PSGs für generische Objekte ohne Typangaben aufgebaut werden, und somit bestimmte Anwendbarkeitsbedingungen nicht abtestbar sind. Auf der anderen Seite erhält man im Laufe der Expansion durch die Propagierung von Operator-Constraints dennoch einige Typ- und Bindungs-Constraints. Für den Abbruch der Expansion wurden daher nur solche Axiome verwendet, die keine Anwendungs-Constraints besitzen oder deren Anwendungsbedingungen durch die Operator-Constraints erfüllt werden.

4.1.2.3 Verallgemeinerter Zustand

Eine Reihe von Verfahren stellt Bedingungen an den Zustand auf. Während das Abtesten solcher Bedingungen für zustandsbasierte Planer ganz einfach ist, steht man bei partiellordnenden Planern vor dem Problem, daß sich dort gar keinen Zustand bestimmen läßt, weil sich dieser für jede Linearisierung des partiell geordneten Plans ändert.

Chapman [Cha87] hat mit dem MTC einen Mechanismus geschaffen, mit dem die Menge der notwendigerweise gültigen und der möglicherweise gültigen Literale bestimmt werden kann. Die Menge der notwendigerweise gültigen Literale besteht vereinfacht gesagt aus allen noch nicht verbrauchten Effekte derjenigen Schritte, die vor der aktuellen Position angeordnet sind. Bei der Operatorauswahl berechnet man aber den Zustand für einen noch nicht erzeugten Operator, für den trivialerweise gilt, daß außer dem Anfangszustand kein anderer Schritt vor ihm angeordnet ist. Die Menge der notwendigerweise gültigen Literale beschränkt sich somit immer auf den Anfangszustand und ist daher zu restriktiv. Wir setzten deshalb einen verallgemeinerten Zustand ein:

Definition 3 (Verallgemeinerter Zustand)
Sei $s \in steps(Plan)$. $genState(s) = \bigcup effects(s') \; \forall \; s' \; such \; that \; s' \preceq s$.

Es wäre wünschenswert, diese Menge noch weiter einschränken zu können auf die Menge der wirklich benutzbaren Effekte, d.h. auf diejenigen Effekte, die bedrohungsfrei phantomisiert werden können. Eine solche Einschränkung ist aber nicht möglich, da die Benutzbarkeit davon abhängig ist, ob die derzeitigen Benutzer eines Effektes diesen auch zerstören und davon, ob der Effekt im noch zu expandierenden Teilbaum ebenfalls verbraucht oder nur benutzt würde. Man kann sich dies leicht klarmachen, indem man einige Situationen durchspielt und dabei im Hinterkopf behält, daß außer dem Anfangszustand kein anderer Schritt vor dem auszuwählenden Operator liegt.

Festzuhalten bleibt: Dadurch daß der verallgemeinerte Zustand eine Obermenge der tatsächlich benutzbaren Literale ist, werden Verfahren, die die Ungültigkeit von Literalen testen, genauso korrekt sein wie für totalordnende Planer, aber nur seltener angewandt werden können. Umgekehrt werden Regeln, die Bedingungen an die Gültigkeit von Literalen stellen, häufiger anwendbar sein, dafür aber weniger korrekte Entscheidungen treffen.

4.1.2.4 PMU – Phantomization Management Unit

Dieser Hilfsstruktur liegt eine alternative Sicht des Planungsproblems zugrunde: Demnach läßt sich das Planungsproblem für Causal Link Planer in zwei Hauptaufgaben aufspalten:

1. Dem Finden eines Plangerüstes von Operatoren, das das Problem löst
2. Konsistente Verteilung der Phantomisierungsmöglichkeiten auf die noch offenen Ziele

Diese Problemaufteilung ergibt sich nur für Planer, die das Ausnützen von Seiteneffekten explizit repräsentieren. Zu den Hauptschwierigkeiten der Planung bei obiger Sichtweise gehört die Entscheidung, welche Ziele durch Domänenoperatoren geplant werden müssen und welche phantomisiert werden können. Mit der PMU wurde nun eine Struktur entwickelt, die die Lösung dieser und einiger weiterer Problemstellungen effizient zu beantworten hilft.

Eröffnung neuer Blickwinkel. Die PMU eröffnet ganz neue Möglichkeiten, das Planungsproblems zu betrachten:

- Die PMU stellt die Transformation eines Teilaspektes der Planung in ein Constraint Satisfaction Problem dar, indem offene Ziele Variablen entsprechen und passende Effekte den Werten. Diese Umformulierung erlaubt die Übertragung von Lösungskonzepte aus dem Constraint Satisfaction Bereich. Allerdings besteht die Schwierigkeit, daß im Unterschied zu CSPs weder die Menge der Variablen noch die Wertebereiche feststehen, weil im Verlaufe der Planung ständig neue Ziele und Phantomisierungsmöglichkeiten hinzukommen.
- Die Formulierung als Constraint Satisfaction Problem erlaubt zwei bisher vernachlässigte Sichtweisen: Zum einen die Auffassung von Phantomisierungsmöglichkeiten als verbrauchbare Ressourcen, die Benutzern zugewiesen werden und zum anderen die Einbeziehung der Erzeugersicht in den Entscheidungsprozeß der Planung. Mit 'Erzeugersicht' ist gemeint, daß auch die Anzahl der phantomisierungswilligen Ziele eines Operatoreffektes darauf Einfluß hat, wie kritisch eine Entscheidung ist.
- Zudem ermöglicht die Verwaltung der Phantomisierungsmöglichkeiten durch die PMU eine *kontinuierliche* Berechnung der Konfliktmenge eines Zieles. Dadurch wird auch die Verwendung teurerer Ausschlußkriterien sinnvoll, was wiederum dazu führt, daß die Menge der tatsächlich benutzbaren Operatoren besser approximiert werden kann.

Realisierung. Die Grundstruktur der PMU ist eine Matrix, die die Phantomisierungsmöglichkeiten enthält und durch die potentiellen Benutzer und Erzeuger indiziert ist. Basiszellen dieser Matrix enthalten außer einem Phantomisierungsoperator zusätzlicher Informationen, die die Verwaltung erleichtern. Dazu gehören etwa die Information, ob der benutzende Schritt die Phantomisierungmöglichkeit auch zerstört, symbolisch kodierte Ausschlußgründe und eine JTMS-Knotenstruktur, die Informationen über den Ausschluß oder die Zulässigkeit einer Phantomisierungsmöglichkeit verwaltet.

Um effizientes Aktualisieren und Beantworten von Anfragen zu ermöglichen, stellt die PMU eine Reihe schneller[5] Zugriffsmöglichkeiten zur Verfügung. Dazu gehören:

1. Menge der Phantomisierungsmöglichkeiten eines Zieles.
2. Menge der Phantomisierungsmöglichkeiten eines Effektes.
3. Menge der Phantomisierungsmöglichkeiten, die eine bestimmte Variable instantiieren.
   Diese Zugriffsmöglichkeit wird für ein effizientes Aktualisieren der PMU benötigt, da durch Einführung von Bindungs-Constraints Phantomisierungen ausgeschlossen werden.
4. Menge der Schritte, die einen bestimmten Effekt erzeugen.
5. Menge der Effekte eines Prädikates.
6. Menge der Phantomisierungen, die eine gegebene Schrittanordnung realisieren.
   Diese Zugriffsmöglichkeit wird benötigt, um bei Einführung eines Ordnungs-Constraints ordnungsinkonsistente Phantomisierungsmöglichkeiten schnell ausschließen zu können.

Aufbauend auf diesen Zugriffsmethoden werden einige Anfragemöglichkeiten zur Verfügung gestellt, die von Kontrollmethoden genutzt werden können:

[5] Diese Zugriffsfunktionen sind durch Hashtabellen realisiert.

1. Menge zulässiger Phantomisierungsmöglichkeiten für ein Ziel.

2. Menge aller möglichen Benutzer eines Effektes.

3. Menge der noch zu planenden Ziele, die nicht phantomisiert werden können.

4. Menge der Ziele, die nicht (mehr) durch Domänenoperatoren erreicht werden können und phantomisiert werden müssen.

Die Verwaltung der Phantomisierungen gliedert sich in zwei Hauptaufgaben, nämlich dem Einfügen neuer Elemente in die PMU und dem Ausschluß von Phantomisierungsmöglichkeiten anhand bestimmter Kriterien.

Einfügen neuer Einträge
Nachdem ein neuer Schritt in dem Plan eingefügt worden ist, müssen für jede Vorbedingung und jeden Effekt des Operators neue Spalten und Zeilen in der Matrix angelegt und ausgefüllt werden. Dadurch werden diese neuen Vorbedingungen und Effekte auch gleichzeitig als potentielle Benutzer bzw. Erzeuger in die schon vorhandenen Listen eingetragen. Ferner muß für das durch den neuen Schritt erplante Ziel vermerkt werden, daß keine Phantomisierungsmöglichkeit mehr benötigt wird.

Ausschluß von Phantomisierungsmöglichkeiten
Zum Ausschluß von Phantomisierungsmöglichkeiten führen folgende Kriterien:

1. Inkonsistente Bindungsconstraints
   Durch Domänenoperatoren, Separation von Threats und Phantomisierungen werden Bindungs-Constraints eingeführt, die andere Phantomisierungen ausschließen. Auf die zu überprüfenden Phantomisierungsmöglichkeiten wird über eine Hashtabelle zugegriffen, die einer Variablen alle sie instantiierende Phantomisierungsmöglichkeiten zuordnet.

2. Inkonsistente Ordnungs-Constraints
   Durch Threatauflösungsoperatoren und Phantomisierungen werden Ordnungs-Constraints eingeführt, die ebenfalls Phantomisierungsmöglichkeiten ausschließen können. Die Menge der hiervon betroffenen Phantomisierungsmöglichkeiten wird bestimmt, indem man zunächst die Menge der durch die neue Anordnung unmöglich gewordenen Ordnungsconstraints berechnet und dann aus einer weiteren Hashtabelle der PMU diejenigen Phantomisierungsmöglichkeiten herausliest, die diese unmöglich gewordenen Anordnungen benötigen.

3. Verbrauchte Phantomisierungen
   Wie eingangs schon erwähnt, betont die PMU u.a. eine Ressourcensicht auf die Phantomisierungsmöglichkeiten. Von besonderer Bedeutung ist dabei, daß Ressourcen verbrauchbar sind und verbrauchte Ressourcen nicht noch einmal verwendet werden können.
   Auf Phantomisierungsmöglichkeiten übertragen bedeutet das, daß ein Effekt nicht durch zwei zerstörende Operatoren genutzt werden kann. In SNLP wird die Ressourcensicht an sich nicht benötigt, weil dieser Sachverhalt auch durch Threats ausgedrückt wird: verbrauchende Schritte haben die Negation der Vorbedingung unter ihren Effekten und bedrohen daher die Causal Links mitbenutzender Schritte. Sind beide Schritte Verbraucher, so bedrohen sie sich gegenseitig und der Konflikt ist unauflösbar. Einen Spezialfall stellen Benutzer dar, die eine Vorbedingung nicht durch ihre Effekte zerstören. Für solche Phantomisierungsmöglichkeiten gilt, daß durch sie keine anderen Phantomisierungsmöglichkeiten ausgeschlossen werden und daß sie selbst nur dadurch ausgeschlossen werden, daß sie notwendigerweise hinter verbrauchenden Schritten liegen.

4. Inkonsistente Phantomisierungen
   Bei der Erzeugung einer neuen PMU-Zelle wird außerdem überprüft, ob diese Phantomisierung unauflösbare Threats zur Folge hätte. Dies wird in einer Weise getan, die in Abschnitt 4.3.3.3 noch ausführlicher besprochen wird. Als inkonsistent werden die Causal Links aufgefaßt, die notwendigerweise parallel zu Schritten liegen, deren Vorbedingungen oder Effekte das durch den Causal Link geschützte Literal notwendigerweise ausschließen. Dies wird mit Hilfe der Axiome festgestellt.

Die oben beschriebenen Auschlußkriterien werden durch ein System von JTMS-Knoten verwaltet, die eine Erweiterung des REDUX-Systems darstellen und von diesem mitverwaltet werden. Die Erweiterung wurde so gestaltet, daß die PMU-Strukturen zwar auf REDUX-Knoten aufbauen können, aber nicht auf das REDUX-System zurückwirken. Die Grundstruktur der Ausschlußrechtfertigungen zeigt Abb. 4.5:

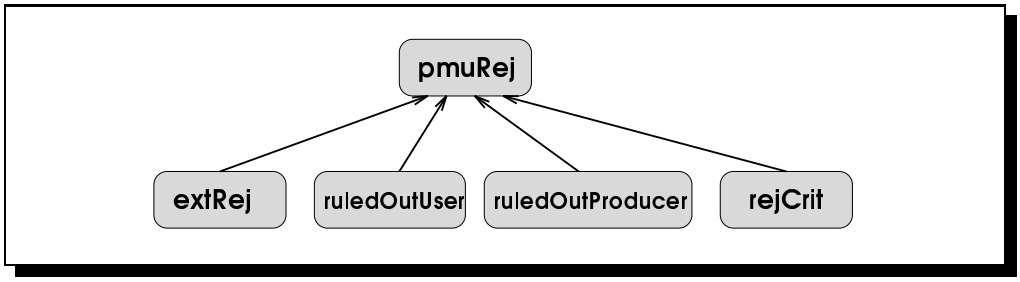

Abbildung 4.5: Knotenstruktur der Ausschlußrechtfertigungen

Der Knoten *pmuRej* symbolisiert den Ausschluß einer Phantomisierungsmöglichkeit. Der Knoten *extRej* steht für externe Ausschlüsse und wird im folgenden noch näher erläutert Die Knoten *ruledOutUser* und *ruledOutProducer* symbolisieren einen Ausschluß des Benutzers bzw. des Erzeugers einer Phantomisierung aufgrund von verschiedenen Kriterien.[6] Schließlich symbolisiert *rejCrit* Ausschlüsse, die darauf basieren, daß ein Causal Link zu unauflösbaren Threats führt. Die Begründungsstrukturen für einen externen Ausschluß sind in Abb. 4.6 zu sehen. Dunkel unterlegte Knoten stehen dabei für PMU-Knoten, hell unterlegte für REDUX-Knoten. Der Knoten *csImp*(constraints impossible) symbolisiert dabei die Ungültigkeit aufgrund widersprüchlicher Constraints.

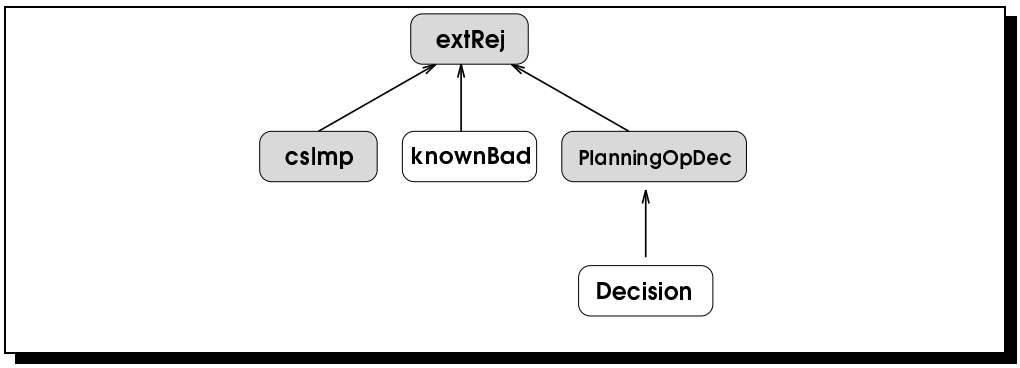

Abbildung 4.6: Knotenstruktur der externer Ausschlußrechtfertigungen

Ähnliche Arbeiten. Verfahren zur Verwaltung von Ressourcen sind insbesondere aus dem Bereich des Operation Research und des Scheduling bekannt, wobei jedoch meistens Ressourcen gesondert repräsentiert sind und nicht wie hier vorgestellt, Phantomisierungsmöglichkeiten generell als Ressourcen betrachtet werden. O-Plan2 [Dra94, TDK94] ist ein Beispiel eines Planungssystems, das ein separates Modul zur Verwaltung von Ressourcen, der sogenannte RUM(Resource Utilisation Manager), besitzt. In O-Plan2 werden Ressourcen ebenfalls danach unterschieden, ob sie verbraucht oder nur benutzt werden. Zusätzlich werden aber z.B. auch Aspekte wie die parallele Benutzbarkeit einer Ressource modelliert.

[6] Der Ausschluß des Benutzers und des Erzeugers mußte durch zwei separate Knotentypen realisiert werden, um Zirkelschlüsse zu vermeiden.

4.1.3 Erläuterung des Klassifikationsschemas

In diesem Abschnitt soll das Klassifikationsschema erläutert werden, das im nachfolgenden Abschnitt jeweils der Beschreibung der einzelnen Lokalheuristiken folgt. Das Klassifikationsschema umfaßt die folgenden Einträge:

Heuristik. Dieser Eintrag bezieht sich auf die Grundheuristik, die der beschriebenen Lokalheuristik unterliegt. Die Grundheuristiken wurden in Abschnitt 4.1.1 eingeführt.

Wirkung. Hier wird die beabsichtigte Wirkung dieser Heuristik beschrieben.

Transformation. Die Anwendung einiger Kontrollmethoden (insbesondere Instanzen der Look-Ahead Heuristik) bringt Kosten mit sich. Im Regelfall sollte der Gewinn die Kosten überschreiten. Man kann dieses Utility-Problem [Min88] auch unter dem Aspekt betrachten, daß gewisse Teile des Planungsproblem in ein anderes, möglicherweise billiger zu lösendes Problem transformiert werden - wie etwa dem Matchen von Zustandsbedingungen im Falle von EBL-Regeln. Die offensichtlichsten Transformationsmöglichkeiten sind die folgenden:

1. Das Planungsproblem läßt sich in ein Matchproblem transformieren, indem etwa die Operatorknoten in einem PSG vernachlässigt werden und der entstehende UND-ODER-Baum von Zielknoten als logischer Ausdruck aufgefaßt wird. Eine solche Transformation von Teilbäumen findet sich oft in EBL-Regelbedingungen, ist aber offensichtlich nur dann von Vorteil, wenn der entsprechende logische Ausdruck stark vereinfacht werden kann.

2. Viele Planungsprobleme lassen sich auch als Constraint Satisfaction Problem formulieren. Das sehr leistungsfähige O-Plan System ist ein Planungssystem, das Pläne gänzlich als Constraintmenge repräsentiert [Tat95]. Da für Constraint Satisfaction Probleme viele Lösungsansätze bekannt sind, kann solch eine Transformation durchaus vorteilhaft sein.

Korrektheitsbedingungen. Manche der Methoden sind unter bestimmten Bedingungen korrekt. Die Verwendung dieser Methoden basiert also implizit auf der Annahme, daß diese Bedingungen in der aktuellen Situation gültig sind. Die Kenntnis dieser impliziten Annahmen erlaubt daher eine informiertere Entscheidung für oder gegen den Einsatz solcher Lokalheuristiken. Der Eintrag 'absolut korrekt' in dieser Spalte bedeutet, daß die Korrektheit dieser Methode nicht an implizite Annahmen gebunden ist; der Eintrag '-' bedeutet, daß überhaupt keine Bedingungen angegeben werden können, unter denen die Methode korrekte Ergebnisse liefert. Diese Methoden drücken meist nur Wahrscheinlichkeiten (im umgangssprachlichen Sinne) aus und gehören zu den schwächsten Kontrollverfahren.

Hilfsmittel Hier sind die Hilfsmittel aufgeführt, die die Methode benötigt. Dies ist deshalb relevant, weil diese Hilfsmittel für manche Domänen nicht zur Verfügung stehen oder ihre Berechnung zu komplex wird. Beispiel: viele Verfahren beruhen auf dem Einsatz von Axiomen. Viele reelle Domänen erlauben jedoch gar keine oder nur eine schwache Axiomatisierung. Für diese Domänen ist eine Verwendung von axiombasierten Verfahren nicht sinnvoll.

Komplexitätsfaktoren. Die Komplexität und damit die Kosten der einzelnen Verfahren hängen meist sehr stark von der jeweiligen Domäne ab. In dieser Spalte werden daher all diejenigen Faktoren genannt, für die eine Zunahme zu einer Erhöhung der Kosten führt. Dies soll ebenfalls die Entscheidung für oder gegen den Einsatz einer Methode erleichtern.

Ergebnistyp. Die Verfahren liefern Ergebnisse, die sich einer der folgenden Typen zuordnen lassen:

1. Bisektion
   Das Verfahren teilt die Alternativen in zwei Klassen ein: die 'Guten' und die 'Schlechten'. Ein Beispiel wäre eine Methode, die inkonsistente Alternativen aus der Konfliktmenge entfernt.

2. Multisektion
   Das Verfahren teilt die Alternativen entsprechend einem Kriterium in mehrere Klassen ein, auf denen eine Totalordnung definiert ist. Im Rahmen dieser Arbeit wird dabei die 'beste' Klasse immer als Minimum definiert sein; dies wurde in Analogie zur Mathematik und Physik gemacht, wo meist (Energie-) Minima gesucht sind.
   Beispiel für eine Multisektion ist die Ordnung von Zielen nach der Anzahl der zu ihrer Lösung zur Verfügung stehenden Operatoren. Die Multisektion wird auch als schwache Ordnung bezeichnet (vgl. [RW90]).

3. Relative Ordnung
   Bei dieser Art von Verfahren liefert das Verfahren eine Partialordnung auf der Menge der Alternativen. Aufbauend auf dieser Partialordnung lassen sich nun durch Schichtenbildung Klassen definieren. Schichten sind dabei Äquivalenzklassen bezüglich des Kantenabstandes vom Minimum (bzw. den Minima).
   Hier ist zu beachten, daß die Klassenbildung über die Partialordnung definiert ist und sich bei

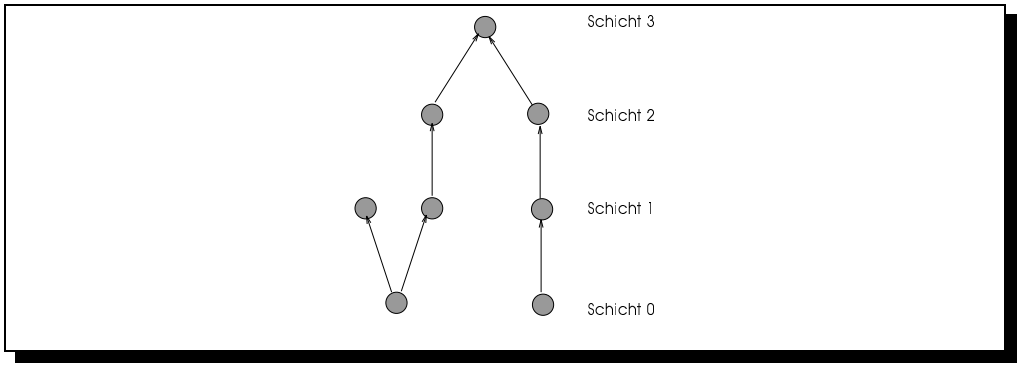

Abbildung 4.7: Schichtenbildung bei gegebener Partialordnung

   jeder Änderung der Partialordnung (z.B. durch Erweiterung oder Verkleinerung der Menge der Alternativen) mitändert.

Der Ergebnistyp spielt bei der Verknüpfung mehrerer Kontrollmethoden eine Rolle und wird daher in Kapitel 5 nochmals gesondert aufgegriffen werden.

4.2 Kontrollverfahren: Änderungen der Feinarchitektur

Nachdem wir nun mit den Basisheuristiken, den Hilfsmitteln und dem Klassifikationsschema vertraut sind, wenden wir uns den eigentlichen Kontrollverfahren zu. In diesem Abschnitt sollen zunächst die vorgeschlagenen Feinarchitekturänderungen vorgestellt werden. Entsprechend der Anforderung, möglichst wenig am Grundalgorithmus zu ändern, wurde in den meisten Fällen eine tatsächliche Umdefinierung vermieden und stattdessen die Änderung auf lokale Auswahlkriterien abgebildet, die im nachfolgenden Abschnitt beschrieben werden. Bei den tatsächlich vorgenommenen Architekturänderungen wurde besondere Sorgfalt darauf verwandt, daß die Korrektheit und Vollständigkeit durch diese Änderungen nicht beeinträchtigt wird.

4.2.1 Spezialisierung des Threatbegriffes

Die direkte Umdefinierung eines Threats wäre nicht ohne einschneidende Änderungen des Algorithmus möglich gewesen. Sie wurde daher auf die Vermeidung separierbarer Protektionsziele abgebildet (siehe

Abschnitt 4.3.2.2).

4.2.2 Verallgemeinerung des Threatbegriffes – Notwendige Schrittanordnungen

Die notwendigen Schrittanordnungen basieren auf zwei Grundideen: Die erste Idee ist, mit Hilfe der Axiome verallgemeinerte Threats zu bestimmen. Die zweite Idee ist, für verallgemeinerte Threats, für die nur eine Auflösungsmöglichkeit besteht, den entsprechenden Ordnungs-Constraint direkt in den Plan einzufügen. Beide Ideen sollen im folgenden etwas näher erläutert werden:

4.2.2.1 Verallgemeinerung des Threatbegriffs

Gewöhnliche Threats sind wie folgt definiert:

Threat(C,$P \xrightarrow{g} U$) gwd. $\exists$ e $\in$ eff(C): (e $\approx$ g)$\vee$ (e $\approx \neg$ g)

Die Verallgemeinerung besteht nun lediglich darin, die Unifizierbarkeit mit einem Literal oder dessen Negation durch die entsprechenden Axiomfolgerungen zu ersetzen (siehe Abschnitt 4.1.2.1):

Threat(C,$P \xrightarrow{g} U$) gwd. $\exists$ e $\in$ eff(C): (e $\approx_{ax}$ g)$\vee$ (e $\approx_{ax} \neg$ g)

4.2.2.2 Einführung von Ordnungen für eindeutig aufzulösenden Threats

Die Beschränkung auf eindeutig auflösbare Threats ist zum einen dadurch motiviert, daß dieses Wissen sofort im Plan verwendet werden kann und zum anderen dadurch, daß die Berechnung verallgemeinerter Threats teurer ist und deshalb schon allein aus Effizienzgründen eine Einschränkung der zu testenden Kandidaten notwendig wird. Wie bei gewöhnlichen Threats unterscheiden wir bei Einführung eines Schritts die Gruppe der Threats, bei denen der neue Schritt alte Causal Links bedroht von der Gruppe der Threats, bei denen der neue Causal Link durch alte Schritte bedroht wird.

