Entwurf, Implementierung und experimentelle Bewertung von Auswahlverfahren für abstrakte Pläne in dem fallbasiertem Planungssystem PARIS

Wolfgang Wilke

Diplomarbeit
Universität Kaiserslautern
Juli 1994

Betreuung:
Prof. Dr. Michael M. Richter
Dipl.-Inform. Ralph Bergmann

Erklärung:

Hiermit erkläre ich, Wolfgang Wilke, daß ich die vorliegende Diplomarbeit selbständig verfaßt und keine anderen Hilfsmittel als die von mir angegeben verwendet habe.

Kaiserslautern, im Juli 1994 (Wolfgang Wilke)

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Eine Möglichkeit das Planen in Planungssystemen zu realisieren, ist das fallbasierte Planen. Vereinfacht kann darunter das Lösen von neuen Planungsproblemen anhand von bereits bekannten Plänen aus der Planungsdomäne verstanden werden. Dazu werden Pläne, die in der Vergangenheit ein Planungsproblem gelöst haben, gesammelt und bei der Lösung neuer Planungsprobleme dahin gehend modifiziert, daß sie das aktuelle Planungsproblem lösen.
Um eine größere Wiederverwendbarkeit der bereits bekannten Pläne zu erreichen, kann man nun eine konkrete Problemstellung mit ihrer Lösung aus der konkreten Planungswelt in eine abstraktere Planungswelt durch eine Abstraktion transformieren. Diese Abstraktion sollte unwichtige Merkmale der Zustände und Operatoren aus der konkreten Planungsdomäne bei der Abbildung in die abstrakte Planungsdomäne unterdrücken. Beim Lösen neuer Planungsprobleme wird nun ein zuvor akquirierter abstrakter Plan ausgewählt, der geeignet erscheint, nach einer Verfeinerung, ein konkreteres Planungsproblem zu lösen. Diese Vorgehensweise wird als fallbasiertes Planen mit Abstraktion und Verfeinerung bezeichnet.
Die Wahl des abstrakten Plans, der zur Lösung eines konkreten Planungsproblems benutzt werden soll, ist hier für das Auffinden einer Lösung und der Zeit, die dafür benötigt wird, von zentraler Bedeutung. Somit wird durch ein gutes Bewertungsverfahren in Verbindung mit einer Selektionsstrategie für diese abstrakten Pläne ein Performanzgewinn des gesamten Planungssystems erzielt. Diese Verfahren zur Bewertung und Selektion abstrakter Pläne sollten unabhängig von einer speziellen Domäne arbeiten, damit sie universell einsetzbar sind.
Das Problem, daß Kontrollwissen in einer Wissensbasis versteckte Kosten verursacht, da dieses Wissen auf seine Anwendbarkeit untersucht werden muß, wurde von [Minton, 1990] als *Utility Problem* bezeichnet. Da sich ab einer gewissen Größe der Wissensbasis der Performanzvorteil des Kontrollwissens durch die Suche und die Überprüfung auf Anwendbarkeit aufhebt, wird der Einsatz dieses Kontrollwissens unnützlich.
Der zentrale Ansatz dieser Diplomarbeit ist der Entwurf, die Implementierung und