Bedrohungen durch den neuen Schritt. Die grundsätzlichen Überlegungen lassen sich anhand von Abb. 4.8 rasch verdeutlichen: In Situation a) sei der Schritt NEW gerade neu in den Plan eingefügt

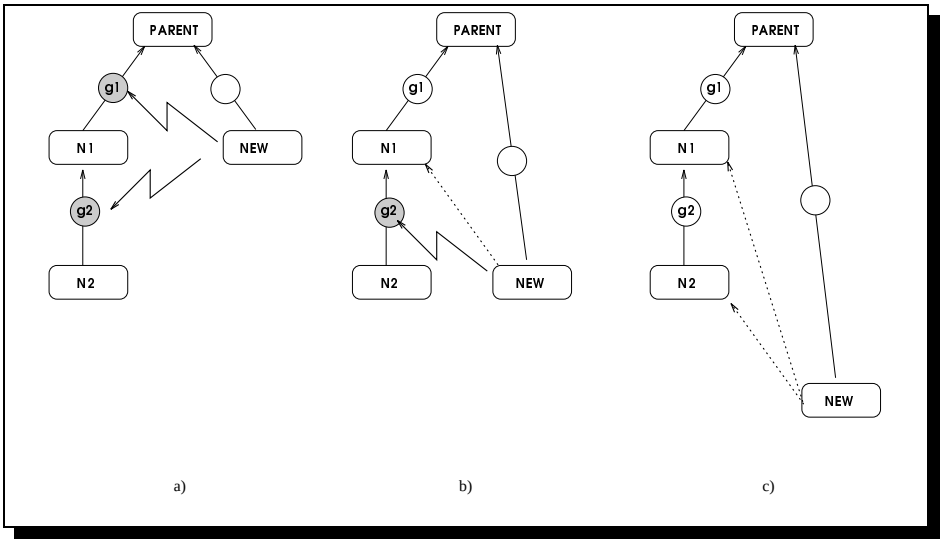

Abbildung 4.8: Einsinken eines bedrohenden Schrittes

worden. Dieser Schritt habe nun Effekte, die mittels der Axiome den Causal Link $N1 \xrightarrow{g1} PARENT$ notwendig, d.h. unabhängig von Variablenbindungen bedrohen. Als Threatauflösungsmöglichkeiten

bleiben dann Promotion und Demotion, wobei die Promotion ebenfalls ausgeschlossen ist, weil NEW schon hinter PARENT angeordnet ist. NEW muß daher notwendigerweise hinter N1 angeordnet werden. Wie aus Situation b) und c) ersichtlich, setzt sich dieser Anordnungsprozeß weiter fort, bis NEW soweit eingesickert ist, daß kein Causal Link mehr durch ihn bedroht wird.
Diese Überlegungen kann man nun nicht nur für Geschwisterzweige machen, sondern auch für Kinder älterer Vorfahren, weil der neue Knoten vor allen seinen Vorfahren angeordnet ist und daher in allen diesen Fällen die Promotion ausgeschlossen ist. Dies wird in Abb. 4.9 nochmals verdeutlicht.

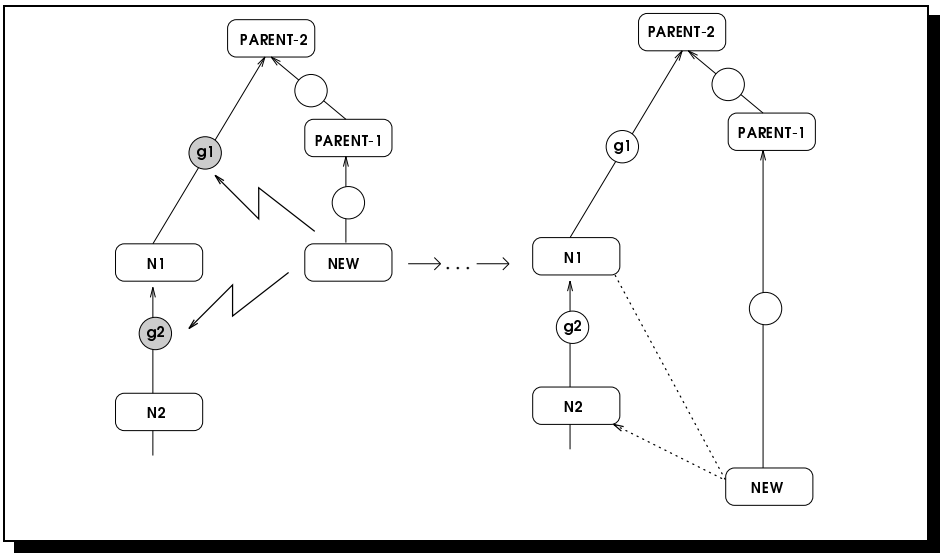

Abbildung 4.9: Einsinken eines bedrohenden Schrittes bei anderem Verwandtschaftsverhältnis

Wie lassen sich diese Anordnungen nun effektiv berechnen? Wir stellen zunächst einmal den Algorithmus vor und zeigen dann, warum dieser intuitive Algorithmus korrekt ist.
Der Algorithmus ist in zwei Phasen eingeteilt; in der ersten bestimmt man auf allen Vorfahrebenen diejenigen Geschwister, deren einführender Causal Link vom neuen Schritt bedroht wird. In einem zweiten Schritt werden dann ausgehend von diesen bedrohten Geschwistern die Menge all der Schritte bestimmt, die dadurch erreichbar sind, daß man bedrohte Links entlangläuft und zwar in entgegengesetzter Richtung. All diese Schritte werden dann vor dem neuen Schritt angeordnet. Sowohl die erste Phase als auch die zweite Phase lassen sich durch rekursive Methoden realisieren.

Bei Betrachtung des Algorithmus ergeben sich unmittelbar die folgenden Fragen:

1. Durch die Anordnung erhält der neue Schritt auch neue Vorfahren. Muß für diese Vorfahren der Algorithmus erneut aufgerufen werden?
   Es genügt, den Algorithmus einmal aufzurufen. Den Grund verdeutlicht man sich wiederum am besten an einer Schemazeichnung(Abb. 4.10). Eine Neuberechnung wäre ja dann notwendig, wenn Geschwister bezüglich des neuen Vorfahren N2 wieder bedroht wären, wie in der Zeichnung durch dunklere Färbung der Causal Links von N3 und N4 angedeutet ist. Wie ebenfalls aus der Zeichnung ersichtlich ist, sind diese Knoten bereits in einer Kette bedrohter Schritte enthalten, die in PARENT-2 beginnt, und die deshalb im ersten Durchlauf des Algorithmus schon berücksichtigt werden.

2. Warum wird nicht nur die jeweils letzten bedrohten Causal Links in solch einer Kette bedrohter Links bestimmt? Eine Anordnung dieses Kettenendes würde ja die vorherigen implizieren.
   Prinzipiell würde diese Anordnung genügen. Aus Gründen der Robustheit muß aber sichergestellt sein, daß die implizierten Ordnungen auch im Zuge des Backtracking erhalten bleiben. Würde nämlich der letzte Schritt einer Kette bedrohter Schritte zurückgezogen, dann wäre bei der sparsamen Variante die Information über die Anordnung des neuen Schrittes bezüglich der darüberliegenden Schritte verloren.

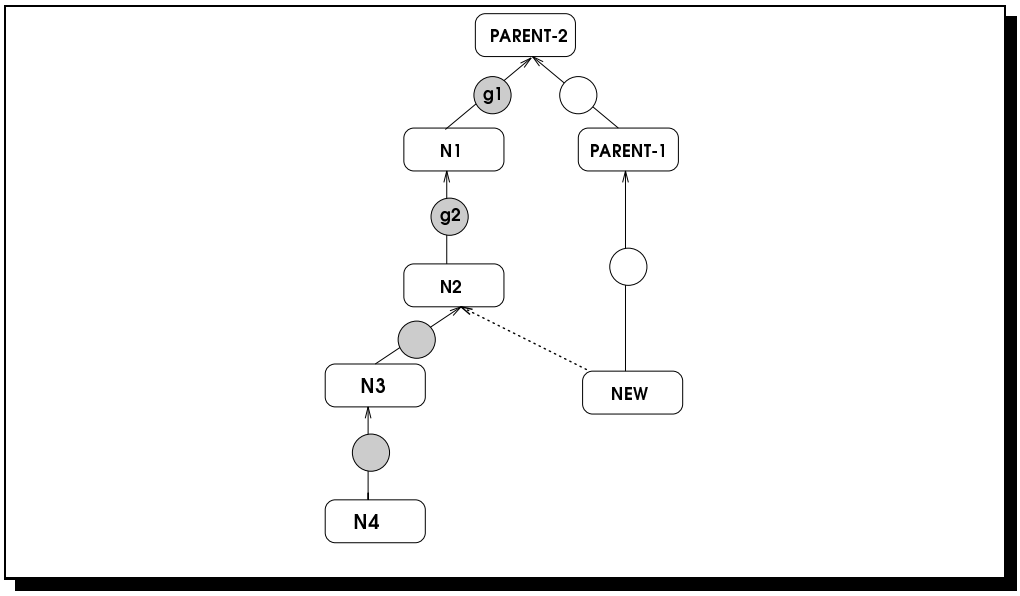

Abbildung 4.10: Eine Iteration des Algorithmus genügt.

Bedrohungen des neuen Causal Link. Wir analysieren die Situation wieder anhand einer Schemazeichnung (Abb. 4.11). Sei also der Causal Link des neu eingeführten Schrittes NEW notwendig

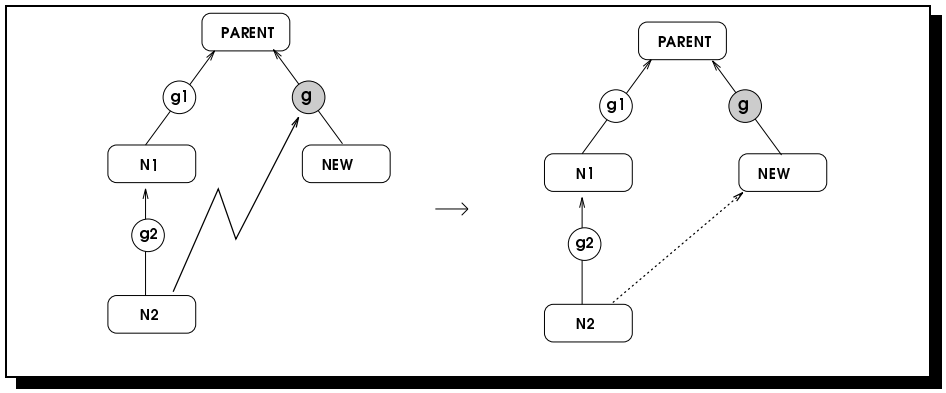

Abbildung 4.11: Bedrohung des neuen Causal Link

bedroht durch einen parallel gelegenen Schritt N2. Die Anordnung von N2 hinter PARENT durch die Teilzielhierarchie schließt die Promotion als Threatauflösungsmöglichkeit aus. Als einzige Möglichkeit bleibt die Demotion, wie in der Schemazeichnung rechts gezeigt. Auch hier müssen wieder ähnliche Fragen geklärt werden wie im obigen Fall.

Behandlung von mehreren Threats durch Knoten desselben Astes.
Andere bedrohende Knoten im selben Ast müssten ebenfalls hinter NEW angeordnet werden. Es gilt jedoch, daß die Anordnung des jeweils höchstgelegenen Knotens die Anordnung all seiner Nachfolger impliziert. Im Gegensatz zum vorherigen Fall bleibt diese Implikation auch bei Backtracking erhalten, weil der höchstgelegene Knoten bei chronologischem Backtracking als letzter zurückgenommen wird, bzw. bei Rückzug dieses höchstgelegenen Knoten durch den Benutzer die abhängigen Knoten ebenfalls zurückgenommen werden.[7] Es läßt sich also festhalten, daß es in diesem Fall genügt, für jeden Geschwisterzweig den höchstgelegenen bedrohenden Knoten zu bestimmen.

[7] Bei der Berechnung der bedrohenden Schritte werden nur die einführenden Causal Links für die Bestimmung eines 'Verwandtschaftsverhältnisses' berücksichtigt. Die Existenz eines Knotens ist aber mit der Existenz seines einführenden Causal Links untrennbar verbunden.

Verallgemeinerung auf Geschwisterzweige bezüglich anderer Vorfahren.
Im Gegensatz zum vorherigen Fall dürfen hier nur echte Geschwisterzweige untersucht werden, also Alternativzweige des direkten Vorfahren. Dies wird aus Abb. 4.12 ersichtlich: In dieser Situation ist

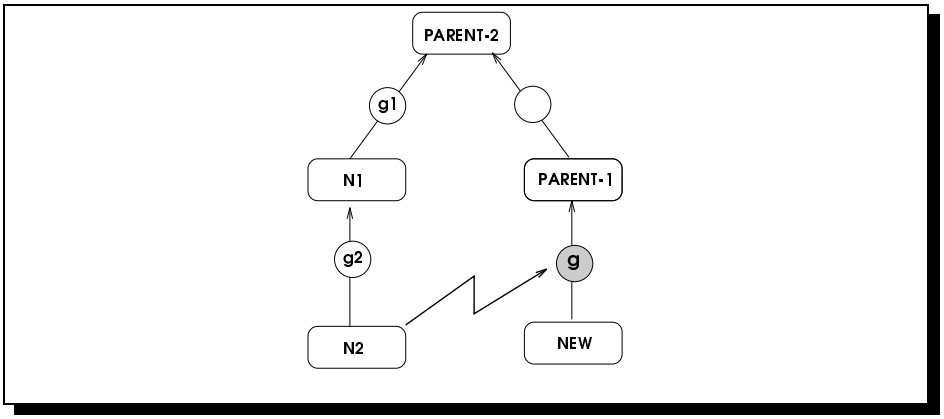

Abbildung 4.12: Keine Verallgemeinerung auf andere Verwandtschaftsverhältnisse

N2 nur bezüglich PARENT-2, nicht jedoch bezüglich PARENT-1 geordnet. Die Promotion N2 $\prec$ PARENT-1 ist daher nicht ausgeschlossen und die Threatauflösung nicht mehr eindeutig.

Kombination. Die Kombination der zwei Algorithmen ist für korrekte Operatoren problemlos, da sich in diesem Fall die erzeugten Ordnungen nicht widersprechen. Problematisch ist die Situation allerdings, wenn der neue Schritt inherent zu unauflösbaren Threats führt, die aber eventuell. erst später offensichtlich werden. Durch den vorausschauenden Effekt der Axiome werden in diesem Falle häufig inkonsistente Ordnungs-Constraintmengen erzeugt, die aber nicht in den Plan eingefügt werden dürfen. Eine Möglichkeit wäre nun, diese vorausschauende Inkonsistenzerkennung zur Auswahl von Operatoren zu nutzen. Die Praxis hat jedoch gezeigt, daß die Berechnung der notwendigen Schrittanordnungen zu teuer ist, um sie für alle Operatoren einer Konfliktmenge durchzuführen.[8] Es wurde deshalb eine Variante realisiert, bei der nach Einführung eines Schrittes die Menge der notwendigen Ordnungen berechnet wird. Diese Menge wird aber nur dann dem Plan hinzugefügt, wenn dadurch keine Inkonsistenzen entstehen. Man nutzt dadurch zwar das Wissen um ein Fehlschlagen eines Operators nicht aus, profitiert aber im Falle korrekter Operatoren von der zusäztlichen Linearisierung des Planes, die viele falsche Threatauflösungsmöglichkeiten und Phantomisierungsmöglichkeiten ausschließt und damit viele Operatorentscheidungen sicherer macht.

Verträglichkeit mit Backtracking. Es ist schon gezeigt worden, daß die Ordnungen mit dem Rückzug von Schritten verträglich sind. Hier soll nun noch gezeigt werden, wie im Falle der Wiederauswahl eines Knotens vorgegangen wird. Bei der Wiederauswahl eines Knotens hat sich häufig die Ordnungsstruktur des Planes geändert und meist sind einige der bei der ersten Auswahl des Knotens parallel gelegene Schritte nun schon in einer Ordnungsbeziehung zu diesem Knoten. Die bloße Reaktivierung der zuvor berechneten Ordnungen führt deshalb häufig zu Inkonsistenzen.

Im Falle einer Wiederauswahl werden deshalb zunächst die bei der ersten Auswahl eingeführten und zwischenzeitlich inaktivierten Ordnungs-Constraints entfernt und dann eine Neuberechnung durchgeführt. Zum Abschluß der Diskussion folgt nun noch das Klassifikationsschema für diese Methode.

[8] Da diese Heuristik aber für kleine Konfliktmengen durchaus Sinn macht, wird sie dem Benutzer dennoch zur Verfügung gestellt.

Heuristik	Look Ahead
Wirkung	Linearisierung der Pläne
Transformation	Anwenden von Axiomen
Korrektheitsbedingungen	absolut korrekt
Hilfsmittel	Axiome
Komplexitätsfaktoren	Verwendung von Axiomen mit Constraints

4.2.3 Präferenz der Threatauflösung

In CAPlan sind Threats durch eine Klasse von Protektionszielen repräsentiert. Dies erlaubt, die Änderung des Threatauflösungszeitpunktes auf die Auswahlreihenfolge von Zielen zu reduzieren. Konkret besteht die Realisation darin, vor der eigentlichen Auswahl eines Zieles durch Aufruf einer Kontrollmethode zu entscheiden, ob ein Domänenziel oder ein Protektionsziel ausgewählt werden soll. Auf die gewählte Gruppe von Zielen werden dann weitere Auswahlverfahren angewendet. Für diese Metaentscheidung wurden drei Heuristiken implementiert.

4.2.3.1 Bevorzugung von Domänenzielen

Diese Heuristik haben Smith und Peot unter dem Namen 'DEnd' beschrieben [SP93]. Diese Heuristik bewirkt, daß Threats erst am Ende des Planungsvorganges bearbeitet werden. Man vermeidet dadurch, Threats explizit zu behandeln, die sich im Verlauf der Planung von selbst auflösen. Diese Strategie scheint jedoch nur sinnvoll für Domänen mit wenigen Threats, da sonst zu oft eigentlich schon inkonsistente Teilpläne weiter expandiert werden.

Heuristik	Least Commitment
Wirkung	Selbstauflösende Threats müssen nicht behandelt werden
Transformation	-
Korrektheitsbedingungen	Korrekt, falls alle Threats konsistent auflösbar sind.
Hilfsmittel	-
Komplexitätsfaktoren	-
Ergebnistyp	Bisektion

4.2.3.2 Bevorzugung von Protektionszielen

Dies ist die vom Originalalgorithmus vorgesehene Strategie: Domänenziele werden erst dann bearbeitet, wenn im bisherigen Plan keine Threats mehr bestehen. Diese Strategie führt in der Regel dazu, daß sich der Planer auf falsche Threatauflösungsstrategien festlegt und dadurch die Backtrackingfrequenz steigt. Allerdings werden durch diese Strategie die Pläne auch stärker geordnet, was viele Phantomisierungsoperatoren und Threatauflösungsmöglichkeiten ausschließt und damit nachfolgende Entscheidungen sicherer macht. Als Faustregel kann gelten, daß in Domänen mit relativ hoher Lösungsdichte eine frühe Threatauflösung vorteilhaft ist. In Domänen mit niedriger Lösungsdichte hingegen sind Verfahren vorzuziehen, die bei der Wahl des Threatauflösungszeitpunktes mitberücksichtigen, wie wahrscheinlich die Korrektheit der Auswahl ist.

Heuristik	-
Wirkung	Inkonsistente Pläne werden nicht weiter expandiert. Linearisierung der Pläne
Transformation	-
Korrektheitsbedingungen	-
Hilfsmittel	-
Komplexitätsfaktoren	-
Ergebnistyp	Bisektion

4.2.3.3 Bevorzugung eindeutig aufzulösender Threats

Diese Heuristik zieht nur eindeutig oder gar nicht aufzulösende Threats der Bearbeitung von Domänenzielen vor. Diese Heuristik wurde ebenfalls von Smith und Peot unter dem Namen 'DUnf' vorgestellt und hat sich in deren empirischen Untersuchungen als allen anderen überlegen herausgestellt. Diese Heuristik ist verwandt mit der in Abschnitt 4.3.2.1 vorgestellten Heuristik zur Auswahl aus einer Menge von Protektionszielen.

Heuristik	Bottleneck Planung
Wirkung	stärkere Einschränkung des Planes
Transformation	-
Korrektheitsbedingungen	Alle nicht eindeutig aufzulösenden Constraints lassen sich konsistent auflösen.
Hilfsmittel	-
Komplexitätsfaktoren	-
Ergebnistyp	Multisektion: eindeutig aufl. Threats, Domänenziele und nicht eindeutig auflösbare Threats.

4.2.4 Änderung der zugelassenen Threatauflösungsmöglichkeiten

4.2.4.1 Verzicht auf Separationsoperatoren

Der Verzicht auf Separationsoperatoren wurde zu einer Vermeidung von Separationsoperatoren abgeschwächt und wird in Abschnitt 4.3.4.2 besprochen.

4.2.4.2 Gleichzeitige Bearbeitung mehrerer Threats

Die simultane Bearbeitung mehrerer Threats konnte ebenfalls auf schon vorhandene Strukturen abgebildet werden, indem vor Auswahl des ersten Threats einer Gruppe die optimale Auflösung berechnet und dann über mehrere Planungszykel hinweg durchgeführt wird. Die genaue Vorgehensweise wird in Abschnitt 4.3.4.1 beschrieben.

4.2.5 Generalisierung des Inkonsistenzbegriffes

Die Generalisierung konnte ebenfalls ohne Eingriffe in den Algorithmus realisiert werden und wird durch drei separate Verfahren realisiert: Die Vermeidung inkonsistenter Domänenoperatoren wird in Abschnitt 4.3.3.2 beschrieben, die Vermeidung inkonsistenter Phantomisierungen in Abschnitt 4.3.3.3, und die Vermeidung inkonsistenter Threatauflösungen in Abschnitt 4.3.4.1.

4.3 Auswahlsteuerung

4.3.1 Kontrolle der Planungsreihenfolge

In diesem Abschnitt werden die verschiedenen Verfahren und Heuristiken zur lokalen Steuerung der Planungsreihenfolge vorgestellt.

4.3.1.1 Notwendige Zielanordnungen

Notwendige Zielanordnungen sind wie folgt definiert:

Definition 4 (Notwendige Zielanordnungen $\prec_N$)
Seien g_1, g_2 zwei Ziele mit gemeinsamem Vorfahr.
Es gilt $g_1 \prec_N g_2$ gdw in jeder Operatorsequenz, die $g_1 \wedge g_2$ erzeugt, der letzte g_1 einführende

Operator vor dem letzten g_2 einführenden Operator angeordnet ist.

Notwendige Zielanordnungen beschreiben die Reihenfolge, in der die Ziele bei Ausführung des Planes erzeugt werden. Wie aus obiger Definition hervorgeht, müssen zu ihrer Berechnung lediglich die direkten Erzeuger untersucht werden.

Die Grundsituation ist die folgende:

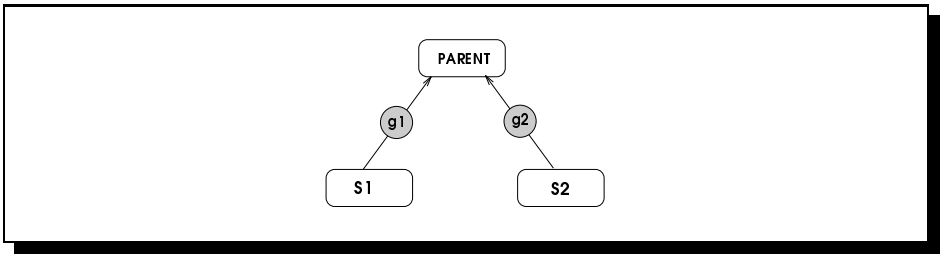

Abbildung 4.13: Grundsituation der Berechnung notwendiger Zielreihenfolgen

Seien g_1, g_2 wiederum Ziele, PARENT, S1, S2 Schritte. Mit precs(o) sei die Menge der Vorbedingungen, mit effs(g) die Menge der Effekte eines Operators o bezeichnet. Mit $\longrightarrow_{ax}$ wird die Folgerung mittels Axiomen bezeichnet. Um die Darstellung zu vereinfachen, wird zunächst angenommen, daß für jedes Ziel g nur ein möglicher Operator o(g) existiert.

1. $\neg g_2 \in effs(o(g_1))$
   Diese Situation entspricht einem gewöhnlichen Threat, bei dem S1 (= o(g_1)) den Causal Link von S2 bedroht. Da die Promotion S1 $\prec$ PARENT nicht möglich ist, ist die Demotion S1 $\succ$ S2 die einzige Möglichkeit zur Threatauflösung. Folglich wird bei Ausführung des Planes g_1 vor g_2 erzeugt werden.

2. $\neg g_2 \longleftarrow_{ax} effs(o(g_1))$
   S1 kann nicht der letzte Schritt vor Anwendung des Operators PARENT sein, da sonst der Zustand axiom-inkonsistent wäre. Daraus folgt die Erzeugungsreihenfolge $g_1 \prec g_2$.

3. $g_2 \in effs(o(g_1))$
   Dies entspricht –analog zu Fall 1)– einem gewöhnlichen Threat, bei dem wiederum die Promotion ausgeschlossen ist. Es folgt $g_1 \prec g_2$.

4. $g_2 \longleftarrow_{ax} effs((o(g_1))$
   Für positive Aussagen gilt, daß sie aus einer Konjunktion negierter Literale hergeleitet werden können. Da das Matchen von Konjunktionen aber wesentlich aufwendiger ist als das Matchen einzelner Literale wird dieser Fall nicht berechnet.

5. $\neg g_2 \in precs(o(g_1))$
   Die Argumentation läuft analog zum Fall 6).

6. $\neg g_2 \longleftarrow_{ax} precs(o(g_1))$
   Betrachten wir hierzu die folgende Zeichnung:
   In dieser Zeichnung wurde angenommen, daß S2 vor S1 angeordnet sei. P sei eine Vorbedingung von S1, die den Axiomen zufolge die Koexistenz von g_2 ausschließt. Wie ersichtlich wird, müßten p und g_2 bei dieser Anordnung aber im selben Zeittrahmen (dunkel unterlegt) gültig sein. Die Annahme ist demnach falsch; als einzig mögliche Anordnung kann $g_1 \prec g_2$ festgehalten werden.

7. $g_2 \in precs(o(g_1))$
   Für diese Situation läßt sich keine eindeutige Anordnung der erzeugenden Schritte angeben: Obige Zeichnung zeigt die zwei Lösungsmöglichkeiten. Die rechte, pathologische Situation tritt auf, wenn zunächst g_2 für PARENT geplant wird. Normalerweise würde dann bei der Planung

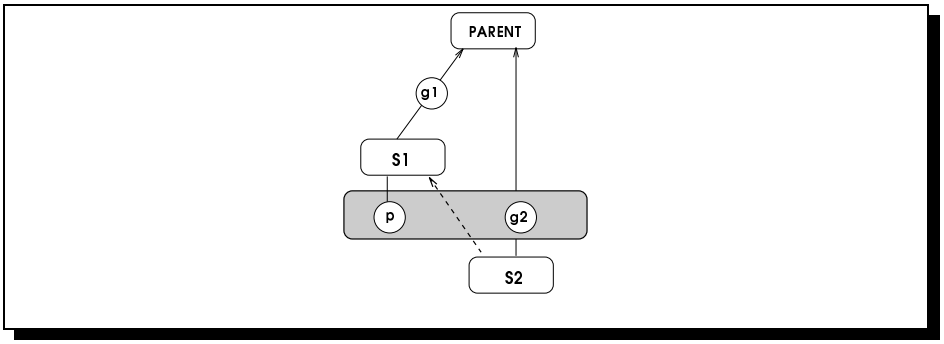

Abbildung 4.14: Die Situation unter Annahme der Ordnung S2 $\prec$ S1

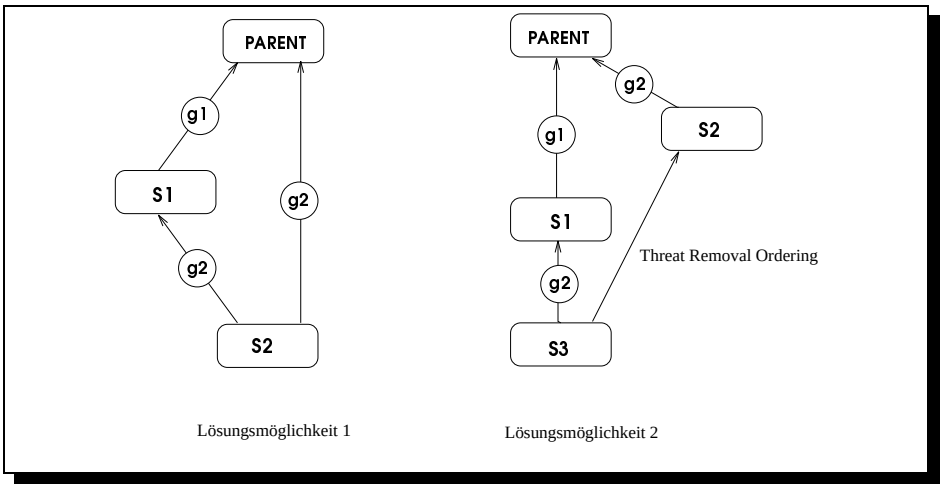

Abbildung 4.15: Die zwei Lösungsmöglichkeiten

von g_2 für S1 eine Phantomisierung mit S2 gewählt werden (Situation links). Ist aber S3 der einzige Operator, der eine weitere Vorbedingung von S1 einführt, so ist man gezwungen, den Seiteneffekt g_2 von S3 für S1 zu nutzen. Aufgrund des Causal Link Konzeptes von SNLP ist es nicht möglich, diesen Seiteneffekt von S3 auch zur Erplanung von g_2 für PARENT auszunutzen, weil man sich zuvor schon für S2 als Erzeuger entschlossen hat.

8. $g_2 \longleftarrow_{ax} precs((o(g_1))$
   Dieser Fall entspricht dem in Abb. 4.15 gezeigten Szenario. Wie ersichtlich, gibt es auch hier keine notwendige Anordnung.

Zusammenfassend läßt sich also sagen, daß nur die Fälle 1)-3, 5) und 6) getestet werden müssen, und daß bei Vorliegen einer dieser Situationen eine Anordnung $g_1 \prec g_2$ angenommen werden kann.

Nun noch einige Ergänzungen zur oben beschriebenen, grundsätzlichen Vorgehensweise:

- Effiziente Berechnung
  Die Berechnung wird wesentlich effizienter, wenn man die Symmetrie der Axiomfolgerungen ausnutzt. Anstatt auf jeden Effekt und jede Vorbedingung von $o(g_1)$ die Axiome anzuwenden, berechnet man nur für g_2 die Axiomfolgerungen, und bildet dann die Schnittmenge dieser Folgerungen mit der Menge der Effekte und Vorbedingungen.

- Erweiterung auf Konfliktmengen
  Die obigen Definitionen waren einer klareren Darstellung wegen zunächst unter der Annahme einer einelementigen Konfliktmenge gemacht worden. Sie lassen sich nun auf mehrere Arten auf größere Konfliktmengen ausweiten. Eine Möglichkeit wäre gewesen, nur die notwendigen Effekte

und Vorbedingungen eines Zieles zu betrachten, wie dies z.B. Etzioni tut. Die Erfahrung hat allerdings gezeigt, daß diese Mengen sehr oft leer oder nur sehr klein sind.
Da die Notwendigen Zielanordnungen aber nur einmalig in einer Domänenanalysephase berechnet werden und später nur auf die 'kompilierten' Ergebnisse zugegriffen wird, habe ich mich hier für eine aufwendigere, aber auch wesentlich mächtigere Vorgehensweise entschieden: Damit $g_1 \prec_{NO} g_2$ gilt, genügt, daß für jeden relevanten Operator $o \in ops(g_1)$ *einer* der Fälle 1)-3),5) oder 6) zutrifft.

- Interessant ist auch, daß abhängig davon, ob auf der strategischen Ebene beschlossen wurde, mit oder gegen die Ausführungsreihenfolge zu planen, die notwendigen Zielanordnungen für die Planungsreihenfolge unterschiedlich interpretiert werden. Der Einsatz ähnlicher Zielanordnungen für Planung entgegengesetzt zur Ausführungsreihenfolge ist von Drummond und Currie [DC88, DC89] unter dem Namen 'Temporal Coherence' beschrieben worden.

Heuristik	Maximale Informiertheit
Wirkung	Planung in Ausführungsreihenfolge; dadurch sicherere Zustandsinformation
Transformation	Testen einer kleinen Menge von Constraints
Korrektheitsbedingungen	Korrekt, falls die Ausführungsreihenfolge die optimale Planungsreihenfolge darstellt. Dies ist z.B. für trivial und aufwendig serialisierbare Ziele [BW93] der Fall
Hilfsmittel	Axiome, Domänenanalyse
Komplexitätsfaktoren	Verwendung von Axiomen mit Constraints
Ergebnistyp	Partialordnung

4.3.1.2 Zielsubsumtionsordnung

Wir haben oben gesehen, daß im Fall 7) keine *notwendige* Anordnung der Ziele angegeben werden kann. Dennoch ist eine der beiden möglichen Planungsreihenfolgen der anderen unbedingt vorzuziehen: Wird zuerst das subsumierende Ziel g1 geplant, so erhält man die links gezeigte, optimalere Lösung.

Heuristik	Opportunismus, Konfliktvermeidung
Wirkung	Begünstigt das Finden minimaler Pläne. Verhindert Backtracking aufgrund von positiven Threats.
Transformation	Testen einer kleinen Menge von Constraints
Korrektheitsbedingungen	-
Hilfsmittel	Domänenanalyse, Axiome
Komplexitätsfaktoren	Axiome mit Constraints
Ergebnistyp	Partialordnung

4.3.1.3 Vermeidung unvollständig instantiierter Ziele

Sind Ziele nicht vollständig instantiiert, dann ist die Menge der Phantomisierungsmöglichkeiten größer und auch die meisten Domänenoperatorauswahlmethoden verlieren an Aussagekraft. Eine gängige Heuristik ist daher, zuerst solche Ziele zu bearbeiten, die möglichst wenige Variablen enthalten und daher auf einer besseren Informationsgrundlage bearbeitet werden können. Man kommt natürlich nicht umhin, auch die Ziele mit Variablen zu bearbeiten; durch die Hinauszögerung wird die Informationsgrundlage jedoch besser, da dem Plan laufend Constraints hinzugefügt werden.

Heuristik	Maximale Informiertheit
Wirkung	Vermeidet unsichere Operatorauswahl - für nicht vollst. instanziierte Ziele ist die Zustandsinformation zu vage.
Transformation	-
Korrektheitsbedingungen	-
Hilfsmittel	
Komplexitätsfaktoren	-
Ergebnistyp	Multisektion

4.3.1.4 Vermeidung phantomisierbarer Ziele

Im praktischen Einsatz des Planers hat sich gezeigt, daß zu Beginn der Planung gemachte Phantomisierungen häufig falsch sind. Diese frühen Fehler sind aber –bedingt durch die Größe des Suchraumes – kaum in akzeptabler Zeit durch Backtracking korrigierbar. Es wird daher versucht, dieser Tendenz zu voreiligen Phantomisierungen entgegenzuwirken, indem zuerst diejenigen Ziele bearbeitet werden, für die keine Phantomisierungsmöglichkeit besteht. Eine zweite Begründungsmöglichkeit dieser Heuristik ist die Tatsache, daß die Menge der Phantomisierungsoperatoren im Gegensatz zur Menge der möglichen Domänenoperatoren während der Planung noch zunehmen kann. Daher ist es möglich, daß bei zu frühzeitiger Bearbeitung eines Zieles die letztendlich richtige Alternative noch gar nicht in der Konfliktmenge enthalten ist.

Heuristik	Maximale Informiertheit, Bottleneck
Wirkung	Verhindert tiefes Backtracking.
Transformation	Konfliktmengenberechnung für alle Ziele
Korrektheitsbedingungen	-
Hilfsmittel	(PMU)[9]
Komplexitätsfaktoren	Viele Ziele, viele Operatoren, viele Effekte, stark parallele Pläne.
Ergebnistyp	Bisektion in phantomisierbare und nicht ph. Ziele.

4.3.1.5 Bevorzugung von tiefgelegenen Zielen

Diese Heuristik realisiert eine Variante der Tiefensuche. Normalerweise wird die Tiefensuche durch eine LIFO-Verwaltung der Konfliktmenge realisiert. Soll diese Heuristik aber mit anderen Verfahren kombinierbar sein, die nicht notwendigerweise die Reihenfolge der Ziele in der Konfliktmenge unverändert lassen, so muß anders vorgegangen werden. Hier wurde eine Möglichkeit gewählt, bei der die Ziele nach der Tiefe der sie einführenden Schritte im Plan geordnet werden. Die Tiefe eines Schrittes ist dabei als Entfernung vom Zielknoten FINISH definiert, der hinter allen anderen Schritten angeordnet ist.

Obwohl die Tiefensuche so etabliert ist, daß sie an sich keiner eigenen Rechtfertigung bedarf, lassen sich –insbesondere auch für die hier vorgestellte Variante– recht starke Argumente für ihren Einsatz finden:
Wie wir bei der vorangegangenen Heuristik schon bemerkt haben, ändert sich die Menge der Phantomisierungsmöglichkeiten im Laufe der Planung. Möchte man nun erreichen, daß die Menge der Phantomisierungen zum Auswahlzeitpunkt vollständig ist, so muß man sicherstellen, daß alle Schritte, die zur Erreichung anderer Ziele dienen und im endgültigen Plan vor der aktuellen Position liegen werden, schon im Plan enthalten sind. Diese Schritte stellen nämlich gerade die Erzeuger von potentiellen Phantomisierungsmöglichkeiten dar.

Die bevorzugte Behandlung von tiefen Zielen unterstützt dies in zweifacher Hinsicht: 1.) Für tiefgelegene Ziele ist (zumindest innerhalb derselben Teilzielhierarchie) die Wahrscheinlichkeit geringer, daß später noch andere Schritte davor angeordnet werden bzw. ist für die höhergelegenen Ziele klar, daß sich noch andere Schritte davorschieben. 2.) Durch die Bearbeitung tief gelegener Ziele vergrößert sich die Informationsgrundlage, auf der die höhergelegenen geplant werden können.

Obiger Argumentationsstrang gilt natürlich nicht nur für die Menge der Phantomisierungsoperatoren,

sondern ganz allgemein für Zustandsinformationen. Mittels dieser Heuristik kann daher die Effektivität aller zustandsbasierten Kontrollmethoden erhöht werden.

Zuletzt läßt sich noch anmerken, daß diese Heuristik auch dazu beiträgt, den maschinellen Planungsvorgang natürlicher erscheinen zu lassen, weil sie eine natürlich scheinende, lokal fokussierende Vorgehensweise darstellt. Im Falle relativ wenig interagierender Ziele läßt sich diese natürliche Strategie auch komplexitätstheoretisch untermauern: Korf [Kor87] hat gezeigt, daß die separate Bearbeitung unabhängiger Teilziele einer Division des Exponenten entspricht.

Heuristik	Maximale Informiertheit
Wirkung	Fokussiertes Planen. Planen in Ausführungsreihenfolge.
Transformation	Minima-Berechnung aller Ordnungs-Constraints im Plan
Korrektheitsbedingungen	-
Hilfsmittel	-
Komplexitätsfaktoren	Viele Ordnungsconstraints, viele Schritte, viele Ziele.
Ergebnistyp	Multisektion

4.3.1.6 Zielanordnungen nach Etzioni

Etzioni [Etz90] hat aus seinen PSGs auch Kontrollinformationen für die Zielauswahl extrahiert. Wir wollen seinen Ansatz hier nur kurz wiederholen und vor allem auf die notwendig gewordenen Änderungen eingehen; für eine ausführliche Beschreibung der Originalmethoden sei auf die Literatur verwiesen.

Definition verschiedener Interaktionstypen. Etzioni berechnet zur Steuerung der Zielauswahl zwei Typen von Zielinteraktionen: prerequisite violations und goal clobberings. Sie sind wie folgt definiert:

Sei der Matchoperator durch | bezeichnet; k sei eine Zustandsbedingung, c und c' seien Operatorsequenzen, s ein Zustand, pg (protected goal)und vg (violating goal) Ziele.

Definition 5 (prerequisite violations)
$\Box V(g, vg, \sigma, k) \equiv \forall s\ s.t. \sigma(k) \mid s\ and\ \forall c\ s.t. \sigma(vg) \mid c(s), \neg\exists c'\ \sigma(pg) \mid c(s)$

Definition 6 (goal clobbering)
$\Box C(g, pg, \sigma, k) \equiv \forall s\ s.t.\ \sigma(pg) \mid s\ \wedge\ \sigma(k) \mid s, and\ \forall c\ s.t.\ \sigma(g) \mid c(s),\ we\ have\ that\ \neg\sigma(pg) \mid c(s)$

Diese zwei Interaktionstypen werden berechnet, indem zunächst mit Hilfe der PSGs die notwendigen Effekte bestimmt werden. Diese werden dann mit den notwendigen, nicht durch Operatoren erzeugbaren Vorbedingungen bzw. den Zielen geschnitten. Durch Kombination der Bedingungen für das Auftreten eines notwendigen Effektes mit den Unifikationsbedingungen und den Bedingungen für das Auftreten notwendiger Vorbedingungen erhält man eine Menge von Bedingungen für das Auftreten der zwei Interaktionstypen.

Dieses Wissen wird dann in Form von Regeln genutzt, die im Falle der prerequisite violations das zerstörende Ziel nach dem Ziel anordnet, das die Vorbedingungen benötigt; im Falle von goal clobberings wird das zerstörende Ziel zuerst bearbeitet. Dieser Originalansatz wurde nun um drei weitere Interaktionstypen erweitert:

1. **goal establishments**
   Da SNLP auch positive Interaktionen als Konflikt betrachtet, bestand die Notwendigkeit, eine positive Variante der goal clobberings mitzuverwenden:

   Definition 7 (goal establishments)
   $\Box C(g, pg, \sigma, k) \equiv \forall s\ s.t. \sigma(k) \mid s, and\ \forall c\ s.t.\ \sigma(g) \mid c(s),\ we\ have\ that\ \sigma(pg) \mid c(s)$

   Ganz analog zu den goal clobberings wird das Wissen um diese Interaktion so ausgenutzt, daß das erzeugende Ziel g zuerst geplant wird, um den Konflikt zu vermeiden. Ein erfreulicher Seiteneffekt dieser Praxis ist die Begünstigung optimalerer Pläne, weil in dieser Reihenfolge für das zweite Ziel eine zusätzliche Phantomisierungsmöglichkeit erzeugt wird.

2. **precondition violations**
   Bei Etzioni's prerequisite violations werden nur solche Vorbedingungen betrachtet, die nicht durch Operatoren erzeugt werden können. Es gibt allerdings viele Domänen, in denen solche Vorbedingungen gar nicht auftreten. Es bestand daher das Bedürfnis, einen Interaktionstyp für gewöhnliche Vorbedingungen zu definieren.
   Es ist jedoch nicht möglich, notwendige Vorbedingungen analog zu den notwendigen Effekten zu definieren.[10] Da bei den notwendigen Effekten die Erfahrung gemacht wurde, daß durch die Schnittbildung im PSG meist nur Effekte der ersten Operatorschicht übrig bleiben, wurden hier gleich von vornherein nur die notwendigen Vorbedingungen der ersten Operatorschicht verwendet. Diese sind wie folgt definiert:

   Definition 8 (Notwendige direkte Vorbedingungen)
   $NecDirPrecs(g) = \cap_{o \in ops(g)} precs(o)$

   Für die precondition violations wird nun die Menge der direkten, notwendigen Vorbedingungen mit den notwendigen Effekten geschnitten. Das Wissen um eine precondition violation wird in der Planung analog zu den prerequisite violations so verwendet, daß das zerstörende Ziel nach dem Ziel angeordnet wird, dessen notwendige direkte Vorbedingungen zerstört werden.

3. **precondition establishment**
   Schließlich bietet sich an, auch für die Vorbedingungen eine positive Variante zu schaffen. Im Falle eines Precondition Establishments wird wiederum das erzeugende Ziel zuerst geplant, um bei Planung des zweiten Zieles mehr Phantomisierungsmöglichkeiten zur Verfügung zu haben.

Kombination der Interaktionstypen. Die Auswertung der Interaktionstypen ergibt oft eine widersprüchliche Menge von Anordnungen. Zusätzlich zu den Einzelheuristiken wurde deshalb eine Kombination dieser vier Heuristiken vordefiniert, die diese vier Interaktionstypen entsprechend folgender Dominanzordnung auswertet:

goal clobberings $\succ$ goal establishment $\succ$ precondition violation $\succ$ precondition establishment.

Die prerequisite violations wurden nicht in diese Standardkombination aufgenommen, weil ihr Einsatz nur in einigen Domänen sinnvoll ist; Die Definition einer solchen Kombination ist aber mit der in Kapitel 5 vorgestellten Kontrollkonfigurationsoberfläche leicht zu bewerkstelligen.

Modifikationen des Originalansatzes. Außer der Ausweitung auf andere Interaktionstypen mußten für den Einsatz in CAPlan auch einige Änderungen vorgenommen werden:

- Behandlung von Phantom-Vorbedingungen
  Für Phantom-Vorbedingungen gilt ja, daß sie nicht durch Domänenoperatoren erplant werden dürfen. Dem muß beim Aufbau der PSGs Rechnung getragen werden, indem auch dort Phantom-Ziele nicht weiter expandiert werden. Es gibt nun allerdings zwei Möglichkeiten, die Phantomziele zu markieren:

  Wurden die Phantom-Vorbedingungen bei der Domänenmodellierung dazu verwendet, Anwendbarkeitsbedingungen zu formulieren, so müssen sie im PSG als 'unachievable' markiert werden, damit sich in den Fehlschlagsbedingungen die intendierten Anwendbarkeitsbedingungen widerspiegeln.

  Wurden die Phantom-Vorbedingungen allerdings dazu verwendet, diejenigen Vorbedingungen zu markieren, durch deren Phantomisierung die noch offenen Variablen eines Operators gebunden werden sollen, so wird von der Gültigkeit dieser Vorbedingungen zum Zeitpunkt der Operatorauswahl ausgegangen. In diesem Falle sollten die Phantom-Ziele im PSG mit 'holds' markiert werden, damit die Fehlschlagsbedingungen nicht unnötig aufgebläht werden und ihre Auswertung billiger wird.

  Da die korrekte Markierungsart von der Art der Domänenmodellierung abhängt, wird vor dem einmaligen Aufbau der PSGs der Benutzer nach der richtigen Variante befragt.

[10] Auf die Gründe soll hier nicht näher eingegangen werden; als Hinweis mag jedoch genügen, daß nur disjunktive Ausdrücke von Vorbedingungen berechnet werden können; diese erlauben jedoch keine Aussagen über notwendige Konflikte.

- Behandlung von Operatorconstraints
  Wie in Abschnitt 4.1.2.2 bereits beschrieben, mußten die PSGs um eine Constraintbehandlung erweitert werden. Dies wirkt sich auch auf die Interaktionsbedingungen aus, weil nun z.B. berücksichtigt werden muß, daß ein Zielknoten deswegen weiter expandiert werden mußte, weil kein Match gefunden werden konnte, der den Constraints genügt hätte. Jede atomare Zustandsbedingung im PSG wurde daher um Constraints erweitert. Bei der Bildung von Schnittmengen werden die Constraintmengen konjunktiv verknüpft.

- Auswirkung der Änderung auf die Semantik
  Zum einen gilt, daß durch das unterschiedliche Zielrekursionskriterium die Fehlschlagsbedingungen nicht mehr korrekt sein müssen. Die Fehlschlagsbedingungen gehen mit in die Bedingungen der notwendigen Effekte und damit auch in die Bedingungen der berechneten Interaktionen sein. Damit sind die Bedingungen also weniger verläßlich als dies für PRODIGY der Fall war. Da für SNLP-Pläne außerdem nur ein verallgemeinerter, weniger korrekter Zustand berechnet werden kann, wird auch die Auswertung der Bedingungen unsicherer.
  Schließlich macht die Mitbetrachtung von Constraints die Auswertung der Bedingungen teurer. Eine Nichtbeachtung der Constraints ist jedoch auch nicht möglich, weil sie zumeist wichtige Bedeutungsträger sind.
  Insgesamt gilt also, daß die Mächtigkeit dieser Kontrollmethode in ihrer Modifikation für CAPlan stark eingeschränkt werden mußte.

Heuristik	Look Ahead, Opportunismus, Monotoner Fortschritt
Wirkung	Natürliches Planen, da vorausschauend. Planen in Ausführungsreihenfolge.
Transformation	Constrained Match mittlerer bis großer Bedingungsausdrücke.
Korrektheitsbedingungen	- (konnten nicht übertragen werden)
Hilfsmittel	Axiome, PSGs, Domänenanalyse
Komplexitätsfaktoren	Parallele Pläne. Großer Zustand. Viele ähnliche Literale. Wenige, sehr ähnliche Operatoren. Viele Invarianten
Ergebnistyp	Partialordnung

4.3.1.7 Bevorzugung von Phantom-Zielen

Wie in Abschnitt 2.2.3 beschrieben, bietet CAPlan die Möglichkeit einer einfachen Typisierung der Vorbedingungen: einige Vorbedingungen können als Phantom-Vorbedingungen deklariert werden. Diese Vorbedingungen dürfen vom Planer nur phantomisiert, nicht aber durch Domänenoperatoren erzeugt werden.

Der sinnvollste Einsatz solcher Phantom-Vorbedingungen ist, durch sie freie Operatorvariablen zu binden. Wenn bei der Domänenmodellierung Vorbedingungen mit dieser Intention als Phantom-Ziele deklariert worden sind, sollte dem während der Planung durch eine bevorzugte Bearbeitung Rechnung getragen werden. Zum einen gilt, daß in diesen Fällen Domänenoperatoren von vornherein ausgeschlossen sind, und dadurch die Konfliktmenge lediglich auf Phantomisierungsoperatoren beschränkt ist. Für Ziele, die selten als Seiteneffekte entstehen, ist daher die Konfliktmenge klein und die Entscheidung somit relativ sicher. Zum andern werden durch das Binden freier Operatorvariablen die anderen durch diesen Operator eingeführten Ziele instantiiert, was wiederum zu einer Einschränkung von deren Konfliktmengen führt. Wie schon im Abschnitt 4.4.1.3 (Vermeidung nicht vollständig instantiierter Ziele) näher erläutert worden ist, erhöht man damit die Sicherheit der Operatorauswahl.

Heuristik	Bottleneck Planung
Wirkung	Bindet Variablen und erhöht dadurch den Informationsstand
Transformation	
Korrektheitsbedingungen	Korrekt, falls die Phantomvorbedingungen bei der Domänenmodellierung in dieser Bedeutung eingesetzt worden sind.
Hilfsmittel	-
Komplexitätsfaktoren	-
Ergebnistyp	Bisektion

4.3.2 Kontrolle der Threatauflösungsreihenfolge

4.3.2.1 Bevorzugung eindeutig auflösbarer Threats

Diese Strategie ist eine Erweiterung der in Abschnitt 4.2.3.3 vorgestellten Heuristik gleichen Namens. Mit letzterer hatte ich bei stark parallelen Plänen die Erfahrung gemacht, daß sich sehr viele unaufgelöste Threats ansammeln können. Eine ganz natürliche Verallgemeinerung dieser Strategie ist deshalb, die Protektionsziele nur nach der Anzahl ihrer Auflösungsmöglichkeiten zu sortieren. Auch Joslin und Pollack [JP94] verwenden diese Erweiterung; sie schlagen darüber hinaus vor, die Ordnung bezüglich der Operatormengengröße uniform auf alle Ziele –Domänenziele und Protektionsziele– auszuweiten.

Heuristik	Maximale Informiertheit
Wirkung	Versucht Threats erst dann aufzulösen, wenn klar ist, wie.
Transformation	Test der Konsistenz von Ordnungen und Bindungsconstraints (die Konfliktmengen müssen nun mehrfach berechnet werden)
Korrektheitsbedingungen	-
Hilfsmittel	-
Komplexitätsfaktoren	Viele Ordnungen im Plan; Verwendung von Separation
Ergebnistyp	Multisektion

4.3.2.2 Vermeidung separierbarer Threats

Diese Strategie ist bei der Diskussion möglicher Feinarchitekturänderungen schon näher erläutert worden und heißt bei Peot und Smith 'DSep' (Delay Separation). Sie bevorzugt die Bearbeitung von Threats, die nicht durch Separation auflösbar sind und daher maximal zwei Auflösungsmöglichkeiten besitzen. Da fertige Pläne stets vollständig instantiiert sind, wandeln sich separierbare Threats nach Instantiierung der betroffenen Variablen entweder in inaktive oder in nicht separierbare Threats um. Durch diese Heuristik gehen also keine Threats verloren und das Planverfahren bleibt korrekt.

Heuristik	Least Commitment
Wirkung	Bearbeitet nur notwendige Threats
Transformation	-
Korrektheitsbedingungen	Korrekt bei entsprechender Definition eines Threat.
Hilfsmittel	-
Komplexitätsfaktoren	-
Ergebnistyp	Bisektion

4.3.3 Kontrolle der Operatorauswahl

4.3.3.1 Fehlschlagsbedingungen

Failure Conditions werden mit Hilfe der PSGs berechnet [Etz90], indem –ganz analog zu EBL-Verfahren– die Fehlschlagsbedingungen eines Blattes den Suchpfad entlang zurückpropagiert werden. Im Gegensatz zu EBL-Verfahren wird dies aber für alle Lösungsmöglichkeiten gleichzeitig gemacht:

die Fehlschlagsbedingungen aller Operatoralternativen an einem Zielknoten werden durch ODER verknüpft. So entstehen ganz universelle Fehlschlagsbedingungen, die durch Termumformung noch vereinfacht werden.

Diese Fehlschlagsbedingungen waren für den totalordnenden Planer PRODIGY korrekt und konnten nicht nur als Präferenzregeln zur Operatorauswahl verwendet werden, sondern sogar zum *Ausschluß* von Operatoralternativen. Diese Eigenschaft ließ sich für SNLP leider nicht erhalten; der Grund dafür liegt in der Verwendung unterschiedlicher Zielrekursionskriterien: PSGs beenden die Expansion, nachdem ein Ziel rekurriert; CAPlan hingegen darf die Expansion erst dann beenden, wenn zudem derselbe Reduktionsoperator ausgewählt wird. Etzioni's Fehlschlagsbedingungen markieren also einen Operator, der ein rekurrierendes Ziel einführt, als Fehlschlag, obwohl für CAPlan dieser Suchpfad nicht oder erst viel später zu einem Fehlschlag führt.

Wie bei der Erläuterung des Zielrekursionskriteriums bereits ausgeführt worden ist, führt in CAPlan die weitere Expansion von rekurrierenden Teilzielen nur dann zu einer Lösung, wenn die Ziele in einer falschen Reihenfolge bearbeitet worden sind. In solch einem Fall müssen dann die Ziele, die eigentlich hätten zuerst geplant werden müssen, durch Seiteneffekte der Expansion des zuerst bearbeiteten Ziels erplant werden. Die Verwendung der Fehlschlagsbedingungen macht also trotz eingeschränkter Korrektheit Sinn, weil sie evtl. dazu führt, dieses unnatürliche 'indirekte' Planen zu vermeiden.

Heuristik	Look Ahead
Wirkung	Vermeidet die Auswahl fehlschlagender Operatoren.
Transformation	Constrained Match mittlerer bis riesiger Ausdrücke
Korrektheitsbedingungen	Bisher keine falsche Zielreihenfolge (Sonst falsche Zustandsinformation).
Hilfsmittel	PSGs, Axiome, Domänenanalyse
Komplexitätsfaktoren	Stark parallele Pläne. Großer Zustand. Viele ähnliche Literale. Wenige, sehr ähnliche Operatoren.

4.3.3.2 Vermeidung NO-inkonsistenter Domänenoperatoren

Die Hauptidee dieses Verfahrens[11] ist es, diejenigen Operatoren zu vermeiden, für die die Menge der Notwendigen Schrittanordnungen (vgl. 4.2.2.2) inkonsistent wäre, das heißt also solche Operatoren, deren Einführung früher oder später zu unauflösbaren Threats führen würde. In diesem Sinne wird also bei diesem Verfahren der Begriff des Threats durch die Verwendung von Axiomen verallgemeinert zu einem vorausschauendem Threatbegriff: Normalerweise tritt ein Threat auf, wenn der Effekt eines Schrittes logisch äquivalent ist zum Literal eines Causal Link bzw. dessen Negation. Ein verallgemeinerter Threat tritt auf, wenn durch Axiomfolgerungen gezeigt werden kann, daß die zwei Literale sich gegenseitig bedingen oder ausschließen.

Wie in Abschnitt 4.2.2.2 beschrieben, werden nicht alle dieser verallgemeinerten Threats berechnet, sondern nur die, für die genau eine Möglichkeit zur Auflösung besteht. Diese eindeutigen Ordnungsconstraints werden dann gleich bei Einführung des Schritts in den Plan eingefügt. Für die Operatorauswahl spielen diese Ordnungen nun insofern eine Rolle, als durch sie solche Operatoren früher erkannt werden können, die zu keiner Lösung führen können, weil die durch sie erzeugten verallgemeinerten Threats nicht konsistent aufgelöst werden können.

Bevor ein Operator endültig ausgewählt wird, wird bei dieser Kontrollmethode die Menge der nötigen Ordnungsconstraints berechnet und auf Konsistenz überprüft. Ist eine konsistente Auflösung unmöglich, so wird dieser Operator zurückgestellt und ein anderer, konsistenter zuerst ausgewählt. Stehen keine konsistenten Operatoren mehr zur Verfügung, so wird einer der inkonsistenten ausgewählt und für diesen Schritt die Einführung der Threatauflösungsconstraints unterdrückt, damit der Plan nicht inkonsistent wird.

[11] NO steht für Necessary Orderings. Durch dieses Verfahren sollen also Operatoren vermieden werden, die bezüglich der notwendigen Schrittanordnungen inkonsistent sind.

Heuristik	Look Ahead
Wirkung	Vermeidet fehlschlagende Operatoren.
Transformation	Constrained Match kleiner bis mittlerer Ausdrücke
Korrektheitsbedingungen	Zum Berechnungszeitpunkt unbedingt korrekt.
Hilfsmittel	Axiome
Komplexitätsfaktoren	Viele Vorbedingungen. Viele Interaktionen. Axiome mit Constraints

4.3.3.3 Vermeidung inkonsistenter Phantomisierungen

Manchmal wählt der Planer Phantomisierungen, die Causal Links mit unauflösbaren Threatmengen erzeugen. Der Planer merkt zwar später, daß eine Auflösung nicht möglich ist; besser ist aber, solche Phantomisierungen von vornherein zu meiden, wenn leicht zu testende Kriterien zur Verfügung stehen, um die Unmöglichkeit einer Phantomisierung festzustellen. Wir verwenden zwei solcher Kriterien:

1. Keine Phantomisierungen mit schon 'verbrauchten' Literalen
   Wird der Effekt, mit dem eine Phantomisierung möglich ist, schon von einem anderen Schritt durch eine Phantomisierung genutzt und verbraucht, so ist eine erneute Phantomisierung sinnlos, weil die entstehenden Threats unauflösbar sind. Eine Auflösung der Threats durch Rücknahme der ersten Phantomisierung ist nur bei Backtracking möglich; im Zuge des Backtracking würde aber auch die neue Phantomisierung zurückgenommen werden.[12]

2. Keine Phantomisierungen, die Causal Links erzeugen, die notwendigerweise parallel zu anderen Erzeugern oder Zerstörern liegen.
   Parallele Erzeuger/Zerstörer sind deshalb zu vermeiden, weil sie unauflösbare Threats erzeugen. Nach Einführung des Causal Links würde zwar auch CAPlan die Unauflösbarkeit der Situation erkennen; dieses Kriterium ist aber deshalb wichtig, weil es unmögliche Alternativen aus der Konfliktmenge aussortiert, und so z.B. frühzeitig erkannt werden kann, daß ein Ziel durch Domänenoperatoren geplant werden muß, weil alle Phantomisierungsoperatoren zu Inkonsistenzen führen.

Heuristik	Look Ahead
Wirkung	Vermeidet fehlschlagende Phantomisierungen.
Transformation	Fortlaufende PMU-Verwaltung. Matching und finden von Pfaden zwischen zwei Knoten
Korrektheitsbedingungen	absolut korrekt
Hilfsmittel	Axiome, (PMU)
Komplexitätsfaktoren	Viele Effekte, viele Vorbedingungen, langgezogene Pläne.

4.3.3.4 Gewichtete Minimum Conspiracy Strategie

Das Minimum Conspiracy Kriterium ist erstmals von Elkan[Elk89] für die Steuerung von Best-First-Suche in UND-ODER-Bäumen beschrieben worden. Nach dieser Strategie wird der Plan mit der kleinsten Menge noch offener Ziele weiter expandiert. Veloso und Blythe [BV92] haben dieses Kriterium erfolgreich für die Steuerung von Tiefensuche eingesetzt: hier reduziert sich das Kriterium darauf, den Operator auszuwählen, der die wenigsten unerfüllten Vorbedingungen hat. Diese Strategie ist in der Literatur auch unter dem Namen 'min-goals' bekannt und wird Minton zugeschrieben.

Bei CAPlan gibt es - im Gegensatz zum totalordnenden Planer PRODIGY- keinen eindeutigen Zustand am Punkt der Operatorauswahl. Daher sind Aussagen über den Erfülltheitsgrad von Vorbedingungen eigentlich nicht möglich. Man zieht sich daher auf den verallgemeinerten Zustand zurück, der aber keine Aussagen darüber zuläßt, ob eine Menge von Matches für die Vorbedingungen eines Operators auch gemeinsam genutzt werden kann. CAPlan bietet durch die Differenzierung der Vorbedingungen aber auch eine Stärkung dieses Kriteriums gegenüber der ursprünglichen Version: das

[12] Bisher sind alle in CAPlan zur Verfügung stehenden Backtrackingverfahren Spielarten des chronologischen Backtracking.

Erfülltsein *relevanter* Vorbedingungen kann jetzt höher gewichtet werden als das Erfülltsein weniger ausschlaggebender Vorbedingungen. Ganz konkret wird dies so realisiert, daß für jeden Operator eine optimale Substitution der eingeführten Variablen bestimmt wird und für jede Klasse von Vorbedingungen die Erfülltheitsrate berechnet wird. Die Menge der möglichen Operatoren wird dann bezüglich dieser Erfülltheitsgrade entsprechend der Wichtigkeit der Vorbedingungklassen lexikographisch geordnet. Die absolute Zahl der Vorbedingungen wird zusätzlich zur Verfeinerung dieser Ordnung eingesetzt. Zur Zeit werden folgende Vorbedingungsklassen verwendet:

1. Phantomvorbedingungen
2. Vorbedingungen, die nur Variablen enthalten, die durch Matchen eines Operatoreffektes mit dem gewünschten Ziel gebunden worden sind
3. Vorbedingungen, die noch ungebundene Variablen oder gar keine Variablen enthalten.

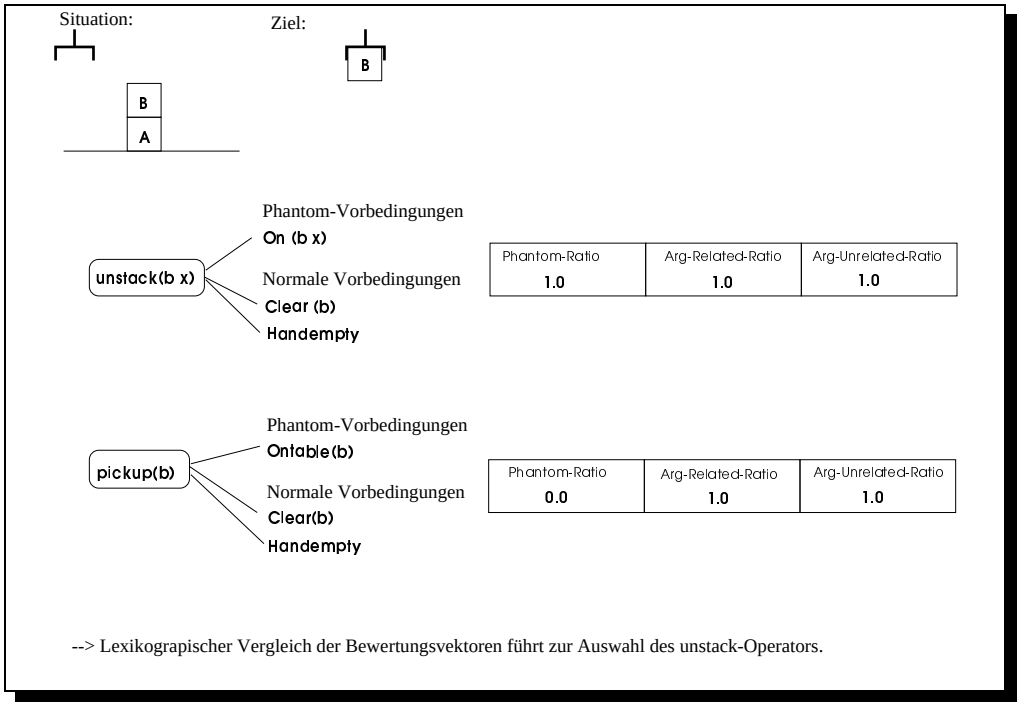

Abbildung 4.16: Arbeitsweise der gewichteten Minimum Conspiracy Strategie

Beispiel:

Heuristik	Small-Is-Quick
Wirkung	Auswahl des passendsten Operators
Transformation	Finden von besten Matches (mehrmals)
Korrektheitsbedingungen	Abhängig von Korrektheit des verallgem. Zustandes und davon, ob in der Domäne die verwendete Relevanzordnung für die Vorbedingungen der Operatoren gilt.
Hilfsmittel	-
Komplexitätsfaktoren	Großer Zustand, viele Operatoren viele Vorbedingungen

4.3.3.5 Bevorzugung von Phantomisierungen

Dies ist eine einfache Strategie, die auch in der Defaultkontrollkomponente SNLP+ enthalten ist: Ziele werden bevorzugt durch Phantomisierungen erreicht, anstatt neue Operatoren einzuführen. Weil

Phantomisierungsoperatoren den Basisfall der Planung darstellen, muß diese Heuristik eigentlich immer eingesetzt werden.

Heuristik	Small-Is-Quick
Wirkung	Beendet Expansion eines Zweiges
Transformation	-
Korrektheitsbedingungen	Führt zu falschen Entscheidungen, wenn viele Interaktionen auftreten.
Hilfsmittel	-
Komplexitätsfaktoren	-

4.3.3.6 Bevorzugung von Haupteffekten

Dies ist eine einfache Strategie, die auch in der Defaultkontrollkomponente SNLP+ enthalten ist: Man bevorzugt die Operatoren, die den gewünschten Effekt als Haupteffekt haben.

Heuristik	Maximale Informiertheit
Wirkung	Bevorzugt Standardoperatoren bzw Haupteffektplanung
Transformation	-
Korrektheitsbedingungen	Keine Angabe von Korrektheitsbedingungen möglich; gut in Domänen, in denen die Ausnutzung von Seiteneffekten keine Rolle spielt.
Hilfsmittel	-
Komplexitätsfaktoren	-

4.3.3.7 Bevorzugung von Invarianten

Invarianten sind spezielle Literale, die nicht durch Operatoren zerstört werden können und deshalb ideale Kandidaten für Phantomisierungen sind: mit einer einzigen Invarianten können beliebig viele Benutzer phantomisiert werden. Für die Phantomisierung mit Invarianten wird auch kein neuer Ordnungsconstraint eingeführt, weil die Invarianten schon im Startzustand gelten und alle Schritte hinter dem Startzustand angeordnet sind. Zu Problemen kann es deshalb nur kommen, wenn durch die Phantomisierung mit Invarianten Variablen auf eine Weise gebunden wurden, die später zu Konflikten führt. Diese Heuristik ist ebenfalls schon in der Defaultkontrollkomponente SNLP+ enthalten.

Heuristik	Least Constraining
Wirkung	Es wird kein Causal Link eingeführt $\rightarrow$ keine Threats, keine zusätzliche Ordnung
Transformation	-
Korrektheitsbedingungen	Absolut korrekt, falls das Ziel vollständig instantiiert ist. Sonst Einführung falscher Bindungen möglich.
Hilfsmittel	-
Komplexitätsfaktoren	-

4.3.4 Kontrolle der Threatauflösung

4.3.4.1 Algebraische Konfliktauflösung nach Yang

Analog zu algebraischen Verfahren zur Lösung von Constraint Satisfaction Problemen hat Yang [Yan, Yan92] ein Regelsystem zur algebraischen Auflösung einer Menge von Threats angegeben, durch das inkonsistente Threatauflösungskombinationen und redundate Auflösungsoperatoren herausgefiltert werden. Durch diese Analyse wird ermöglicht, die Menge der Threats durch eine minimale Anzahl von in den Plan einzuführenden Constraints aufzulösen. Eine minimale Auflösung muß allerdings nicht die korrekte Lösung sein, sie entspricht jedoch einem Least Constraining Ansatz. Durch die Erkennung von Inkonsistenzen wird außerdem ein Look-Ahead realisiert, der zum Beispiel auch die Erkennung nicht konsistent auflösbarer Threatmengen erlaubt. Aus Effizienzgründen wurde der Auswertung durch die Regeln eine Pruningphase vorangestellt. In dieser Pruningphase werden, wie bei

Constraint Satisfaction Problemen üblich, durch ein Preprocessing einige Möglichkeiten von vornherein ausgeschlossen.

Pruning. Yang verwendet drei Pruningkriterien, die nacheinander auf die Konfliktmengen $Conf_i$ angewendet werden.

- Partial Arc Consistency
  Das Arc Consistency-Kriterium ist eines der ältesten und bekanntesten Kriterien für Constraint Satisfaction Probleme. Es beruht auf der Idee, daß ein Wert, der mit allen Belegungen einer anderen Variable unverträglich ist, aus der Wertemenge gelöscht werden kann. Auf Konfliktmengen übertragen bedeutet dies also, daß Auflösungsconstraints, die mit allen Auflösungsmöglichkeiten eines anderen Threats unvereinbar sind, aus der Konfliktmenge entfernt werden können. Beim gewöhnlichen Arc Consistency Test werden jeweils zwei Konfliktmengen zusammen betrachtet, und durch mehrfache Wiederholungen die transitive Hülle gebildet. Yang zufolge ist für die Verarbeitung von Konfliktmengen die Bildung der transitiven Hülle zu ineffektiv. Er wendet deshalb nur eine partielles Arc Consistency Kriterium an, bei dem lediglich ein Durchlauf gemacht wird.

- Redundancy Pruning
  Im Unterschied zu normalen Constraint Satisfaction Problemen können Auflösungsmöglichkeiten eines Threats auch einen anderen mitauflösen. Da dadurch die Zahl getroffener Entscheidungen verringert werden kann, wird beim Redundancy Pruning überprüft, ob jede Auflösungsmöglichkeit eines Threats die Auflösung eines zweiten impliziert. Ist dies der Fall, so kann der zweite Threat bei den Berechnungen vernachlässigt werden - in der Terminologie der CSPs wird ein Netzknoten gelöscht.

- Ausschluß einzelner Werte
  Als drittes Kriterium werden einzelne Werte durch eine Kombination der Inkonsistenzrelation und der Subsumtionsrelation ausgeschlossen. Dieses Kriterium läßt sich am einfachsten mathematisch beschreiben:
  Seien $Conf_1, Conf_2$ *zwei Konflikmengen.*
  Sei $R_2 \in Conf_2$, *so daß* $\forall R_1 \in Conf_1$ *entweder*
  1.) $inconsistent(R_1, R_2)$ *oder*
  2.) $\exists R_3 \in Conf_2$, *so daß* $R_2 \neq R_3$ *und* $subsumes(R_1, R_3)$ *gilt.*

Algebraische Regeln. Nach dem Preprocessing wird jetzt nach irredundanten, konsistenten Auflösungsmöglichkeiten gesucht. Dazu werden die Konfliktmengen als algebraische Terme aufgefaßt und mittels einiger Regeln vereinfacht.

- Darstellung von Konfliktmengen als algebraische Terme
  Für die Berechnung der Auflösungskombinationen gilt, daß für jeden Threat eine Auflösungsmöglichkeit ausgewählt werden muß und alle diese Auflösungsconstraints zusammen angewandt werden. Daher bietet sich eine Und-Oder-Darstellung der Möglichkeiten an, etwa in der Form $(c_{11} + c_{12} + c_{13}) * (c_{21} + c_{22})$. Bringt man diesen Ausdruck nun in DNF, so stellt jede der Teilkonjunktionen (Kombination) eine Lösung des Gesamtproblems dar.

- Vereinfachung durch Regeln
  Mit Hilfe von Regeln werden auf diesen algebraischen Ausdrücken Vereinfachungen durchgeführt, die die Inkonsistenzrelation und die Subsumtionsrelation zwischen Threatauflösungsconstraints ausnutzen.
  Eingesetzte Regeln sind:

  1. Sind die Constraints innerhalb einer Kombination inkonsistent, so kann sie gelöscht werden.
  2. Constraints, die von den übrigen Constraints einer Kombination impliziert werden, sind überfüssig und können aus der Kombination entfernt werden.

Yang führt noch einige weitere Regeln an, deren Auswertung jedoch zu aufwendig erschien, und die deshalb nicht realisiert wurden.

Ein weiterer Unterschied zu Yang's Implementierung ist, daß dieser mittels Tiefensuche nach Kombinationen sucht. Wir haben uns für Breitensuche entschieden, da diese leichter zu implementieren war und zur Bestimmung einer minimalen Lösung ohnehin die Berechnung aller Kombinationen nötig ist. Die Regeln werden hierbei schon während des Aufbaus der Kombinationen angewandt, um den Pruningeffekt des Inkonsistenztestens frühzeitig nutzen zu können.

In der Praxis hat sich jedoch gezeigt, daß sich häufig sehr viele unaufgelöste Threats ansammeln können und daß dann dieses Verfahren der kombinatorischen Explosion erliegt. Vor Anwendung der verschiedenen Pruningstrategien wird daher die Anzahl der möglichen Kombinationen grob abgeschätzt und bei Überschreitung von 10000 Kombinationen auf die Anwendung des Verfahrens verzichtet.

Heuristik	Look Ahead und Least Constraining
Wirkung	Erkennt Unauflösbarkeit von Threatmengen und bestimmt minimale Lösung(en) zur Auflösung aller Threats
Transformation	Lösen eines CSP; vor allem Inkonsistenz- u. Subsumtionstests
Korrektheitsbedingungen	Bestimmung inkonsistenter Threatauflösungskomb. ist korrekt; die minimale Auflösung muß aber nicht die richtige sein !
Hilfsmittel	-
Komplexitätsfaktoren	Viele Threats. Viele Ordnungen bzw. viele Variablen.

4.3.4.2 Vermeidung von Separationsoperatoren

Separationsoperatoren führen mindestens zwei Constraints in den Plan ein: ein oder zwei Ordnungsconstraints, die den bedrohenden Schritt notwendigerweise parallel zum bedrohten Causal Link anordnen und ein oder mehrere Bindungsconstraints. Im Vergleich zur Demotion und Promotion, die jeweils nur einen Bindungsconstraint einführen, werden durch Separation also wesentlich mehr Constraints in den Plan eingeführt. Gemäß der Least-Constraining-Heuristik sollten daher die Demotion und Promotion bevorzugt werden.

Heuristik	Least Constraining
Wirkung	Vermeidet Einschränkungen
Transformation	-
Korrektheitsbedingungen	-
Hilfsmittel	-
Komplexitätsfaktoren	-

Kapitel 5

Synthese von Kontrollkomponenten

Nachdem nun eine Vielzahl von Methoden vorgestellt worden ist, beschäftigt sich dieses Kapitel mit der Synthese von Kontrollverfahren aus diesen Grundbausteinen und zwar sowohl lokal zur Steuerung eines einzelnen Kontrollpunktes, als auch global.
Neben der Frage nach der syntaktischen Lösung der Integration sollen auch semantischen Aspekte wie z.B. synergistische und antagonistische Effekte untersucht werden. Local Variables:

5.1 Lokale Integration

Der menschliche Entscheidungsfindungsprozeß ist gekennzeichnet durch ein sorgfältiges Abwägen verschiedener Einflußfaktoren. Die genaue Vorgehensweise bleibt der Introspektion jedoch weitgehend entzogen. Während wir nun gar nicht erst versuchen wollen, diesen menschlichen Entscheidungsprozeß nachzubilden, stehen wir dennoch vor dem Problem, aus den Aussagen mehrerer Kontrollmethoden eine sinnvolle Gesamtentscheidung synthetisieren zu müssen. Zunächst machen wir uns aber noch einmal bewußt, von welcher Art die 'Eingaben' der gesuchten Verknüpfungsmethode sind.

Wie bereits in Abschnitt 4.1.3 näher ausgeführt, liefern die vorgestellten Kontrollheuristiken einen der folgenden Ergebnistypen:

1. Bisektion
2. Multisektion bzw. schwache Ordnung
3. Partialordnung

Wie man sich leicht überzeugen kann, lassen sich jedoch alle drei Typen auch als Spezialfall der jeweils anderen Typen interpretieren:

Filtersicht
Bisektionsmethoden lassen sich als Filter interpretieren, die die gewünschten Kandidaten von den unerwünschten trennen. (Cinderella-Prinzip, vgl. auch: 'Das Sieb des Eratosthenes'). Aber auch die beiden anderen Ergebnistypen werden letztendlich in dieser Betrachtungsweise verwendet: Ausgewählt werden jeweils Mitglieder der minimalen Klasse bzw. der minimalen Schicht.

Klassifikationssicht
Man kann die Ergebnistypen auch als Klassifikationsmethoden betrachten; bei der Bisektion gibt es genau 2 Klassen, bei der Partialordnung nur künstliche Klassen, die Schichten.

Ordnungssicht
Betrachtet man die Ergebnisse bezüglich ihres ordnenden Charakters, so stellt die Bisektion den Spezialfall einer sehr flachen Ordnung dar. Bei der Multisektion ist der Ordnungsaspekt durch die schwache Ordnung, d.h. die Ordnung auf den Klassen gegeben.

Trotz dieser gegenseitigen Verflechtungen macht eine Unterscheidung dieser Sichtweisen jedoch Sinn, weil sie uns zu verschiedenen Ansätzen bei der Verknüpfung der Methoden führt.

5.1.1 Verknüpfungsmethoden

Abhängig von der Betrachtungsweise der Ergebnisse bieten sich verschiedene Verknüpfungsmethoden an:

Ordnungssicht: Amalgamationsfunktional
Ordnungsrelationen werden gewöhnlich durch gewichtete Summen oder allgemeiner durch Amalgamationsfunktionale verknüpft. Gehören alle der zu verknüpfenden Ordnungsrelationen zu den schwachen Ordnungen (d.h. Ergebnistyp Bisektion und Multisektion), so läßt sich zeigen, daß kein Amalgamationsfunktional gefunden werden kann [RW90], das die folgenden drei naheliegenden Forderungen erfüllt:

- **Demokratieprinzip:** Ordnen alle der zu verknüpfenden Ordnungsrelationen ein Element vor ein anderes, so soll dies auch für die Ordnung gelten, die Ergebnis des Amalgamationsfunktionals ist.
- **Vermeidung von Diktatorfunktionen:** die entstehende Gesamtordnung soll nicht durch eine einzelne Ordnungsrelation dominiert werden
- die Anordnung zweier Elemente durch die entstehende Gesamtordnung soll unabhängig von den übrigen Alternativen sein.

Dieser *Unmöglichkeitssatz von Arrow* läßt sich wegen der Verwendung von Partialordnungen (die selbst schon Bedingung 3 verletzen) nicht direkt auf das hier vorliegende Problem anwenden; wir können aber festhalten, daß das Finden eines sinnvollen Amalgamationsfunktionals oder das Finden geeigneter Gewichte schwieriger ist, als man dies intuitiv erwarten würde. Selbst unter der Annahme, daß es möglich wäre, ein Amalgamationsfunktional zu bestimmen, das zu akzeptablen Gesamtbewertungen führt, hätte diese Methode aber einen gravierenden Nachteil: Zur Bestimmung des globalen Minimums entsprechend der Gesamtbewertung müßte jede Kontrollmethode auf die gesamte Konfliktmenge angewandt werden – dies ist in Hinsicht auf die Effizienz der Gesamtmethode unannehmbar.

Multisektionssicht: Entscheidungsbäume
Betrachtet man die Kontrollverfahren unter dem Aspekt, daß sie die Menge der Alternativen in mehrere Klassen einteilen, so bieten sich Entscheidungsbäume als Verknüpfungsmöglichkeit an. Entscheidungsbäume würden eine ganz feine Staffelung des Entscheidungsprozesses erlauben und insbesondere eine unterschiedliche Behandlung verschiedener Konfliktmengentypen ermöglichen. Es scheint jedoch relativ aufwendig, die Definition eines solchen Entscheidungsbaumes benutzerfreundlich durch eine entsprechende Oberfläche zu unterstützen und den Aufbau dieser Bäume übersichtlich zu gestalten.

Bisektionssicht: gestaffeltes Filtern
Eine weitere, naheliegende Verknüpfungsmethode ist die Hintereinanderschaltung von Filterkriterien; ein Verfahren, das schon in der Antike von Eratosthenes zur effizienten Bestimmung von Primzahlen eingesetzt worden ist: Mehrere Methoden, die die Menge der Alternativen verringern, werden einfach konkateniert. Das gestaffelte Filtern stellt einen Spezialfall eines Amalgamationsfunktionals dar, bei dem das Diktatorprinzip bewußt verletzt wird, indem man explizit eine Dominanzordnung auf den Filtermethoden festlegt. Diese Vorgehensweise läßt sich auch als lexikographische Verknüpfung von Ordnungrelationen auffassen.

Im Gegensatz zu Ordnungsamalgamationsfunktionalen werden bei der Filtersicht jedoch nicht alle Methoden auf die gesamte Konfliktmenge angewandt. Stattdessen werden nachgeordnete Kontrollmethoden nur für die noch nicht zurückgewiesenen Alternativen ausgewertet.

Im folgenden werden wir den Ansatz des gestaffelten Filterns näher erläutern.

5.1.2 Gestaffeltes Filtern

Standardisierung der Schnittstellen. Um die freie Kombinierbarkeit der verschiedenen Methoden zu ermöglichen, wurde eine Standardisierung der Schnittstellen notwendig. Diese Standardisierung kann einerseits gleich bei der Entwicklung einer Kontrollmethode mitberücksichtigt werden oder aber dadurch, daß für Methoden mit anderem Schnittstellenformat Hüllfunktionen geschrieben werden, die wie ein Schnittstellenadapter eingesetzt werden.

Eingabeschnittstelle
Die Eingabeschnittstelle der Zielauswahlmethoden wurde auf die Übergabe der Konfliktmenge beschränkt, die der Operatormethoden auf die Konfliktmenge und das zu reduzierende Ziel. Die Entscheidung, ob gewöhnliche Ziele oder Protektionsziele als nächstes bearbeitet werden, soll auf Grundlage der beiden Zielmengen getroffen werden. Schließlich wurden Templates für die Funktionsnamen vorgegeben, denen alle Methoden bzw. deren Hüllfunktionen genügen müssen.

Standardisierung der Funktionalität
Die Methoden sollen jeweils davon ausgehen können, daß ihnen nichtleere Konfliktmengen übergeben werden. Sie müssen daher auch ihrerseits garantieren, nichtleere Mengen ausgewählter Alternativen zurückzugeben. Sollten durch Anwendung der Heuristik an sich alle Alternativen ausgeschlossen sein, so soll lediglich die Eingabemenge an die Ausgabeschnittstelle weitergeleitet werden.

Auswahl- und Filtermodus. Da letztlich nur eine Alternative ausgewählt wird, genügt es bei Auswertung der letzten Methode einer Kombination, den ersten Vertreter der gewünschten Klasse zurückzugeben ohne die restlichen Alternativen auszuwerten. Dies macht die Auswertung von Bisektionsmethoden billiger; bei den anderen zwei Ergebnistypen ist die beste Klasse nicht von vornherein eindeutig und man kommt nicht umhin, die Methoden doch für alle Alternativen zu berechnen.
Um die effektivere Auswertung zumindest für die Bisektionsmethoden zu ermöglichen, wird der Eingabeschnittstelle ein Parameter hinzugefügt, in dem der Auswertungsmodus angegeben wird.

Benutzeroberfläche. Um die Konfiguration einer Filterkombination möglichst benutzerfreundlich zu gestalten, wurde eine komfortable Benutzeroberfäche geschaffen (s. Abb. 5.1 und 5.2). Nachdem

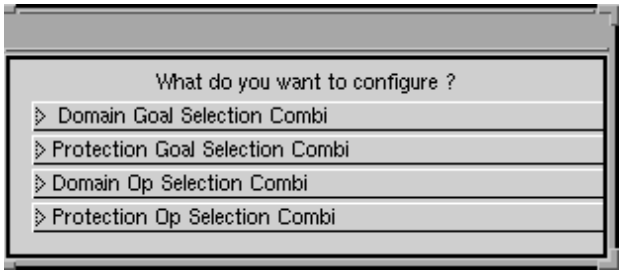

Abbildung 5.1: Auswahl des Kontrollpunktes

der Benutzer über ein Menü den Kontrollpunkt ausgewählt hat, für den er eine neue Kombination zusammenstellen möchte, gelangt er in ein Konfigurationsmenü. Die Reihenfolge der Methoden läßt sich mittels einer Swap-Funktion bequem modifizieren. Nachdem die Reihenfolge der Filter eingestellt worden ist, lassen sich die gewünschten Methoden mit der Maus auswählen. Schließlich wird der Benutzer nach dem Namen der neuen Kombination gefragt, und automatisch der Kode dafür generiert. Der Kodegenerierungsvorgang entspricht einem Precompiling des Kodetemplates in Abbildung 5.3.

Bedeutung der Filteranordnung. Die Anordnung der Filtermethoden beeinflußt sowohl Kosten als auch Ergebnis der Auswertung:

Kosten der Auswertung
Die Auswertungskosten der verschiedenen Methoden sind unterschiedlich hoch. Um die Gesamtkosten

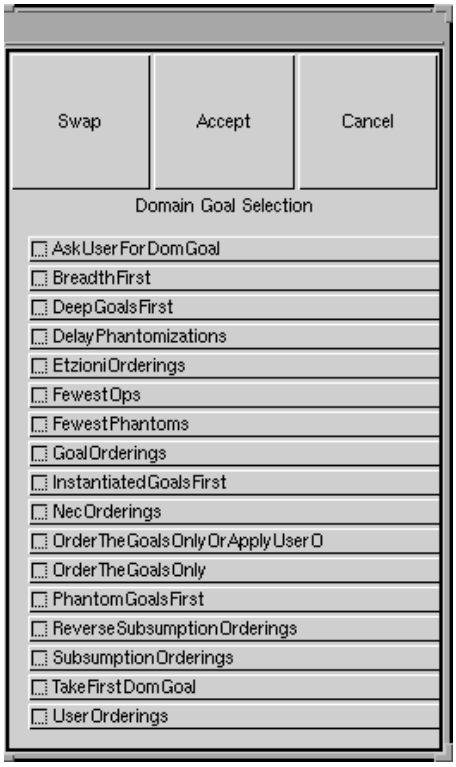

Abbildung 5.2: Beispiel eines Konfigurationsmenüs

```
index := 1
while index < NoOfFilters do
    if size of conflict set = 1, return it.
    else
        apply methodArray[index] in Filter-Mode.
        index := index + 1.
apply methodArray[NoOfFilters] in Detect-Mode.
return conflict set.
```

Abbildung 5.3: Kodetemplate für Kodegenerierung

zu minimieren, sollten die teueren Methoden auf möglichst kleine Konfliktmengen angewandt werden. Dies läßt sich durch zwei Strategien erreichen: 1.) Anordnung stark filternder Methoden zu Beginn der Kombination, 2.) Anordnung teuerer Methoden am Ende der Kombination. Offensichtlich liefern diese zwei Strategien häufig widersprüchliche Aussagen und müssen sorgfältig gegeneinander abgewogen werden.

Beeinflussung des Auswertungsergebnisses
Ist die Menge der Alternativen, die alle Filter passieren, nicht leer, so ist das Filtern äquivalent zu einer konjunktiven Verknüpfung der Methoden und das Ergebnis unabhängig von der Reihenfolge der Filteranordnung. Ist diese Menge jedoch leer, so wird durch die Reihenfolge der Filter eine Relevanzordnung der Kriterien ausgedrückt. Um ein möglichst korrektes Ergebnis zu erhalten, muß daher versucht werden, die Methoden entsprechend der Sicherheit ihrer Ergebnisse zu staffeln.

Zusammenfassend läßt sich also sagen, daß der Benutzer bei der Entscheidung der Filterreihenfolge die Filtermächtigkeit, die Kosten und die Wichtigkeit bzw. Korrektheit der verschiedenen Methoden gegeneinander abwägen muß.

Integration anderer Kontrollmöglichkeiten. Offensichtlich erlaubt die Kapselung des eigentlichen Kontrollverfahrens auch eine einfache Integration anderer Kontrollverfahren.

So wurde etwa für jeden der Kontrollpunkte auch eine Methode realisiert, die den Benutzer um eine Auswahl bittet. Wird diese Benutzeranfrage als Abschluß einer Kombination ausgewählt, so wird der Benutzer immer dann um Rat gefragt, wenn die ausgewählten Kontrollverfahren nicht ausreichen, um eine eindeutige Entscheidung zu fällen. Dies erlaubt eine gezielte Benutzerinteraktion und könnte auch Ansatzpunkt für einen Apprenticeship-Ansatz sein.

Eine andere Möglichkeit wäre, die Auswertung gelernter Regeln ebenfalls mit einer standarisierten Hüllfunktion zu versehen und so die Regeln an beliebiger Stelle im Entscheidungsprozeß auszuwerten.

Diskussion. Für den Einsatz des gestaffelten Filterns spricht, daß das Gesamtverfahren durch die schrittweise Einschränkung der Konfliktmenge wesentlich effizienter wird. Vorteilhaft sind auch die starke Modularität und die transparente Entscheidungsfindung, durch die die Benutzerfreundlichkeit erhöht wird. Nachteilig ist, daß durch die Beschränkung auf die Filtersicht eine Dominanzordnung der Kontrollverfahren angegeben werden muß. Dies führt zu Abstrichen bezüglich der Modellierbarkeit:

- Auch für an sich gleichwertige Kontrollverfahren muß eine Anordnung angegeben werden
- Bei der Filterverknüpfung werden die Kontrollverfahren uniform auf die gesamte Menge der ihnen übergebenen Alternativen angewandt. Soll eine bestimmte Gruppe von Alternativen jedoch unterschiedlich behandelt werden bzw. ist eine Kontrollmethode nur auf Alternativen mit bestimmten Eigenschaften anwendbar, so müssen diese Gruppen innerhalb jeder dieser Kontrollmethoden neu bestimmt werden.

Verknüpfungsmöglichkeiten aus der Literatur. Aus der Literatur sind speziell für Kontrollmethoden noch zwei weitere Vorgehensweisen bekannt:

1. Kombination von EBL-Regeln
   Minton [Min88] erzeugt Selektions-, Ausschluß- und Präferenzregeln. Diese werden in der gegebenen Reihenfolge ausgewertet. Faßt man die Selektions- und Ausschlußregeln als Bisektionsverfahren sowie die Präferenzregeln als Partialordnungsverfahren auf, so läßt sich dieses Verfahren sehr schön in das allgemeinere Verfahren des gestaffelten Filterns einbetten.

2. Logische Kombination von Kontrollverfahren
   Dengler beschreibt in [Den95] ein System, das ebenfalls die individuelle Konfiguration von Kontrollverfahren erlaubt. Die Eingabemethoden für die Verknüpfungsmethoden beschränken sich allerdings nicht auf reine Auswahlmethoden, sondern umfassen z.B. auch Methoden, die dazu dienen, die Informationsgrundlage zu vergrößern. Für solche 'tactics' stehen die folgenden Verknüpfungsmöglichkeiten zur Verfügung:

   (a) $\prec$ – die Präferenz: die Methoden werden in der gegebenen Reihenfolge durchprobiert, bis eine Methode anwendbar ist.

   (b) ; – die Hintereinanderausführung (z.B. von informationseinholenden und auswählenden tactics)

   (c) $\wedge$ – die Konjunktion: nur solche Alternativen werden ausgewählt, die allen Kriterien genügen.

   (d) $\vee$ – die Disjunktion: indeterministisch wird eine Methode ausgewählt und ausgewertet. Schlägt sie fehl, so wird eine andere ausprobiert.

   Für das vorliegende Problem ist die Hintereinanderausführung irrelevant, da informationseinholende Aktionen direkt in die einzelnen Kontrollverfahren integriert sind. Auch die Disjunktion scheint nicht sehr sinnvoll, da für reale Maschinen die Klauseln einer Disjunktion immer in einer festen Reihenfolge getestet werden, wodurch die Disjunktion äquivalent zur Präferenz wird.

   Schließlich läßt sich das oben beschriebene Filterkonzept als eine Kombination der Präferenzverknüpfung und der Konjunktion darstellen: Wie schon erläutert, sind die Filtermethoden alle

so realisiert, daß sie niemals eine leere Menge von Alternativen zurückgeben, sondern im Falle der Zurückweisung aller Alternativen die Eingabemenge unverändert an die nächste Stufe weiterleiten. Fehlschläge werden also wie bei der Präferenzverknüpfung behandelt; wie bei der Konjunktion werden aber mehrere Kontrollkriterien für die Entscheidungsfindung kombiniert.

5.1.3 Konfiguration von Kontrollkomponenten - ein Toolbox-Ansatz

Nachdem wir uns im vorangegangenen Abschnitt näher mit der lokalen Integration von Kontrollmethoden beschäftigt haben, wenden wir uns nun der Komposition einer Kontrollkomponente aus Filterkombinationen zu.

Zusammenstellung einer Kontrollkomponente durch den Benutzer. Da die Kontrollpunkte an sich feststehen, gestaltet sich der Kompositionsvorgang denkbar einfach: Der Benutzer bekommt eine Übersicht der schon generierten Kombinationen für die verschiedenen Kontrollpunkte präsentiert und kann dann durch Auswahl je einer Filterkombination eine Kontrollkomponente zusammenstellen (vgl. Abb. 5.4). Nachdem der Benutzer einen Namen für die neue Kontrollkomponente angegeben hat, wird eine entsprechende Komponente kompiliert. Diese Komponente kann dann sofort für die Planung ausgewählt werden.

Hilfsmittel Aus den Klassifikationstabellen in Kapitel 3 ist ersichtlich, daß die meisten Kontrollmethoden einige Hilfsmittel benötigen. Diese Hilfsmittel müssen nun bei Selektion der Kontrollkomponente und/oder vor Beginn eines jeden Planungsvorganges initialisiert werden. Um auch den zur Initialisierung benötigten Kode automatisch generieren zu können, wurde folgendes Verfahren gewählt:

- Jedes Hilfmittel stellt für jeden der zwei Zeitpunkte eine Initialisierungsroutine mit generischem Namen zur Verfügung. Falls für einen der zwei Zeitpunkte keine Initialisierung benötigt wird, bleibt der Funktionsrumpf leer.
- Für jede Kontrollmethode muß der Programmierer in einer Tabelle die Namen der benötigten Hilfsmittel eintragen.
- Nachdem der Benutzer eine Filterkombination zusammengestellt hat, wird die Menge der für diese Kombination benötigten Hilfsmittel berechnet und gespeichert.
- Bei Zusammenstellung der Kontrollkomponente wird aus diesen Einträgen nun wiederum die Menge der benötigten Hilfsmittel bestimmt. Aus den Initialisierungsroutinen der einzelnen Hilfsmittel werden dann die Initialisierungsmethoden für die Kontrollkomponente zusammengesetzt.

Durch diese Vorgehensweise beschränkt sich die Konfigurierung von Filterkombinationen und Kontrollkomponenten durch den Benutzer rein auf das Auswählen. Dies ermöglicht selbst programmierunerfahrenen Benutzern das Zusammenstellen neuer Kontrollkomponenten. Ein weiterer Vorteil ist die Reduzierung der Fehlerwahrscheinlichkeit, da vom Programmierer nur die Kontrollmethoden geschrieben werden müssen und der übrige Kode automatisch erzeugt wird.

Grundgerüst der Kontrollkomponenten Der Aufruf der jeweiligen Filterkombinationen muß in ein Grundgerüst eingebettet sein, das eine Basisfunktionalität garantiert. Dazu gehören:

- Behandlung leerer Konfliktmengen
  Da alle Kontrollmethoden annehmen, daß ihre Eingabemenge nicht leer ist, wird dieser Ausnahmefall durch das Grundgerüst abgefangen.
- Abfangen von Zielblockaden
  Da Zielblockaden Backtracking unvermeidlich machen, werden blockierte Ziele sofort bearbeitet.[1]

[1] Blockierte Phantomziele werden nicht sofort ausgewählt, weil für sie im Laufe der Planung noch Phantomisierungsmöglichkeiten hinzukommen können, die die Blockade aufheben.

Abbildung 5.4: Das Kompositionsmenü

- Sind entweder nur normale Ziele oder nur Protektionsziele vorhanden, so wird die Methode zur Entscheidung, welcher Zieltyp behandelt werden soll, nicht ausgewertet, sondern gleich die entwprechende Filterkombination aufgerufen.

5.2 Semantische Effekte

Obwohl bei der Besprechung der einzelnen Kontrollmethoden vereinzelt schon Hinweise darauf gegeben worden sind, wie verschiedene Methoden zusammenwirken, sollen hier synergistische und antagonistische Effekte nochmals explizit aufgegriffen werden, weil die Kenntnis solcher Interdependenzen das Zusammenstellen effektiver und effizienter Kontrollkomponenten erleichtert.

5.2.1 Synergistische Effekte

Zunächst einmal ist festzuhalten, daß sich prinzipiell alle korrekt entscheidenden Kontrollmethoden gegenseitig positiv beeinflussen. Dies liegt daran, daß jede Kontrollmethode nur *relativ zu ihrer Aufrufumgebung* korrekt entscheidet; implizit nimmt also jede Kontrollmethode an, daß alle vorherigen Kontrollentscheidungen richtig waren.

Auch die praktische Erfahrung hat gezeigt, daß es nicht genügt, die Entscheidungsfindung an einem einzigen Kontrollpunkt zu verbessern, sondern daß vielmehr für alle in der jeweiligen Domäne kritischen Entscheidungspunkte gute Verfahren benötigt werden, um zu einem akzeptablen Planungsverhalten zu gelangen. Grund hierfür ist, daß falsche Entscheidungen nur durch Backtracking (oder Benutzerinteraktion) revidiert werden können; die Größe des Suchraumes schließt aber in den meisten Fällen aus, daß der Planer *in praktikabler Zeit* zur Stelle des tatsächlichen Fehlers zurückfindet. Abgesehen von diesen grundsätzlichen Überlegungen gibt es jedoch durchaus Methoden, die direkt die Korrektheit anderer Methoden beeinflussen, indem sie deren implizite Korrektheitsbedingungen (vgl. Klassifikationstabellen) gültig oder zumindest sicherer machen.

Zustand. Die wichtigste Korrektheitsbedingung, die von anderen Kontrollmethoden beeinflußbar ist, ist die Annahme, daß der verallgemeinerte Zustand, der zum Planungszeitpunkt berechnet wird, der Menge der zum Ausführungszeitpunkt zur Verfügung stehenden Zustandsliteralen entspricht. Dazu müssen zwei Anforderungen erfüllt sein:

Vollständigkeit
Damit der verallgemeinerte Zustand alle relevanten Literale enthält, müssen zum Planungszeitpunkt schon alle Schritte im Plan enthalten sein, die bei einer tatsächlichen Ausführung vor der Stelle im Plan ausgeführt werden, für die der verallgemeinerte Zustand gerade berechnet werden soll. Dies kann dadurch erreicht werden, daß mit Hilfe der Zielauswahlkontrolle die Planungsreihenfolge die Ausführungsreihenfolge approximiert.

Korrektheit
Um zu vermeiden, daß der verallgemeinerte Zustand zuviele überfüssige Literale enthält, sollte der Plan so linear wie möglich sein. Deshalb sollten sichere Phantomisierungen und Threatauflösungen nicht unnötig aufgeschoben werden. Auch die in Abschnitt 4.2.2 beschriebenen notwendigen Schrittanordnung tragen zur Linearisierung der Pläne bei.

Aus den Wirkungen und den Korrektheitsbedingungen der einzelnen Kontrollmethoden läßt sich nun ableiten, welche Methoden die Zustandsinformationen sicherer machen bzw. welche Methoden darauf aufbauen. Bei Verwendung einer der Methoden der Liste 5.1 sollten also ein oder mehrere Methoden der Liste 5.2 ebenfalls in die Kontrollkomponente mitaufgenommen werden.

Annahmen über den Interaktionsgrad. Einige Heuristiken sind nur dann korrekt, wenn die Ziele serialisierbar sind. Es ist daher sinnvoll, diesen Methoden Filtermethoden voranzustellen, die Interaktionen entdecken können.

Minimum Conspiracy
Operatorfehlschlagsbedingungen
Vermeidung von phantomisierbaren Zielen
implizit alle Methoden zur Phantomisierungsauswahl[2]

Tabelle 5.1: Kontrollmethoden, die Zustandsinformation benötigen

Notwendige Zielanordnungen
Zielsubsumptionsordnung
Bevorzugung tiefgelegener Ziele
Bevorzugung von Phantomzielen
Bevorzugung von Phantomisierungen
Notwendige Schrittanordnungen

Tabelle 5.2: Kontrollmethoden, die Zustandsinformation sicherer machen

Notwendige Zielanordnungen
Bevorzugung tiefgelegener Ziele
Bevorzugung von Phantomzielen
Bevorzugung von Phantomisierungen
Minimum Conspiracy

Tabelle 5.3: Kontrollmethoden, die Interaktionsarmut annehmen

Notwendige Zielanordnungen(ganz generelle Konflikte)
Zielsubsumtionsordnung
Etzioni's Zielanordnungen
Bevorzugung NO-konsistenter Domänenop.
Vermeidung inkonsistenter Phantomisierungen

Tabelle 5.4: Kontrollmethoden, die Interaktionen berechnen

Auflösbarkeit von Threats. Wie schon in Abschnitt 3.3.3 erwähnt, bleibt die Systematik des Algorithmus auch bei Änderung des Threatauflösungszeitpunktes erhalten, falls sichergestellt ist, daß alle noch nicht aufgelösten Threats konsistent auflösbar sind. Daher ist eine Kombination von Yang's Inkonsistenztest mit Methoden, die den Threatauflösungszeitpunkt bestimmen, sinnvoll. Weil der Inkonsistenztest allein schon durch die Bestimmung der Konfliktmengen teuer ist, lohnt sich der zusätzliche Aufwand jedoch nur in Domänen mit geringer Lösungsdichte.

5.2.2 Antagonistische Effekte

Widersprüchliche Verfahren. Dem Planungsproblem liegt ein ganz inherenter Konflikt zugrunde: einerseits wird versucht, sich möglichst wenig festzulegen, um keine Fehler zu machen. Auf der anderen Seite wird aber auch versucht m öglichst viel festzulegen, um Sackgassen frühzeitig erkennen zu können und um für die Entscheidungsfindung eine möglichst breite und sichere Informationsgrundlage zur Verfügung zu haben.

Alle Methoden, die diese zwei gegensätzlichen Strategien verfolgen, widersprechen sich in gewissem Sinne. Diese Widersprüchlichkeit läßt sich aber sehr gut nutzen, um fein abgestufte Kompromisse

zu realisieren. Der Basiskompromiß besteht immer darin, möglichst viele sichere Entscheidungen zu treffen und möglichst viele unsichere hinauszuzögern.

Vermeidung nicht vollständig instantiierten Ziele
Vermeidung von phantomisierbaren Zielen
Least Commitment Threatauflösung
Vermeidung von Separation

Tabelle 5.5: Kontrollmethoden, die Entscheidungen hinauszögern

Notwendige Schrittanordnungen
Bevorzugung von Phantomisierungen
Vermeidung von phantom. Zielen
(= Bevorzugung von zu planenden Zielen)
Bevorzugung tiefgelegener Ziele
Bevorzugung von Phantomzielen
Bevorzugung von Invarianten
Algebraische Konfliktauflösung

Tabelle 5.6: Kontrollmethoden, die die Informationsgrundlage vergrößern

Neben diesen semantischen Abhängigkeiten der verschiedenen Methoden gibt es noch eine Reihe weiterer Kriterien, die bei der Zusammenstellung einer Kontrollkomponente berücksichtigt werden müssen. Das folgende Kapitel ist daher dem konkreten Konfigurieren von Kontrollkomponenten gewidmet.

5.3 Gezielte Komposition von Kontrollkomponenten

Dieses Kapitel ist der Praxis gewidmet; hier soll Hilfestellung für das konkrete Zusammenstellen von Kontrollkomponenten gegeben werden. Ein erster Abschnitt beschäftigt sich mit mit der Identifikation domänenspezifischer Schwierigkeiten und Charakteristika und Möglichkeiten, diese bei der Kontrollkonfiguration zu berücksichtigen. In einem zweiten Abschnitt wird nochmals auf das Utility-Problem eingegangen, da es sich als eine der Hauptschwierigkeiten herausgestellt hat. Die Anwendung der zuvor gegebenen Hinweise wird in einem letzten Abschnitt anhand dreier Beispieldomänen demonstriert.

5.3.1 Identifikation und Berücksichtigung domänenspezifischer Charakteristika

Vor der Entscheidung für eine Kontrollmethode müssen drei Faktoren überprüft werden:

1. Gewinnpotential
2. Anwendbarkeit
3. Anwendungskosten

Hier wollen wir uns zunächst mit den zwei erstgenannten Punkten beschäftigen, die Diskussion der Anwendungskosten wird im folgenden Abschnitt behandelt.

5.3.1.1 Gewinnpotential

Ein Kontrollverfahren sollte nur dann verwendet werden, wenn es zumindest das Potential besitzt, den Entscheidungsprozeß positiv beeinflussen zu können. Die meisten Kontrollverfahren dienen dazu,

einen ganz spezifischen Konflikttyp zu erkennen und zu vermeiden oder vorteilhafte Situationen zu bemerken und auszunutzen. Ein Kontrollverfahren kann demnach die Entscheidungsfindung nur dann positiv beeinflussen, wenn die Domäne das Phänomen aufweist, auf dessen Erkennung die Methode ausgelegt ist.
Beispiel:

> Die Methode der notwendigen Zielanordnungen berechnet Anordnungen von Zielen, die in jedem Plan zur Erreichung beider Ziele gelten. In der Blocksworld können beispielsweise Türme nur von unten nach oben gebaut werden.
> Es gibt aber Domänen, für die keine solchen Ordnungsstrukturen existieren. So gibt es etwa in der Transportation Domain keine *notwendige* Reihenfolge, in der einzelne Päckchen befördert werden müssen. Der Versuch, in solchen Domänen notwendige Zielanordnungen bestimmen zu wollen, muß daher fruchtlos bleiben.

Im folgenden werden für alle Methoden die zugrundeliegenden Phänomene aufgeführt, sowie Möglichkeiten, deren Existenz in einer Domäne zu überprüfen. Ist keine Methode zur Erkennung eines Phänomens angegeben, so läßt sich dessen Auftreten meist durch simples Betrachten der Domänendefinition entscheiden.

1. **Notwendige Zielanordnungen**
   - Phänomen: Es gibt Ziele, die nur in einer festgelegten Reihenfolge erzeugt werden können.
   - Erkennung: Da die notwendigen Zielanordnungen in einer Domänenanalysephase berechnet und später nur die kompilierten Ergebnisse verwendet werden, ist es am einfachsten, die Domänenanalyse durchzuführen und nachzuschauen, ob notwendige Zielanordnungen gefunden werden konnten.

2. **Zielsubsumtion**
   - Phänomen: 1. Es gibt Probleme, bei denen Toplevel-Ziele [3]in einer Teilzielbeziehung stehen. Oder:
     2. Toplevel-Ziele können gemeinsame Unterziele haben
   - Erkennung: Auch die Zielsubsumtionen werden in der Domänenanalysephase berechnet und erlauben so eine einfache Überprüfung.

3. **Vermeidung unvollständig instantiierter Ziel**
   - Phänomen: Es gibt Operatoren, die Variablen enthalten, die nicht durch Matchen des Operators mit einem Ziel gebunden werden.
   - Erkennung: -

4. **Vermeidung phantomisierbarer Ziele**
   - Phänomen: Die Wahrscheinlichkeit falscher Phantomisierungen ist in Domänen mit großem Anfangszustand und häufiger Erzeugung und Zerstörung von Zielen besonders hoch. Solche Domänen sind für den Einsatz dieser Heuristik geradezu prädestiniert.
   - Erkennung: Gibt es viele negative Interaktionen in der Domäne ?
     Kann der Anfangszustand ein gesuchtes Ziel schon enthalten ?
     Werden Ziele häufig durch Seiteneffekte erzeugt ?

5. **Bevorzugung tiefer Ziele**
   - Phänomen: -
   - Erkennung: -

6. **Zielanordnung nach Etzioni**
   - Phänomen: Auftreten negativer und positiver Interaktionen.[4]
     Es *gibt* eine falsche Reihenfolge, Ziele zu bearbeiten.
   - Erkennung: Die Anordnungen nach Etzioni werden in der Domänenanalysephase berechnet. Dies erlaubt wiederum, eine Existenz der berücksichtigten Interaktionstypen durch Ausrechnen nachzuweisen.

[3] einzelne Ziele der Zielkonjunktion eines Problems

[4] Treten nur einige Interaktionstypen auf, so empfiehlt sich, die jeweiligen Einzelmethoden statt der Standardkombination zu verwenden

7. **Bevorzugung von Phantomzielen**
   - Phänomen: Abhängig von der Domänenmodellierung: sollte nur dann eingesetzt werden, falls diejenigen Vorbedingungen als Phantomziele markiert worden sind, die zur Instantiierung des Operators dienen.
   - Erkennung: -

8. **Bevorzugung eindeutig auflösbarer Threats**
   - Phänomen: -
   - Erkennung: -

9. **Vermeidung separierbarer Threats**
   - Phänomen: Separierbare Threats sollen wegen der Größe ihrer Konfliktmenge vermieden werden. Diese Heuristik ist daher besonders wichtig in Domänen, in denen Prädikate mit hoher Arität verwendet werden.
   - Erkennung: -

10. **Fehlschlagsbedingungen**
    - Phänomen: Es gibt für ein Ziel mehrere Domänenoperatoralternativen, die unterschiedlich geeignet sein können, d.h. es besteht überhaupt die Möglichkeit, einen falschen Operator auszuwählen.
    - Erkennung: Die Fehlschlagsbedingungen werden ebenfalls in der Domänenanalysephase berechnet und können dann auf Existenz (und Relevanz) überprüft werden.

11. **Vermeidung NO-inkonsistenter Domänenoperatoren**
    - Phänomen: -
    - Erkennung: -

12. **Vermeidung inkonsistenter Phantomisierungen**
    - Phänomen: -
    - Erkennung: -

13. **Gewichtete Minimum Conspiracy Heuristik**
    - Phänomen: Diese Heuristik bestimmt den Operator, der am besten zur Situation paßt, d.h. für den die meisten Vorbedingungen schon erfüllt sind. Gibt es aber keinen eigentlichen Anfangszustand und treten wenige Seiteneffekte auf, so gibt es gar keine 'passenden' Operatoren.
    - Erkennung: -

14. **Bevorzugung von Haupteffekten**
    - Phänomen: Nur sinnvoll, wenn bei der Modellierung in Haupt- und Seiteneffekte unterschieden worden ist.
    - Erkennung: -

15. **Bevorzugung von Invarianten**
    - Phänomen: Anwendbar auf Domänen, in denen es Invarianten gibt.
    - Erkennung: -

16. **Konfliktauflösung nach Yang**
    - Phänomen: Viele Interaktionen
    - Erkennung: Am einfachsten werden ein paar Beispielprobleme gelöst und die Größe der Metaagenda beobachtet. Außerdem sollte durch Beobachten der Backtrackinghäufigkeit beurteilt werden, ob die Threatauflösung wirklich kritisch ist.

5.3.1.2 Anwendbarkeit

Ein zweiter ausschlaggebender Punkt ist die Frage, ob eine Methode für eine Domäne überhaupt berechenbar ist. Eine ganze Reihe der vorgestellten Verfahren verwendet explizit oder implizit (durch Verwendung der PSGs) Axiome. Ist eine Domäne aber nicht axiomierbar, so können all diese Verfahren keine Aussagen treffen oder dies nur in abgeschwächter Form tun. Der Einsatz axiombasierter Verfahren für nicht axiomatisierbare Domänen sollte daher vermieden werden.

Die folgenden Verfahren verwenden implizit oder explizit Axiome:

1. Notwendige Zielanordnungen
2. Zielanordnungen nach Etzioni
3. Fehlschlagsbedingungen
4. NO-inkonsistente Operatoren
5. Inkonsistente Phantomisierungen

Außer den Axiomen konnten für die vorgestellten Verfahren keine Faktoren gefunden werden, die eine Berechnung verhindert hätten.

5.3.2 Das Utility-Problem

Der Einsatz von Kontrollverfahren ist nur dann sinnvoll, wenn der erzielte Nutzen die Kosten aufwiegt. Mit zu den schwierigsten Aufgaben bei der Zusammenstellung einer Kontrollkomponente gehört die Abschätzung dieses Kosten/Nutzen- Verhältnisses, das sogenannten Utility-Problem [Min88].

Minton, der dieses Problem für den Bereich der Planung als erster thematisiert hat, löste das Utility-Problem für seine EBL-Regeln dadurch, daß er bei Erzeugung einer Regel die Anzahl der Knoten berechnet hat, die er im vorliegenden Beispiel bei vorheriger Kenntnis der Regel hätte sparen können. Ferner schätzt er die Anwendbarkeitshäufigkeit ab und setzt diese zwei Faktoren in Verhältnis zu den Anwendungskosten [5]. Diese Nützlichkeitsabschätzungen werden regelmäßig aktualisiert und Regeln mit negativer Nützlichkeit aus dem Kontrollregelfundus entfernt.

Gratch und DeJong [GD90] haben darauf hingewiesen, daß die Nützlichkeit von Kontrollwissen aber immer nur relativ zu zusätzlich vorhandenem Kontrollwissen zu sehen ist (das sogenannte 'Composability Problem'). Insbesondere läßt sich nur schwer von der Nützlichkeit einer isoliert eingesetzten Regel auf deren Nützlichkeit in Verbindung mit anderen Regeln schließen.

Eines der Hauptprobleme ist, daß sich Kontrollregeln oder -verfahren häufig in ihren Aussagen überschneiden. Die Auswertung mehrerer Verfahren erhöht dann nur die Kosten, ohne die Menge der Alternativen weiter einschränken zu können. Dieses Problem wird natürlich um so kritischer, je korrekter die verschiedenen Methoden arbeiten. Ein Ausweg ist, stets darauf zu achten, nur solche Verfahren zu kombinieren, die auf möglichst unterschiedlichen Kriterien basieren. Diese These wird auch durch die Analyse erfolgreicher integrierter Kontrollsysteme wie etwa [MP92, KME91] bestätigt, die sehr verschiedenen Verfahren kombinieren. Die Anwendungskosten sind allerdings selbst ohne Überschneidungen ein kritischer Faktor, weil grundsätzlich nur ein bestimmter Anteil der Planungsarbeit einsparbar ist. Dieses begrenzte Einsparpotential erlaubt daher auch nur ein gewisses Maß an Analysearbeit, das zwischen den Verfahren aufzuteilen ist. Eine positive Beeinflussung der Nützlichkeit ist hingegen zu erwarten, wenn man allgemeinen Regeln solche Verfahren voranstellt, die deren Ausnahmen erkennen können, weil sich solche Kombinationen gerade in ihren Aussagen ergänzen.

Trotz dieser Schwierigkeiten, die Nützlichkeit von Verfahren a priori abzuschätzen, wurde aber keine eigene Komponente zur Verwaltung der Nützlichkeit der verschiedenen Kontrollverfahren entwickelt, da eine Kontrollkomponente nur einmal konfiguriert wird, und nicht wie bei regelbasierten, lernenden

[5] Diese Funktion läßt sich auch als Nutzenfunktion im Sinne der allgemeinen Nutzentheorie auffassen, die in ihren Grundzügen auf von Neumann und Morgenstern zurückgeht.

Systemen ständig neue Kontrollregeln hinzukommen. Stattdessen wurde versucht, durch Strukturanalysen den Konfigurierer mit so viel Information auszustatten, daß ihm die Entwicklung effizienter und effektiver Kontrollkomponenten auch durch Auswerten dieser Angaben möglich ist.

Obwohl in den Klassifikationstabellen der einzelnen Methoden schon jeweils Faktoren angegeben worden sind, deren Zunahme eine Kostensteigerung impliziert, sollen im folgenden nochmals einige generelle Hinweise zu kostenkritischen Punkten gegeben werden.

1. Separates Austesten
   Trotz den genannten Vorbehalten kann das separate Austesten kostenintensiver Kontrollverfahren für eine Domäne wichtige Aufschlüsse über das domänenspezifische Kostenverhalten liefern.

   Beispiel 1:
   Eine typische Fehlschlagsbedingung in der Nilsson-Blocksworld [Nil80] umfaßt etwa 2 Literale mit jeweils 2-3 Constraints, die es zu matchen gilt. Einige der Fehlschlagsbedingungen in der Werkstückdomäne umfassen hingegen mehr als 40 Konjunktionen von jeweils rund 5 Literalen mit jeweils etwa 4 Constraints.[6] Das Matchen solcher Fehlschlagsbedingungen ist zu teuer, um jemals zu einer Verbesserung des Laufzeitverhaltens führen.

   Beispiel 2:
   Für einige Zielauswahlverfahren muß für jedes Ziel der Konfliktmenge die Menge der Reduktionsoperatoren berechnet werden. In der Blocksworld-Domäne fällt dies nicht ins Gewicht, da die Anzahl der Ziel auf der Agenda gering bleibt. In der Werkstückdomäne hingegen werden sehr viele Ziele erzeugt und die Berechnung der Konfliktmengen für die Ziele dominiert die Kosten des gesamten Kontrollprozesses.

2. Für die Verfahren, die in einer einmaligen Domänenanalyse berechnet werden, lassen sich die dynamischen Anwendungskosten gut durch Betrachten der während des Planungsvorganges noch zu instantiierenden Informationen abschätzen.

3. Unter bestimmten Bedingungen erweist sich auch die Anwendung von Axiomen zur Laufzeit als Kostenfaktor, weil beim Anwenden von Axiomen Matchkosten entstehen und die Konsistenz von Constraints überprüft werden muß. Besonders die Überprüfung der Constraints wird aber mit Fortschreiten des Planungsprozesses immer teurer, weil die Constraintmenge im Plan (außer bei Backtracking) stetig größer wird. Verfahren, die explizit, d.h. zur Laufzeit Axiome auswerten, sind deshalb in Domänen kritisch, in denen viel Information durch Constraints modelliert wird.

4. Beim Einsatz von Methoden, die die Bearbeitung von phantomisierbaren Domänenzielen oder Threats verzögern, muß immer mitbedacht werden, daß dies auch eine Verkleinerung der Konfliktmenge verhindert: Die Bearbeitung eines phantomisierbaren Zieles oder eines Threats entfernt dieses Ziel aus der Konfliktmenge und führt meistens nicht zur Erzeugung neuer Threats. Hingegen führt die Bearbeitung eines Zieles durch Einführung eines Operators zu neuen Zielen und vergrößert in aller Regel die Konfliktmenge. Eine Vergrößerung der Konfliktmenge führt aber auch zu einer Kostensteigerung, weil bei späteren Auswahlentscheidungen die Ausgangsmenge größer ist, für die die verschiedenen Kriterien ausgewertet werden müssen. Dies wird insbesondere bei Domänen mit vielen Zielen zu einem kritischen Faktor.

5.3.3 Zusammenfassung aller zu berücksichtigender Faktoren

Hier soll nochmals kurz zusammengefaßt werden, welche Faktoren bei der Zusammenstellung einer Kontrollkomponente eine Rolle spielen, und wie sich daraus eine generelle Strategie ableiten läßt.

Die Konfiguration einer Kontrollkomponente teilt sich prinzipiell in zwei Phasen auf:

[6] Die Fehlschlagsbedingungen werden so groß, weil in dieser Domäne viele nicht-erplanbaren Vorbedingungen auftreten. Das Verfahren zur Berechnung bestimmt nun alle Bedingungen, unter denen das Fehlen solch einer Bedingung in der Initialsituation zu einem Fehlschlag führt. Für den Planungsvorgang wird aber immer angenommen, daß alle dieser unplanbaren Bedingungen vorhanden sind. Ein Fehlschlag aufgrund fehlender unplanbarer Bedingungen kommt daher gar nicht vor, was die Regeln zusätzlich völlig aussagelos macht.

1. In einer ersten Phase wird die Menge aller Verfahren bestimmt, die prinzipiell relevant sind für die vorliegenden Domäne. In dieser Phase müssen also für jedes Verfahren das Gewinnpotential und die Anwendbarkeit bestimmt werden und evtl. durch Experimente abgesichert werden.

2. In der zweiten Phase muß dann versucht werden, für jeden Entscheidungspunkt eine Teilmenge der relevanten Verfahren auszuwählen und solchermaßen anzuordnen, daß sich das Laufzeitverhalten des Planers durch Einsatz der Gesamtkomponente verbessert.

Bei der Anordnung der Verfahren gilt zu berücksichtigen,

1. daß kostenintensive Verfahren nur auf relativ kleine Konfliktmengen angewandt werden sollten und deshalb weit hinten angeordnet werden müssen

2. daß Verfahren mit starker Filterwirkung möglichst früh angewandt werden sollten, um die Konfliktmenge schnell zu reduzieren

3. daß korrektere Verfahren zu Anfang einer Kombination stehen sollten, da sie im Falle widersprüchlicher Aussagen ausschlaggebend sein werden

4. daß allgemeinen Verfahren Verfahren vorangestellt werden sollten, die deren Ausnahmen berechnen

5. daß nicht zuviele aufwendige Verfahren zusammen verwandt werden dürfen

6. und schließlich, daß vermieden werden sollte, Verfahren zu kombinieren, die ganz ähnliche Aussagen machen.

5.3.4 Demonstration anhand von Beispieldomänen

In diesem Abschnitt soll anhand einiger Beispieldomänen aufgezeigt werden, wie mit Hilfe der Toolbox effiziente Kontrollkomponenten konfiguriert werden können. Darüberhinaus soll noch einmal das Utility-Problem und das Composability-Problem aufgezeigt werden. Die Spezifikation der Domänen findet sich im Anhang.

5.3.4.1 Die Drehteildomäne

Wir beginnen damit, die **Anwendbarkeit** der Verfahren zu überprüfen. Zunächst fällt auf, daß es für diese große Domäne nur sehr wenige Axiome gibt. Dies läßt vermuten, daß axiombasierte Verfahren für diese Domäne nicht sehr gut geeignet sind. Dieser Verdacht läßt sich schnell anhand der statischen Berechnungen bestätigen: Für einige Verfahren wie etwa den notwendigen Zielanordnungen war die Axiomatisierung der Domäne zu schwach, um überhaupt Kontrollpräferenzen zu finden. Für andere Verfahren wie die Operatorfehlschlagsbedingungen können zwar Präferenzen berechnet werden, jedoch sind diese unbrauchbar, weil die Anwendbarkeitsbedingungen mangels Axiomen nicht wirkungsvoll beschnitten werden konnten und so keine effiziente Auswertung zur Laufzeit erlauben. Aus ähnlichen Gründen entfernen wir auch die Zielanordnungen nach Etzioni, den NO-Inkonsistenztest für Operatoren und den Konsistenztest für Phantomisierungen aus der Menge möglicher Kontrollverfahren.

Im nächsten Schritt überprüfen wir das **Gewinnpotential** der verbleibenden Methoden nach den in Abschnitt 6.1 angegebenen Kriterien. Hinter dem Namen jeder Methode ist die Bewertung angegeben; diese wird im nachfolgenden Text jeweils kurz erläutert.

1. Zielsubsumtion: ?
   Durch die statische Berechnung werden nur sehr wenige Subsumtionsbeziehungen gefunden. Die Nützlichkeit dieser Methode muß daher empirisch überprüft werden.

2. Vermeidung unvollständig instantiierter Ziele: +
   Die meisten Operatoren werden durch Matchen mit einem Ziel nur unvollständig instantiiert. Meist ist etwa die Kontur oder die Spannkontur noch nicht gebunden.

3. Vermeidung phantomisierbarer Ziele: ?
   In dieser Domäne gibt es keinen Anfangszustand im eigentlichen Sinne und daher besteht auch nicht die Gefahr falscher Phantomisierungen mit dem Anfangszustand. Da Interaktionen durch Aufspannoperationen und Werkzeugwechsel aber die eigentlichen Schwierigkeiten der Domäne darstellen, muß die Nützlichkeit dieser Methode in der Praxis überprüft werden.

4. Bevorzugung tiefer Ziele: ?
   Basiert auf keinem speziellen Phänomen und muß daher empirisch überprüft werden.

5. Bevorzugung von Phantomzielen: +
   In dieser Domäne sind diejenigen Vorbedingungen als Phantomvorbedingungen modelliert worden, die zur Variablenbindung dienen. Beispiel: Der Operator `FertigeFeatureAllg` besitzt die zwei Phantomvorbedingungen `(+ unterbereich feature kont)` `(+ drehmeisselLinks wzg)`. Die Prädikate `unterbereich` und `drehmeisselLinks` können nicht durch Operatoranwendungen erzeugt werden, sondern treten lediglich als (eindeutige) Invarianten auf. Man kann also diese Phantomvorbedingungen bedenkenlos an die Invarianten binden und instantiiert so die Variablen `wzg` und `kont`.Daher ist es sinnvoll, diese Ziele anderen gegenüber zu bevorzugen.

6. Bevorzugung eindeutig auflösbarer Threats: ?
   beruht auf keinem speziellen Phänomen.

7. Vermeidung separierbarer Threats: ?
   Es gibt in dieser Domäne zwar Prädikate mit hoher Arität, die an Konflikten beteiligten Prädikate `gespanntAuf` und `wzgEingelegt` haben aber lediglich Arität 1. Somit erhöht der Einsatz der Separation den Branchingfaktor nur unwesentlich. Es ist unklar, ob es von Vorteil ist, den Planer zur Anordnung von Schritten zu zwingen, oder ob eine Konfliktaufösung durch alternative Bindung von Variablen nicht auch sinnvoll sein kann. Wir werden dies in der Praxis überprüfen.

8. Minimum Conspiracy Heuristik: ?
   Die Pläne in dieser Domäne werden sehr groß, insbesondere gibt es sehr viele Phantomisierungsmöglichkeiten für häufig auftretende Prädikate wie `gespanntAuf` und `wzgEingelegt`. Daher ist es sehr wahrscheinlich, daß die zur Auswahl stehenden Operatoren gleich gut zur Anwendungssituation passen. Wir müssen daher auch für diese Methode testen, ob sie sich in dieser Domäne bewährt.

9. Bevorzugung von Haupteffekten: -
   In der Drehteildomäne sind Ziele immer nur ausschließlich durch Haupteffekte oder ausschließlich durch Seiteneffekte erplanbar. Somit kann nie zwischen einem Haupteffekt-Operator und einem Seiteneffekt-Operator ausgewählt werden, und diese Methode würde in der Praxis nie zur Anwendung kommen.

10. Bevorzugung von Invarianten: -
    Diejenigen Prädikate, die als Invarianten auftreten, sind nicht durch Operatoranwendungen erplanbar. Bei der Operatorauswahl für solch ein Ziel kann man also ohnehin nur unter mehreren Invarianten auswählen, was eine Bevorzugung überflüssig macht.

11. Konfliktauflösung nach Yang: +
    Da die konsistente Konfliktauflösung eines der Hauptprobleme dieser Domäne darstellt, erscheint diese Methode besonders vielversprechend.

In einem nächsten Schritt versuchen wir für jeden der lokalen Entscheidungspunkte eine sinnvolle **Auswahl und Anordnung** der noch verbliebenen Methoden zu finden.

Auswahl von Domänenzielen. Nach der Vorauswahl sind noch die folgenden Methoden übriggeblieben:

1. die Vermeidung unvollständig instantiierter Ziele,

2. die Bevorzugung tiefer Ziele, und

3. die Bevorzugung von Phantomzielen

Da dies billige Methoden sind, müssen wir diese Menge nicht weiter einschränken und suchen nun nach einer Anordnung. Diese ergibt sich sofort aus folgenden Überlegungen:

Durch die Bevorzugung von Phantomzielen werden Ziele instantiiert. Dies macht den Einsatz der ersten Methode erst richtig sinnvoll und legt daher eine Anordnung von Methode 1 hinter Methode 3 nahe. Die Bevorzugung tiefer Ziele bringt weniger offensichtlich Vorteile als die beiden anderen Methoden und wird daher erst nach den zwei anderen Methoden eingesetzt.

Wir erhalten also folgende Kombination:

```
selectByWP1domgoalFrom: aConflictSet
   |filteredConflictSet|

   filteredConflictSet := aConflictSet.
   (filteredConflictSet size = 1)
         ifTrue: [ ^filteredConflictSet]
         ifFalse: [ filteredConflictSet := self applyPhantomGoalsFirstTo: filteredConflictSet mode: #filter.].
   (filteredConflictSet size = 1)
         ifTrue: [ ^filteredConflictSet]
         ifFalse: [ filteredConflictSet := self applyInstantiatedGoalsFirstTo: filteredConflictSet mode: #filter.].
   (filteredConflictSet size = 1)
         ifTrue: [ ^filteredConflictSet]
         ifFalse: [ ^self applyDeepGoalsFirstTo: filteredConflictSet mode: #detect.
```

Auswahl von Protektionszielen. Da die Methode von Yang sowohl die Auswahl der Protektionsoperatoren als auch die Auswahl der Protektionsziele steuert, entscheiden wir uns hier für die Auswahl durch Yang. Da die Methode von Yang nur Berechnungen anstellt, wenn keines der Ziele eindeutig aufzulösen ist, ist implizit die Bevorzugung eindeutig auflösbarer Threats dieser Methode vorangestellt.

Auswahl von Domänenoperatoren. Nach der Vorauswahl sind für diesen Entscheidungspunkt nur noch die Bevorzugung von Phantomisierungsoperatoren und die Minimum Conspiracy Heuristik übrig, deren Nutzen jedoch noch zu überprüfen ist. Wegen der höheren Kosten wenden wir die Minimum Conspiracy Heuristik nach der Bevorzugung von Phantomisierungsoperatoren an.

Auswahl von Protektionsoperatoren. Wie schon erwähnt, räumen wir der Konfliktauflösung nach Yang die größten Chancen ein und beschränken uns auf diese Methode.

Entscheidung zwischen Domänenzielen und Protektionszielen. Es ist sehr schwierig, für diesen Entscheidungspunkt Effizienzvoraussagen zu machen. Wir müssen daher die verschiedenen Möglichkeiten empirisch austesten und beginnen mit der Methode 'Bevorzugung von eindeutig auflösbaren Threats'.

Verwendung von Feinarchitektur-verändernden Methoden. Da die Methode der notwendigen Schrittanordnungen auf der Anwendung von Axiomen basiert, scheint ihr Einsatz in dieser Domäne wenig sinnvoll. Auch für die PMU hat sich gezeigt, daß die Berechnung aufgrund der riesigen Pläne in dieser Domäne zu kostspielig ist.

Eine erste Auswertung. Wir nennen diese Kontrollkomponente WP1[7] und vergleichen deren Ergebnisse mit der Standardkontrolle SNLP+. Abbildung 5.5 zeigt den empirischen Vergleich der Laufzeiten für eine Testmenge von 5 Problemen.

[7] WP steht für WorkPieces

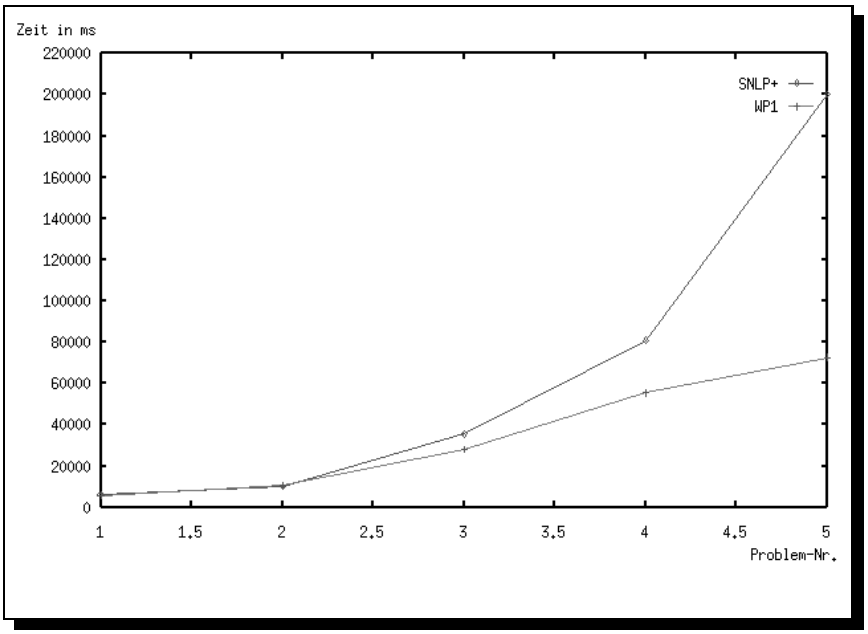

Abbildung 5.5: Ergebnisse der Kontrollkomponente WP1.

Wie wir diesen Graphen entnehmen können, bewirkt der Einsatz der Kontrolle, daß größere Probleme etwa doppelt so schnell gelöst werden können wie mit der Standardkontrolle. Bei zu kleinen Problemen ergibt sich allerdings aufgrund des Utility-Problems kein Vorteil.

Betrachtet man aber außer den Laufzeiten auch die übrigen statistischen Daten (siehe Tabelle 5.7) und insbesondere die Zahl der benötigten Inferenzschritte (siehe Abbildung 5.6), so zeigt sich sehr deut-

	SNLP+		WP1	
Ø(Zeit)	66594 ms	+/- 59133.66 ms	38867.3 ms	+/- 24830.6 ms
Ø(# Inferenzschritte)	221.6	+/- 197.83	64.4	+/- 23.766
Ø(Planlänge)	20.5	+/- 7.89	13.6	+/- 3.720

Tabelle 5.7: Statistische Ergebnisse der Standardkontrolle SNLP+ und der Kontrollkomponente WP1

lich, daß der Planer mit der zusätzlichen Kontrolle wesentlich weniger sucht und signifikant kürzere Pläne erzeugt. Ein Großteil der eingesparten Zeit wird allerdings durch den zusätzlichen Berechnungsaufwand wieder aufgebraucht, so daß sich die Leistungsverbesserung nicht so prägnant auf die Laufzeit auswirkt.

Um zu demonstrieren, wie stark sich selbst kleine Änderungen der Kontrollkomponente auswirken können, betrachten wir nun noch die Ergebnisse von zwei plausibel scheinenden Varianten: Bei der ersten (noPreferDeepGoals) wurde auf die Bevorzugung tiefgelegener Ziele bei der Domänenzielauswahl verzichtet, bei der zweiten (noMinConspiracy) verzichtete man auf die Minimum Conspiracy Heuristik (siehe Abb. 5.7). Es ist ganz klar ersichtlich, daß jede dieser Methoden ganz wesentlich zum guten Ergebnis der Gesamtkomponente beigetragen hat. Die außergewöhnlich starken Auswirkungen von kleinen Änderungen verdeutlichen aber auch das Composability-Problem, also das Problem, daß aus dem Zusammenspiel der verschiedenen Methoden ganz unerwartete positive wie negative Effekte entstehen können.

5.3.4.2 Die Blocksworlddomäne mit 2 Operatoren

Wir beginnen wieder mit einer **Eignungsprüfung** der Verfahren für diese Domäne. Im Gegensatz zur Drehteildomäne ist die Blocksworlddomäne gut axiomatisierbar und somit sind alle Kontrollmethoden

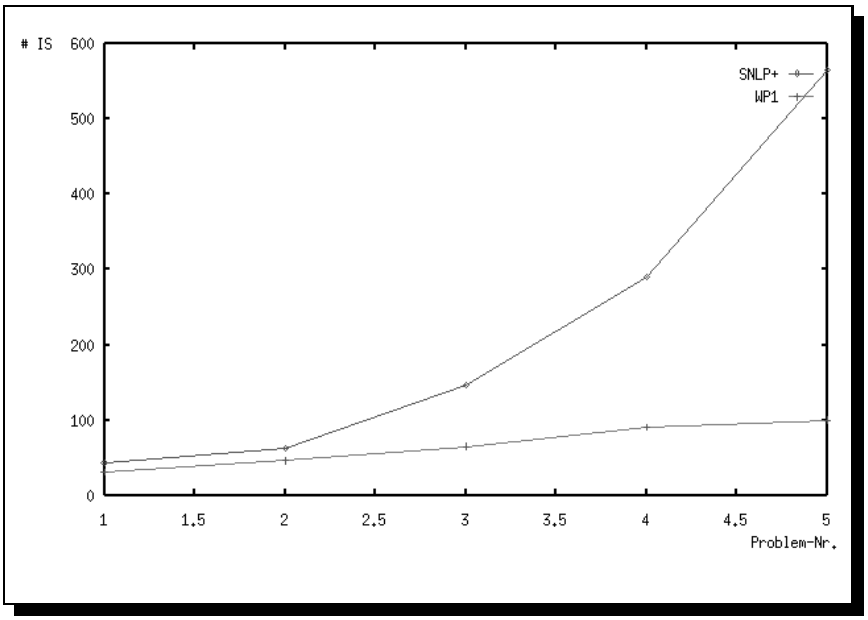

Abbildung 5.6: Ein Vergleich der benötigten Inferenzschritte für die Standardkontrolle und die Kontrollkomponente WP1.

prinzipiell anwendbar.

Als nächstes wird wieder jede Kontrollmethode auf ihr **Gewinnpotential** in dieser Domäne untersucht.

1. Notwendige Zielanordnungen: +
   In dieser Domäne treten viele Interaktionen auf und es gibt auch für einige Ziele eine feste Reihenfolge, in der sie gelöst werden müssen. Ein Blick auf die Ergebnisse der statischen Berechnungen bestätigt diesen ersten Eindruck.

2. Zielsubsumtion: +
   Die Ergebnisse der statischen Berechnungen zeigen, daß es in dieser Domäne auch Zielsubsumtionen gibt.

3. Vermeidung unvollständig instantiierter Ziele: +
   Scheint sinnvoll, da es Operatoren gibt, die durch das Matchen mit einem Ziel noch nicht vollständig instantiiert sind.

4. Vermeidung phantomisierbarer Ziele: +
   Die Blocksworlddomäne ist eine der Domänen, in dem ein Teilziel schon im Anfangszustand gelten kann, während der Planung aber wieder aufgrund von Zielinteraktionen zerstört wird. Vorschnelle Phantomisierungen mit dem Anfangszustand können daher zu tiefem Backtracking führen und die Anwendung dieser Methode erscheint vielversprechend.

5. Bevorzugung tiefer Ziele: ?
   Basiert auf keinem speziellen Phänomen und muß daher empirisch überprüft werden.

6. Zielanordnungen nach Etzioni: -
   Beim Betrachten der statisch berechneten Anordnungspräferenzen fällt auf, daß diese eine Untermenge der von der Methode der notwendigen Zielanordnungen berechneten Präferenzen sind. Im Gegensatz zur letzteren muß vor Anwendung solch einer statisch berechneten Präferenz jedoch eine Bedingungskonjunktion mit einem erweiterten Zustand gematcht werden. Man kann daher erwarten, daß die Zielanordnungspräferenzen nach Etzioni wesentlich höhere Anwendungskosten haben werden als die der notwendigen Zielanordnungen.

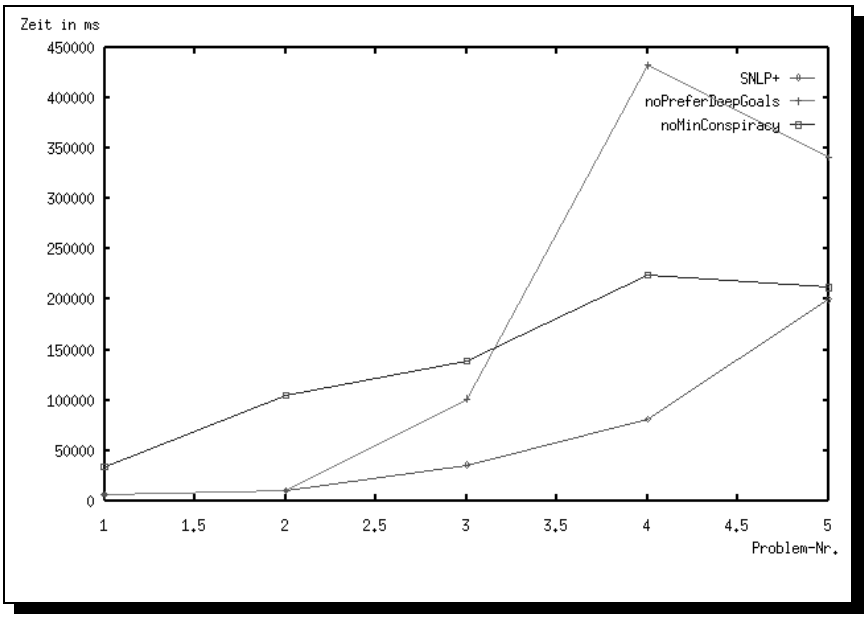

Abbildung 5.7: Ergebnisse von Kontrollkomponenten, bei denen versuchsweise auf die Bevorzugung tiefgelegener Ziele bzw. auf die Minimum Conspiracy Heuristik verzichtet worden ist.

7. Bevorzugung von Phantomzielen: ?
   In dieser Domäne dienen die Phantomvorbedingungen eher als Anwendbarkeitsbedingungen zur Operatorauswahl. Eine Bevorzugung von Phantomzielen ist daher von der Modellierung der Domäne her nicht unbedingt rechtfertigbar und muß daher empirisch überprüft werden.

8. Bevorzugung eindeutig auflösbarer Threats: ?
   beruht auf keinem speziellen Phänomen.

9. Vermeidung separierbarer Threats: ?
   In dieser Domäne haben alle Prädikate eine niedrige Arität. Wir werden den Nutzen dieser Heuristik aber dennoch empirisch überprüfen, weil sie sich in der Literatur so gut bewährt hat.

10. Fehlschlagsbedingungen nach Etzioni: +
    Ein Blick auf die Ergebnisse der statischen Berechnungen zeigt, daß in dieser Domäne Fehlschlagsbedingungen berechnet werden konnten und daß die Bedingungsausdrücke von moderater Größe sind. Wir vermuten daher einen positiven Effekt auf den Planungsvorgang.

11. Vermeidung NO-inkonsistenter Domänenoperatoren: +
    Da die Domäne gut axiomatisiert ist, müßte diese Methode relativ gute Ergebnisse liefern unter der Einschränkung, daß sie wegen ihrer hohen Kosten erst spät im Entscheidungsprozess eingesetzt wird.

12. Minimum Conspiracy Heuristik: +
    In dieser Domäne tritt das Phänomen auf, daß Operatoren mehr oder weniger gut zu einer Situation passen. Diese Methode ist daher vielversprechend.

13. Bevorzugung von Haupteffekten: +
    Einige Ziele in dieser Domäne können sowohl durch Haupt- als auch durch Seiteneffekte von Operatoren erzeugt werden. Der Einsatz dieser Heuristik scheint daher sinnvoll.

14. Bevorzugung von Invarianten: +
    Einzige Invariante in dieser Domäne ist für gewöhnlich `(+ Clear t)` . Da es eine gute Strategie ist, Klötzchen auf den Tisch zu stellen anstatt auf andere freie Klötzchen, müßte diese Methode dazu beitragen, Konflikte zu vermeiden.

15. Konfliktauflösung nach Yang: ?
    In dieser Domäne treten viele Konflikte auf. Beobachtet man jedoch den Planer (unter der Standardkontrollkomponente SNLP+), so fällt auf, daß meistens durch das Auflösen eines Threats sehr viele andere wegfallen. Der Effekt 'sparsamen' Threatauflösens, denn man sich von der Konfliktauflösung nach Yang verspricht, tritt hier also in natürlicher Weise auf. Es bleibt zu überprüfen, ob die algebraische Konfliktauflösung zusätzliche Vorteile bieten kann.

In der Vorauswahl konnte also keine der Kontrollmethoden ausgeschlossen werden. Daher muß nun die **Auswahl und Anordnung** besonders sorgfältig erfolgen. Man beginnt zunächst damit, nur die eindeutig vorteilhaften Methoden einzusetzen und die so entstandenen Komponenten so weit wie möglich auszudünnen. Auf der Basis einer solchen fast-optimalen Lösung werden dann die Methoden zweifelhafter Güte nacheinander getestet.

Auswahl von Domänenzielen. Wir beginnen mit der folgenden Kombination:

1. Bevorzugung von Phantomzielen
2. Vermeidung phantomisierbarer Ziele
3. Geringste Anzahl von Phantomisierungsoperatoren
4. Notwendige Zielanordnungen
5. Bevorzugung tiefer Ziele
6. Vermeidung unvollständig instantiierter Ziele
7. Geringste Anzahl von Operatoren

Methode 3 und 7 sind bisher nicht besprochen worden, basieren aber auf einem Least Commitment Ansatz. Bis auf die Methode der notwendigen Zielanordnungen wurden also vor allem billige, bewährte Methoden ausgewählt.

Auswahl von Protektionszielen. Wir wählen auch hier zunächst eine billige Methode, die Least-Commitment-Strategie, die diejenigen Protektionsziele bevorzugt, die die geringste Anzahl von Auflösungsmöglichkeiten haben.

Auswahl von Domänenoperatoren. Für die Auswahl von Domänenoperatoren versuchen wir zunächst die folgende Kombination:

1. Bevorzugung von Phantomisierungsoperatoren
2. Bevorzugung von Invarianten
3. Least-Constraining-Heuristik für Phantomisierungsoperatoren
4. Bevorzugung konsistenter Phantomisierungsoperatoren
5. Operatorauswahl nach Etzioni
6. Bevorzugung NO-konsistenter Operatoren
7. Bevorzugung von Haupteffekten

Methoden 1 bis 4 regeln die Auswahl zwischen verschiedenen Phantomisierungsoperatoren, die wir allgemein den Domänenoperatoren vorziehen wollen. Da nach der ersten Filterfunktion nur noch Phantomisierungsoperatoren übrig sind, wurden diese Methoden den anderen vorangestellt. Wenn die Konfliktmenge allerdings keine Phantomisierungsoperatoren enthält, so muß die Operatormenge

erst die vier oberen Filter passieren, bis sie zu Filtern gelangt, die die Menge tatschlich einschränken können. Innerhalb der Phantomisierungsoperatoren stellen wir z.B. die Bevorzugung konsistenter Phantomisierungsoperatoren an das Ende der Filterkette, weil die Auswertung dieser Methode relativ teuer ist. Zur Auswahl der Operatoren vertrauen wir auf die Fehlschlagsbedingungen nach Etzioni, den Ausschluß inkonsistenter Operatoren und die Bevorzugung von Haupteffekten. Letzter wenden wir trotz der geringen Kosten nur als letzte Methode an, weil sie die geringste Plausibilität besitzt.

Auswahl von Protektionsoperatoren. Hier verwenden wir wieder als möglichst preisgünstige Methode die Vermeidung der Separation.

Entscheidung zwischen Domänenzielen und Protektionszielen. Wir entscheiden uns hier für die Bevorzugung eindeutig auflösbarer Threats.

Verwendung von Feinarchitektur-verändernden Methoden. Da die Domäne gut axiomatisiert ist, scheint die Verwendung der notwendigen Schrittanordnungen sinnvoll. Die PMU hat sich leider auch in dieser Domäne als zu kostenintensiv erwiesen.

Empirische Auswertung

Wir nennen diese Kontrollkomponente BW1 und vergleichen deren Ergebnisse mit der Standardkontrolle SNLP+. Abbildung 5.8 zeigt den empirischen Vergleich für eine Testmenge von 50 Problemen bei einer Zeitschranke von 100 Sekunden pro Problem.

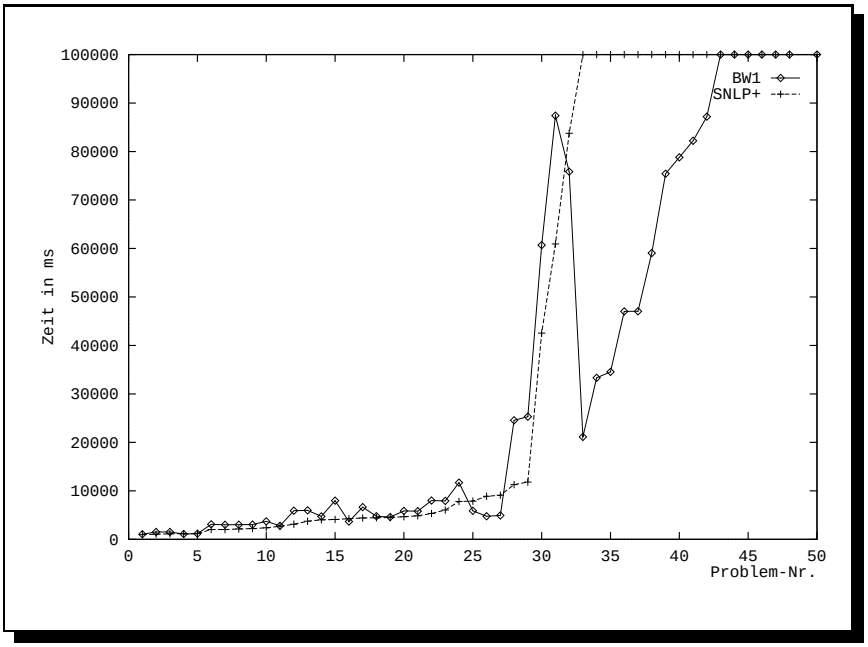

Abbildung 5.8: Ergebnisse der Kontrollkomponente BW1. Die Probleme sind aufsteigend nach den Laufzeiten der Standardkontrolle geordnet.

Wie im Abschnitt zum Utility-Problem etwas näher erläutert, gilt bei der Kontrolle des Planprozesses oft, daß 'weniger mehr ist'. Wir testen daher Varianten der Domänenzielkomposition, bei denen einige Methoden weggelassen werden. Die folgende, leicht abgespeckte Version (BW2) erweist sich dann als optimal:

1. Bevorzugung von Phantomzielen

	SNLP+		BW1	
Ø(Zeit)	43318.9 ms	+/- 46874.7 ms	33722.0 ms	+/- 59133.66 ms
Ø(# Inferenzschritte)	43.88	+/- 36.2	27.54	+/- 22.22
Ø(Planlänge)	5.88	+/- 3.66	5.4	+/- 3.55
ØLösungsrate	64%		86%	

Tabelle 5.8: Statistischer Vergleich der Komponenten BW1 und SNLP+

2. Vermeidung phantomisierbarer Ziele
3. Bevorzugung tiefer Ziele
4. Vermeidung unvollständig instantiierter Ziele

Abb. 5.9 zeigt die Ergebnisse: Bis auf eine Ausnahme (Problem Nr.32) ist die neue Komponente deutlich schneller als BW1. Wenn man Tabelle 5.8 mit Tabelle 5.9 vergleicht, so zeigt sich, daß die höhere Geschwindigkeit durch einen Utility-Tradeoff erkauft worden ist: die neue Komponente benötigt etwas mehr Inferenzschritte, ist aber insgesamt schneller, weil die Zeit pro Inferenzschritt reduziert werden konnte.

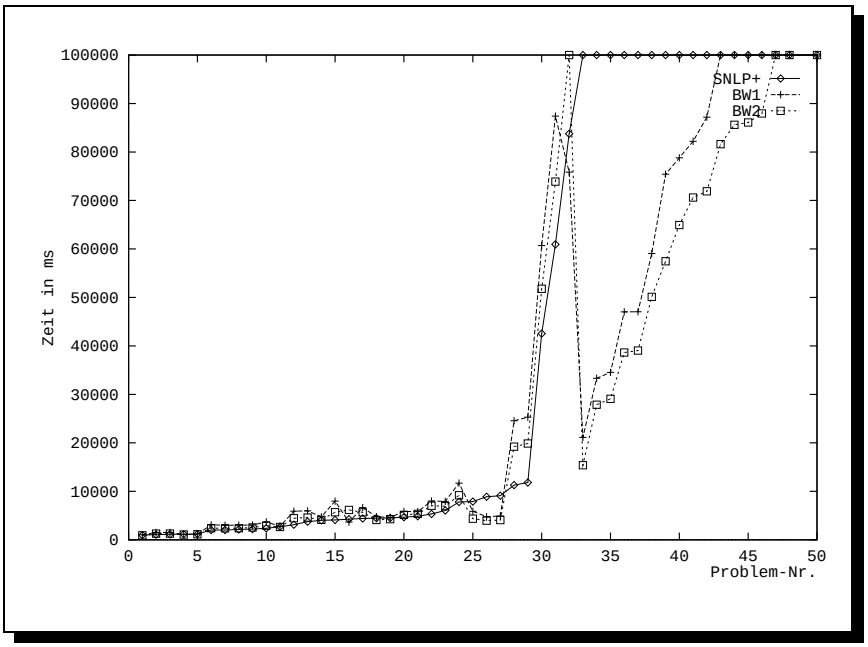

Abbildung 5.9: Ergebnisse der Komponente BW2 im Vergleich zur Standardkontrolle und zu BW1.

	SNLP+		BW2	
Ø(Zeit)	43318.9 ms	+/- 46874.7 ms	29867.4 ms	+/- 34713.1 ms
Ø(# Inferenzschritte)	43.88	+/- 36.2	28.92	+/- 24.2
Ø(Planlänge)	5.88	+/- 3.66	5.38	+/- 3.50
ØLösungsrate	64%		92%	

Tabelle 5.9: Statistischer Vergleich der Komponenten BW2 und SNLP+

Nun ersetzen wir probeweise die beiden Kombinationen zur Auswahl der Protektionsziele und Protektionsoperatoren durch die Methode von Yang. Wie schon vermutet, ist in dieser Domäne der Overhead dieser Methode jedoch zu groß, wie Abb. 5.10 zeigt.

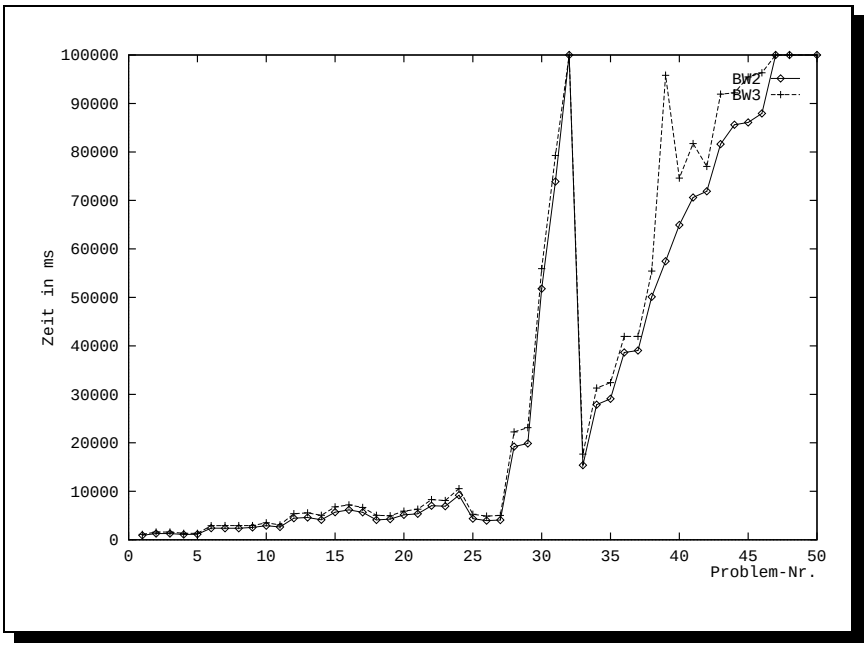

Abbildung 5.10: Die Ergebnisse der Kontrollkomponente BW3, bei der die Konflikauflösung nach Yang verwendet wird.

Wir probieren also eine weitere Variante von BW2 aus, bei der die Entscheidung zwischen der Bearbeitung von Domänenzielen und Protektionszielen immer zugunsten der Protektionsziele ausfällt. Diese Strategie muß gewählt werden, wenn man streng systematisch planen will. Wie Abb. 5.11 zeigt, wirkt sich diese Änderung jedoch sehr negativ auf das Planverhalten aus und die Lösungsrate sinkt auf nur noch 52% und damit sogar unter die der Standardkontrolle.

Zuletzt untersuchen wir noch, wie sich eine Verdoppelung des Zeitrahmens auf die Ergebnisse auswirkt. Wie Abb. 5.12 und Tabelle 5.10 belegen, nimmt der Vorsprung der Kontrollkomponente BW1 mit größerem Zeitrahmen noch zu: während mit BW1 jetzt beachtliche 94% der Probleme gelöst werden konnten, ist die Lösungsrate der Standardkontrolle mit jetzt 68%[8] immer noch sehr niedrig.

	SNLP+		BW2	
Ø(Zeit)	76835.8 ms	+/- 92839.5 ms	37923.1 ms	+/- 53135.2 ms
Ø(# Inferenzschritte)	54.35	+/- 51.61	33.5	+/- 37.3
Ø(Planlänge)	6.67	+/- 4.51	5.75	+/- 3.88
ØLösungsrate	68%		94%	

Tabelle 5.10: Statistischer Vergleich der Komponenten BW2 und SNLP+ bei verdoppelter Zeitschranke

[8]Die zusätzlichen 4% rühren davon, daß jetzt 2 Probleme mehr gelöst werden konnten, die in Abb. 5.12 auch als Zacken erkennbar sind

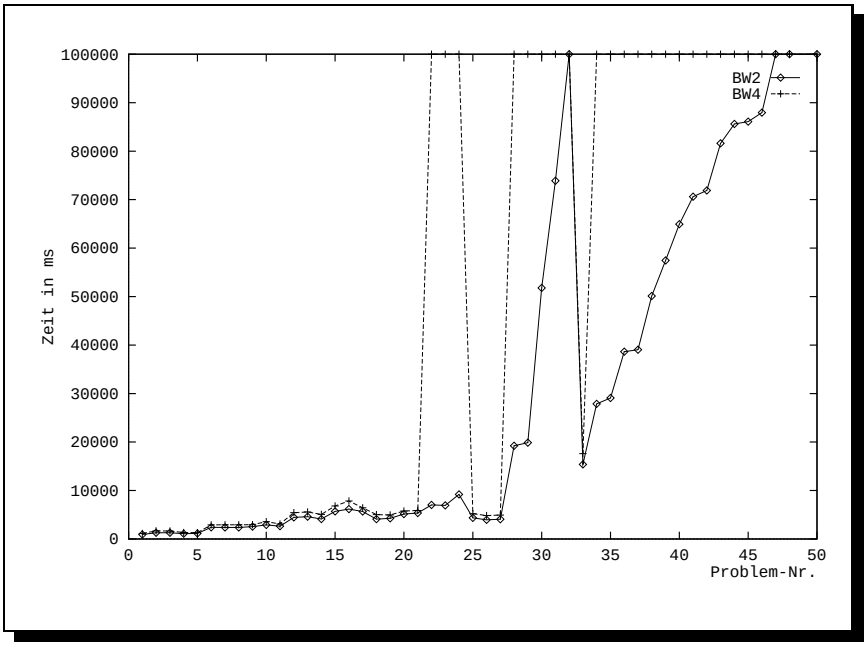

Abbildung 5.11: Die Ergebnisse der Kontrollkomponente BW4, einer Variante von BW2, bei der Protektionsziele bevorzugt bearbeitet werden.

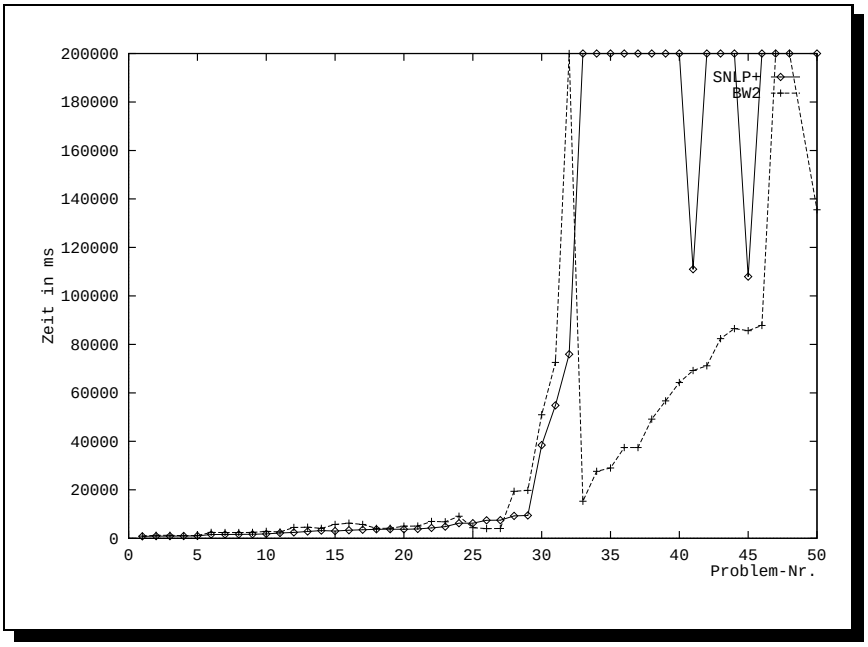

Abbildung 5.12: Ergebnisse von SNLP+ und BW2 bei verdoppelter Zeitschranke.

Kapitel 6

Zusammenfassung und Ausblick

6.1 Zusammenfassung

Mit der vorliegenden Arbeit wurde in zweifacher Hinsicht ein Grundstein für die Kontrolle des Planungssystems CAPlan gelegt.

Zum einen wurden eine ganze Reihe statischer Kontrollverfahren entwickelt und implementiert, die lernende Verfahren davon entlasten können, allgemeine Heuristiken durch einen kostspieligen Erfahrungsprozess zu 'entdecken'. Diese Verfahren wurden gut analysiert und dokumentiert, so daß eine systematische, fundierte Zusammenstellung von maßgeschneiderten Kontrollkomponenten möglich wird. Die Konfiguration wird außerdem durch eine komfortable Benutzerschnittstelle unterstützt.

Zum anderen wurde auch eine theoretische Basis gelegt und das Verständnis des Kontrollproblems in vielerlei Hinsicht vertieft. Dazu gehörten die Formulierung von Qualitätsanforderungen, die Identifikation prinzipieller Kontrollmöglichkeiten, die Entwicklung eines Klassifikations- und Bewertungsschemas für die vorgestellten Verfahren sowie eine ausführliche Diskussion der Möglichkeiten und Implikationen der Kombination mehrerer Verfahren.

6.2 Ausblick

Zuletzt noch ein kurzer Ausblick auf anstehende Arbeiten.

Um die Kontrolle noch stärker auf eine gegebene Domäne zuschneiden zu können, müssen jetzt *lernende* Kontrollverfahren hinzugefügt werden, die dem Planer ermöglichen, die *Ausnahmen* der allgemeinen Heuristiken für eine Domäne zu erlernen.

Zur Konzeption eines Assistenzsystemes würde wohl am besten eine Apprenticeship-Komponente passen, die aus Eingriffen und Änderungen eines Experten in den automatischen Planungsvorgang lernt. Solch eine Apprenticeship-Komponente wäre insbesondere auch in der Lage, die Optimierung von Plänen zu erlernen und den Planer an die Gepflogenheiten einer Domäne anzupassen.

Eine zweite naheliegende Möglichkeit sind Kontrollkomponenten, die nach Art der EBl-Verfahren durch Introspektion lernen. Für solche Verfahren sind aber sekundäre Zielkonzepte, wie etwa die Optimalität oder Natürlichkeit von Plänen schwierig zu formulieren. Eine weitere, sehr reizvolle Forschungsrichtung wäre, die Überlegungen aus Kapitel 6 zu einer fundierten Theorie auszubauen und die Zusammenstellung optimaler Kontrollkomponenten zu automatisieren, etwa mit Hilfe generischer Algorithmen.

Danksagungen

Ich möchte mich an dieser Stelle ganz herzlich bei meinen beiden Betreuern Reinhard Praeger und Prof. M.M. Richter bedanken, ferner bei Jürgen Paulokat für viele hilfreiche Erläuterungen und Vorschläge und in ganz besonderer Weise bei Frank Weberskirch, ohne dessen beständige Hilfe die Implementation einiger Verfahren sicherlich nicht möglich gewesen wäre.

Anhang A

Domänendefinitionen

A.1 Die Blocksworlddomäne

```
****Objekte*****************

CAPObjectTypeClass
        makeNewClass: Table
        definitions:  ()
CAPObjectTypeClass
        makeNewClass: Block
        definitions:  ()


****Pra"dikate**************

CAPPredicateClass
        makeNewClass: On
        definitions: (Arguments (x y ))
CAPPredicateClass
        makeNewClass: Clear
        definitions: (Arguments (x ))

****Axiome******************

CAPAxiomClass
        makeNewClass: Axiom3
        definitions: (Definition ((+ On x y ) )
                        Constraints ((IsOfType Block x ) )
                        Universal (x )
                        Arguments (x y )
                        Comment 'A block can only be on top of one block.')
CAPAxiomClass
        makeNewClass: Axiom1
        definitions: (Definition ((+ Clear x ) (+ On y x ) )
                        Constraints ((IsOfType Block x ) )
                        Universal (x )
                        Arguments (x y )
                        Comment 'Either a block is clear or another one
                                  is on top of it.')

****Operatoren***************

CAPDomainOperatorClass
```

```
        makeNewClass: PutOn
        definitions: (Arguments (x y z )
                        Constraints ((IsOfType Block x )
                                        (NotSame x y )
                                        (NotSame x z )
                                        (NotSame y z ) )
                        Subgoals ((+ Clear x ) (+ Clear y ) )
                        Comment 'PutOn-Operator: Transformation
                                x                   x
                                z    y  ===>   z    y'
                        Phantoms ((+ On x z))
                        Side Effects ((- On x z ) (+ Clear z ) (- Clear y))
                        Purposes ((+ On x y)))


CAPDomainOperatorClass
        makeNewClass: ClearTop
        definitions: (Arguments (x y z )
                        Constraints ((IsOfType Block x y )
                                     (NotSame x z )
                                     (NotSame x y )
                                     (NotSame y z))
                        Subgoals ((+ Clear y))
                        Comment 'ClearTop-Operator: Transformation
                                 y                 y
                                 x    z  ===>   x  z'
                        Phantoms ((+ Clear z ) (+ On y x))
                        Side Effects ((- On y x ) (+ On y z ) (- Clear z))
                        Purposes ((+ Clear x )))
```

A.2 Die Werkstückdomäne

```
****Objekte**************************

CAPObjectTypeClass
        makeNewClass: FaseAussen
        definitions: (Superclass (KonturFeatureAussen ))
CAPObjectTypeClass
        makeNewClass: KonturHor
        definitions: (Superclass (Kontur ))
CAPObjectTypeClass
        makeNewClass: Innendrehmeissel
        definitions: (Superclass (Werkzeug ))
CAPObjectTypeClass
        makeNewClass: Aussenkontur
        definitions: (Superclass (Kontur ))
CAPObjectTypeClass
        makeNewClass: Kontur
        definitions: ()
CAPObjectTypeClass
        makeNewClass: Innenkontur
        definitions: (Superclass (Kontur ))
CAPObjectTypeClass
        makeNewClass: Sonderwerkzeug
        definitions: (Superclass (Werkzeug ))
CAPObjectTypeClass
        makeNewClass: Stirnseite
        definitions: ()
```

```
CAPObjectTypeClass
        makeNewClass: Linksdrehmeissel
        definitions: (Superclass (Aussendrehmeissel ))
CAPObjectTypeClass
        makeNewClass: FreistichAussen
        definitions: (Superclass (KonturFeatureAussen ))
CAPObjectTypeClass
        makeNewClass: KonturAuf1
        definitions: (Superclass (Kontur ))
CAPObjectTypeClass
        makeNewClass: Bohrer
        definitions: (Superclass (Werkzeug ))
CAPObjectTypeClass
        makeNewClass: FinishAussen
        definitions: (Superclass (KonturFeatureAussen ))
CAPObjectTypeClass
        makeNewClass: Rechtsdrehmeissel
        definitions: (Superclass (Aussendrehmeissel ))
CAPObjectTypeClass
        makeNewClass: KonturFeatureAussen
        definitions: ()
CAPObjectTypeClass
        makeNewClass: Stechmeissel
        definitions: (Superclass (Werkzeug ))
CAPObjectTypeClass
        makeNewClass: RundungAussen
        definitions: (Superclass (KonturFeatureAussen ))
CAPObjectTypeClass
        makeNewClass: KonturAuf2
        definitions: (Superclass (Kontur ))
CAPObjectTypeClass
        makeNewClass: Werkzeug
        definitions: ()
CAPObjectTypeClass
        makeNewClass: Dreibackenfutter
        definitions: (Superclass (Spannmittel ))
CAPObjectTypeClass
        makeNewClass: Gewindewerkzeug
        definitions: (Superclass (Werkzeug ))
CAPObjectTypeClass
        makeNewClass: GewindeAussen
        definitions: (Superclass (KonturFeatureAussen ))
CAPObjectTypeClass
        makeNewClass: NutAussen
        definitions: (Superclass (KonturFeatureAussen ))
CAPObjectTypeClass
        makeNewClass: Aussendrehmeissel
        definitions: (Superclass (Werkzeug ))
CAPObjectTypeClass
        makeNewClass: HinterschneidungAussen
        definitions: ()
CAPObjectTypeClass
        makeNewClass: Spannmittel
        definitions: ()

****Pra"dikate********************

CAPPredicateClass
        makeNewClass: stechmeissel
        definitions: (Arguments (sM ))
CAPPredicateClass
```

```
        makeNewClass: isSpannpunktS1
        definitions: (Arguments (seite ))
CAPPredicateClass
        makeNewClass: benachbart
        definitions: (Arguments (x y ))
CAPPredicateClass
        makeNewClass: gefertigtHintS1
        definitions: (Arguments (hint kont ))
CAPPredicateClass
        makeNewClass: gefertigt
        definitions: (Arguments (x ))
CAPPredicateClass
        makeNewClass: gespanntAuf
        definitions: (Arguments (x ))
CAPPredicateClass
        makeNewClass: drehmeissel
        definitions: (Arguments (dM ))
CAPPredicateClass
        makeNewClass: drehmeisselRechtsInnen
        definitions: (Arguments (dMRInn ))
CAPPredicateClass
        makeNewClass: gefStirn
        definitions: (Arguments (seite ))
CAPPredicateClass
        makeNewClass: stechmeisselInnen
        definitions: ( Arguments (sMInn ))
CAPPredicateClass
        makeNewClass: kontGespanntAuf
        definitions: (Arguments (kont spannkont ))
CAPPredicateClass
        makeNewClass: gefertigtInnen
        definitions: ( Arguments (inn ))
CAPPredicateClass
        makeNewClass: gefertigtHintS2
        definitions: ( Arguments (hint kont ))
CAPPredicateClass
        makeNewClass: isStirnseite
        definitions: ( Arguments (x ))
CAPPredicateClass
        makeNewClass: drehmeisselLinks
        definitions: ( Arguments (dML ))
CAPPredicateClass
        makeNewClass: wzghalterFrei
        definitions: ( Arguments ())
CAPPredicateClass
        makeNewClass: drehmeisselRechts
        definitions: ( Arguments (sMR ))
CAPPredicateClass
        makeNewClass: wzgEingelegt
        definitions: ( Arguments (wzg ))
CAPPredicateClass
        makeNewClass: drehmeisselLinksInnen
        definitions: ( Arguments (dMLInn ))
CAPPredicateClass
        makeNewClass: unterbereich
        definitions: ( Arguments (x y ))
CAPPredicateClass
        makeNewClass: isSpannpunktS2
        definitions: ( Arguments (seite ))

****Axiome**********************
```

```
CAPAxiomClass
        makeNewClass: freiGespanntAuf
        definitions: ( Definition ((+ frei kont ) (+ gespanntAuf kont ))
                        Universal (kont )
                        Arguments (kont )
                        Comment '(+ frei x ) -> (- gespanntAuf x y)')
CAPAxiomClass
        makeNewClass: nurEineAufspannung
        definitions: ( Definition ((+ gespanntAuf kont ))
                        Arguments (kont ))
CAPAxiomClass
        makeNewClass: 'wzgEingelegt-wzghalterFrei'
        definitions: ( Definition ((+ wzgEingelegt wzg ) (+ wzghalterFrei )
)
                        Arguments (wzg ))

****Operatoren*******************

CAPDomainOperatorClass
        makeNewClass: FertigeHintS2SW
        definitions: ( Arguments (hint wzg kont spannkont spannkont1 )
                       Constraints ((IsOfType Linksdrehmeissel wzg )
                                    (IsOfType Kontur spannkont )
                                    (IsOfType Kontur spannkont1 )
                                    (NotSame spannkont spannkont1 )
                                    (NotSame kont spannkont1 ))
                       Subgoals ((+ wzgEingelegt wzg ) (+ gefertigt kont )
                                 (+ gespanntAuf spannkont1 )
                                 (+kontGespanntAuf kont spannkont ))
                       Phantoms ()
                       Purposes ((+ gefertigtHintS2 hint kont )))

CAPDomainOperatorClas
        makeNewClass: SpanneAufS
        definitions: (Arguments (kont seite seite1 kont1 )
                      Constraints((IsOfType Kontur kont1 )
                                  (IsOfType Stirnseite seite )
                                  (IsOfType Stirnseite seite1 )
                                  (NotSame kont kont1 )
                                  (NotSame seite seite1 ))
                      Subgoals ()
                      Comment ''
                      Phantoms ((+ benachbart kont seite1 )
                                (+benachbartkont1 seite ))
                      Side Effects ((- gespanntAuf kont1 ))
                      Purposes ((+gespanntAuf kont )))

CAPDomainOperatorClass
        makeNewClass: FertigeHintS1
        definitions: (Arguments (hint kont wzg spannkont )
                      Constraints ((IsOfType Linksdrehmeissel wzg )
                                   (IsOfType Kontur spannkont )
                                   (NotSame kont spannkont ))
                      Subgoals ((+ gefertigt kont )
                                (+ kontGespanntAuf kontspannkont )
                                (+ wzgEingelegt wzg )
                                (+ gespanntAuf spannkont ))
                      Phantoms ((+drehmeisselLinks wzg ))
                      Side Effects ()
                      Purposes ((+ gefertigtHintS1 hintkont )))
```

```
CAPDomainOperatorClass
        makeNewClass:  FertigeFeatureAllg
        definitions:  ( Arguments (feature kont wzg spannkont )
                        Constraints (( IsOfType Linksdrehmeissel wzg )
                                     ( IsOfType Kontur kont )
                                     ( IsOfType Kontur spannkont )
                                     ( IsOfType KonturFeatureAussen feature
))
                        Subgoals (( + gefertigt kont )
                                  ( + kontGespanntAuf kont spannkont )
                                  ( + gespanntAuf spannkont )
                                  ( + wzgEingelegt wzg ))
                        Phantoms (( + unterbereich feature kont )
                                  ( + #drehmeisselLinks wzg ))
                        Purposes ((+ gefertigt feature )))

CAPDomainOperatorClass
        makeNewClass: FertigeHint
        definitions: ( Arguments (hint kont )
                       Constraints ((IsOfType Kontur kont )
                                    (IsOfType HinterschneidungAussen hint
))
                       Subgoals ((+ gefertigtHintS1 hint kont )
                                 (+ gefertigtHintS2 hint kont ))
                       Phantoms ((+ unterbereich hint kont ))
                       Purposes ((+ gefertigt hint )))

CAPDomainOperatorClass
        makeNewClass: Fertige
        definitions: ( Arguments (kont wzg spannkont seite )
                       Constraints ((IsOfType Kontur kont )
                                    (IsOfType Kontur spannkont )
                                    (NotSame kont spannkont )
                                    (IsOfType Stirnseite seite )
                                    (IsOfType Linksdrehmeissel wzg ))
                       Subgoals ((+ gespanntAuf spannkont )
                                 (+ wzgEingelegt wzg ))
                       Phantoms ((+ benachbart spannkont seite )
                                 (+ drehmeisselLinks wzg ))
                       Side Effects ((+ kontGespanntAuf kont spannkont ))
                       Purposes ((+ gefertigt kont )))


CAPDomainOperatorClass
        makeNewClass: FertigeHintS2WW
        definitions: ( Arguments (hint wzg kont spannkont )
                       Constraints ((IsOfType Rechtsdrehmeissel wzg )
                                    (IsOfType Kontur spannkont )) S
                       Subgoals ((+ wzgEingelegt wzg )
                                 (+ gefertigt kont )
                                 (+ kontGespanntAuf kont spannkont )
                                 (+ gespanntAuf spannkont ))
                       Phantoms ((+ drehmeisselRechts wzg ))
                       Purposes ((+ gefertigtHintS2 hint kont )))

CAPDomainOperatorClass
        makeNewClass: LegeWzgEin
        definitions: ( Arguments (wzg1 )
                       Constraints ()
                       Subgoals ((+ wzghalterFrei ))
```

```
                        Comment ''
                        Phantoms ()
                        Side Effects ((- wzghalterFrei ))
                        Purposes ((+ wzgEingelegt wzg1 )))

CAPDomainOperatorClass
        makeNewClass: FertigeStirn
        definitions: ( Arguments (seite spannkont wzg seite1 )
                        Constraints ((IsOfType Kontur spannkont )
                                     (IsOfType Stirnseite seite )
                                     (IsOfType Stirnseite seite1 )
                                     (IsOfType Linksdrehmeissel wzg )
                                     (NotSame seite seite1 ))
                        Subgoals ((+ gespanntAuf spannkont )
                                  (+ wzgEingelegt wzg ))
                        Comment ''
                        Phantoms ((+ benachbart spannkont seite1 )
                                  (+ drehmeisselLinks wzg ))
                        Side Effects ()
                        Purposes ((+ gefStirn seite )))


CAPDomainOperatorClass
        makeNewClass: MacheWzghalterFrei
        definitions: ( Arguments (wzg )
                        Constraints ((IsOfType Werkzeug wzg ))
                        Subgoals ((+ wzgEingelegt wzg ))
                        Phantoms ()
                        Side Effects ((- wzgEingelegt wzg ))
                        Purposes ((+ wzghalterFrei )))
```

Literaturverzeichnis

[Bäc94] Christer Bäckström. Finding least constrained plans and optimal parallel executions is harder than we thought. In C. Bäckström and E. Sandewall, editors, *Current Trends in AI Planning*. IOS Press, 1994.

[Ber92] Ralph Bergmann. Learning plan abstractions. In *Proceedings of GWAI 92, 16th German Workshop on AI*. Springer Lecture Notes on AI, 1992.

[Bly93] Jim Blythe. Backtracking in 4.0. October 1993.

[BV92] J. Blythe and M. Veloso. An analysis of search techniques for a totally-ordered nonlinear planner. In *Proceedings of 1st International Conference on AI Planning Systems*, pages 13–19, 1992.

[BV94] Daniel Borrajo and Manuela Veloso. Incremental learning of control knowledge for improvement of planning efficiency and plan quality. In *Proceedings of AAAI Fall Symposium 1994: 'Planning and Learning: Onto Real Applications'*, 1994.

[BW92] A. Barrett and D.S. Weld. Partial-order planning: Evaluating possible efficiency gains. Technical report 92-05-01, Dep. of Computer Science and Engineering, University of Washington, Seattle, WA 98195, 1992.

[BW93] A. Barrett and D.S. Weld. Characterizing subgoal interactions for planning. In *Proceedings of IJCAI-93*, pages 1388–1393, 1993.

[BW94] A. Barrett and D. Weld. Schema parsing: Hierarchical planning for expressive languages. In *Proceedings of AAAI 94*, 1994.

[Byl91] Tom Bylander. Complexity results for planning. In *Proceedings of 12th IJCAI*, 1991.

[Cha87] D. Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32:333–337, 1987.

[CP92] Gregg Collins and Louise Pryor. Achieving the functionality of filter conditions in a partial order planner. In *Proceedings of AAAI 92*, 1992.

[CT91] K. Currie and A. Tate. O-plan: the open planning architecture. *Artificial Intelligence*, 51(1), 1991.

[CtPRG92] J.G. Carbonell and the PRODIGY Research Group. Prodigy4.0: The manual and tutorial. Technical Report CMU-CS-92-150, School of Computer Science, Carnegie Mellon University, 1992.

[DC88] Mark Drummon and Ken Currie. Exploiting temporal coherence in nonlinear plan construction. *Computational Intelligence*, 4:341–348, 1988.

[DC89] Mark Drummond and Ken Currie. Goal ordering in partially ordered plans. In *Proceedings of 11th IJCAI*, 1989.

[Den95] Dietmar Dengler. Adaptives Planen in Intelligenten Assistenzsystemen. In *Beiträge zum 9. Workshop 'Planen und Konfigurieren'*, 1995.

[DGT94] Brain Drabble, Yolanda Gil, and Austin Tate. Acquiring criteria for plan quality control. Extended Abstract, October 1994.

[Dra94] Brian Drabble. O-plan project second year demonstration: Reasoning with resources. Technical report, AIAI, University of Edinburgh, September 1994. Tech Report ARPA-RL/O-Plan/TR/19 Version 1.

[EHN94] K. Erol, J. Hendler, and D. Nau. UMCP: A sound and complete Procedure for Hierarchical Task-Network Planning. In *Proceedings of AIPS 94*, 1994.

[Elk89] Charles Elkan. Conspiracy numbers and caching for searching and/or trees and theorem-proving. In *Proceedings of 11th IJCAI*, 1989.

[ENS92] K. Erol, D.S. Nau, and V.S. Subrahmanian. When is planning decidable? In *Proceedings of 1st International Conference on AI Planning Systems*, pages 222–227, 1992.

[Etz90] Oren Etzioni. *A Structural Theory of Explanation-Based Learning.* Tec Report CMU-CS-90-185, Carnegie Mellon University, 1990.

[Etz93] Oren Etzioni. Acquiring search-control knowledge via static analysis. *Artificial Intelligence*, 62, 1993.

[FLY91] David E. Foulser, Ming Li, and Qiang Yang. A quantitative theory for plan merging. In *Proceedings of AAAI 91*, 1991.

[FV92] E. Fink and M. Veloso. Prodigy planning algorithm. Technical report CMU-CS-94-123, School of Computer Science, Carnegie Mellon University, 1992.

[GD90] Jonathan M. Gratch and Gerald F. DeJong. Utility generalization and composability problems in explanation-based learning. UIUUCDCS-R-91-1681, U. of Illinois at Urbana-Champaign, 1990.

[GK91] P. Godefroid and F. Kabanza. An efficient reactive planner for synthesizing reactive plans. In *Proceedings of the 9th National Conference on Artificial Intelligence*, 1991.

[GL94] Yolanda Gil and Marc Linster. On analyzing planning applications. In *1994 AAAI Fall Symposium on Planning and Learning: On to Real Applications*, 1994.

[GN91] Naresh Gupta and Dana S. Nau. Complexity results for blocks-world planning. In *Proceedings of AAAI 91*, 1991.

[GPB82] A. Goldberg, P.W. Purdom, and C.A. Brown. Average time analysis of simplified davis-putnam procedures. *Information Process. Lett.*, 15:72–75, 1982.

[Her95] J. Hertzberg. Anytime - Algorithmen. *KI*, 1:40, January 1995.

[HGS93] W.D. Harvey, M.L. Ginsberg, and D.E. Smith. Deferring conflict resolution retains systematicity. In *Proceedings of AAAI-93*, 1993.

[HRHR79] B. Hayes-Roth and F. Hayes-Roth. A cognitive model of planning. *Cognitive Science*, 3, 1979.

[JP94] David Joslin and Martha E. Pollack. Least-cost flaw repair: A plan refinement strategy for partial-order planning. In *Proceedings of AAAI94*, 1994.

[Kam92] S. Kambhampati. Characterizing multi-contributor causal structures for planning. In *Proceedings of 1st International Conference on AI Planning Systems*, pages 116–125, 1992.

[Kam93a] S. Kambhampati. On the utility of systematicity: Understanding tradeoffs between redundancy and commitment in partial-order planning. In *Proceedings of IJCAI-93*, pages 116–125, 1993.

[Kam93b] Subbarao Kambhampati. A comparative analysis of search space size, systematicity and performance of po planners. CSE Tech Report, Arizona State University, 1993.

[Kam94] S. Kambhampati. Multi-contributor causal structures for planning: A formalization and evaluation. *Artificial Intelligence*, 69:235–278, 1994.

[Kam95] S. Kambhampati. Admissible pruning strategies based on plan minimality for plan space planning. In *Proceedings of IJCAI 95*, 1995.

[KK94a] S. Kambhampati and S. Kedar. A unified framework for explanation-based generalization of partially ordered and partially instantiated plans. *Artificial Intelligence*, 67:29–70, 1994.

[KK94b] S. Katukam and S. Kambhampati. Learning explanation-based search control rules for partial order planning. In *Proceedings of the 12th AAAI*, pages 582–587, 1994.

[KKY94] Subbarao Kambhampati, Craig A. Knoblock, and Quiang Yang. Planning as refinement search: A unified framework for evaluating design tradeoffs in partial-order planning. October 1994.

[KME91] Craig A. Knoblock, Steven Minton, and Oren Etzioni. Integrating abstraction and explanation-based learning in prodigy. In *Proceedings of AAAI 91*, 1991.

[Kno90] C. A. Knoblock. *Automatically Generating Abstractions for Problem Solving.* PhD thesis, CMU, SCS, 1990. also: Tech report CMU-CS-91-120.

[Kor87] R.E. Korf. Planning as search: A quantitative approach. *Artificial Intelligence*, 33:65–88, 1987.

[KY95] Craig A. Knoblock and Qiang Yang. Relating the performance of partial-order planning algorithms to domain features. *SIGART Bulletin*, 6(1), January 1995.

[Len83] Douglas B. Lenat. Toward a theory of heuristics. In R. u. M. Groner and W.F. Bischof, editors, *Methods Of Heuristics*. Lawrence Erlbaum Associates, 1983.

[MBD94] Steven Minton, John Bresina, and Mark Drummond. Total-order and partial-order planning: A comparative analysis. *Journal of Artificial Intelligence Research*, 2, 1994.

[Min88] S. Minton. *Learning Search Control Knowledge: An Explanation-Based Approach.* Kluwer Academic Publishers, Boston, 1988.

[MLG91] Donal F. Geddis Matthew L. Ginsberg. Is there any need for domain-dependent control information. In *Proceedings of AAAI 91*, 1991.

[MP92] T.L. McCluskey and J.M. Porteous. Some empirical results in planning speed-up using combinations of on-line adaptation techniques. In *Proceedings of the Workshop on Knowledge Compilation and Speedup Learning.* Univ. of Aberdeen, Scotland, July 1992.

[MR91] D. McAllester and D. Rosenblitt. Systematic nonlinear planning. In *Proceedings of AAAI-91*, pages 634–639, 1991.

[Mus94] Nicola Muscettola. On the utility of bottleneck reasoning for scheduling. In *Proceedings of AAAI 94*, 1994.

[Nil80] N. Nilsson. *Principles of Artificial Intelligence.* Palo Alto, California, Tioga Press, 1980.

[NSS60] Allen Newell, J.C. Shaw, and Herb Simon. Report on a general problem-solving program. In *Proceedings of International Conference on Information Processing*, Paris, 1960. UNESCO.

[PC93] M. A. Perez and J.G. Carbonell. Automated acquisition of control knowledge to improve the quality of plans. TR CMU-CS-93-142, CMU, 1993.

[Pea84] Judea Pearl. *Heuristics –Intelligent Search Strategies for Computer Problem Solving.* Addison Wesley, 1984.

[Pet91] Ch. Petrie. *Planning and Replanning with Reason Maintenance.* PhD thesis, University of Texas/Austin, 1991.

[Pet92] Ch. Petrie. Constrained decision revision. *Proceedings of AAAI-92*, pages 393–400, 1992.

[Pol95] Martha E. Pollack. Evaluating planners, plans and planning agents. *SIGART Bulletin*, 6(1), January 1995.

[Rob65] J.A. Robinson. A machine-oriented logic based on the resolution principle. *JACM*, 12(1), January 1965.

[RW90] M.M. Richter and O. Wendel. Vorlesungsskript: Lernende Systeme, Band 1, 1990.

[Sac73] E. Sacerdoti. Planning in a hierarchy of abstraction spaces. In *Proceedings of IJCAI-73*, 1973.

[Sac75] E. Sacerdoti. The nonlinear nature of plans. In *Proceedings of IJCAI-75*, pages 206–214, 1975.

[SP93] D. Smith and M. Peot. Postponing threats in partial-order planning. In *Proceedings of AAAI-93*, pages 500–506, 1993.

[Sus75] G. Sussman. *A Computer Model of Skill Acquisition.* American Elsevier, New York, 1975.

[SVB94] P. Stone, M. Veloso, and J. Blythe. The need for different domain-independent heuristics. In *Proceedings of 2nd International Conference on AI Planning Systems*, pages 13–19, 1994.

[Tat74] A. Tate. Interplan: A plan generation system which can deal with interactions between goals. Machine Intelligence Research Unit Memorandum MIP-R-109, University of Edinburgh, 1974.

[Tat95] Austin Tate. Characterising plans as a set of constraints - the ¡i-n-ova¿ model - a framework for comparative analysis. *SIGART Bulletin*, 6(1), January 1995.

[TDD94] Austin Tate, Brian Drabble, and Jeff Dalton. The use of condition types to restrict search in an ai planner. In *Proceedings of AAAI 94 (?)*, 1994.

[TDK94] A. Tate, B. Drabble, and R. Kirby. O-plan2: an open architecture for command planning and control. In M. Fox and M. Zweben, editors, *Intelligent Scheduling.* Morgan Kaufmann, 1994.

[VB94] M. Veloso and J. Blythe. Linkability: Examining causal link commitments in partial-order planning. In *Proceedings of 2nd International Conference on AI Planning Systems*, pages 13–19, 1994.

[Vel89] Manuela M. Veloso. Nonlinear problem solving using intelligent casual-commitment. CMU-CS-89-210, CMU, December 1989.

[Vel92] M.M. Veloso. *Learning by Analogical Reasoning in General Problem Solving.* PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 1992.

[Völ94] Bernd Völker. Implementierung von Problem Space Graphen für das Planungssystem CAPlan. Projektarbeit, Universität Kaiserslautern, 1994.

[Wal75] R. Waldinger. Achieving several goals simultaneously. *Machine Intelligence*, 8, 1975. Ellis Harwood, Ltd.

[War74] D.H.D. Warren. A system for generating plans. Department of Computational Logic Memo No. 76, University of Edinburgh, 1974.

[Wäs93] Jürgen Wäsch. Spezifikation und Implementierung eines generativen Planungssystems zur Unterstützung der Arbeitsplanung bei der Computerintegrierten Fertigung (CIM). Master's thesis, Universität Kaiserslautern, 1993.

[Web94] F. Weberskirch. Realisierung eines nichtlinearen Planungssystems zur Unterstützung der Arbeitsplanerstellung bei der computerintegrierten Fertigung (CIM). Diplomarbeit, Universität Kaiserslautern, 1994.

[Wel94] D.S. Weld. An introduction to least commitment planning. *AI Magazine*, 15(4):27–61, 1994.

[WH94] Mike Williamson and Steve Hanks. Utility-directed planning. In *Proceedings of AAAI 94*, 1994.

[Wil84] D.E. Wilkins. Domain-independent planning: Representation and plan generation. *Artificial Intelligence*, 22:269–301, 1984.

[Wil88] D.E. Wilkins. *Practical Planning - Extending the classical AI Planning Paradigm*. Morgan Kaufmann, 1988.

[Wil94] David E. Wilkins. Comparative analysis of ai planning systems. *AI magazine*, 15(4), 1994.

[Yan] Qiang Yang. An algebraic approach to conflict resolution in planning.

[Yan92] Qiang Yang. A theory of conflict resolution in planning. *Artificial Intelligence*, 58, 1992.

[YNH92] Qiang Yang, Dana S. Nau, and James Hendler. Merging separately generated plans with restricted interactions. *Computational Intelligence*, 8(4), 1992.

[YT90] Qiang Yang and Josh Tenenberg. Abtweak: Abstracting a nonlinear, least-committment planner. In *Proceedings of AAAI 90*, pages 204–209, August 1990.

[Zil93] S. Zilberstein. *Operational Rationality through Compilation of Anytime Algorithms*. Ph.D. dissertation, University of California, 1993. Also available as TR No. CSD-93-743.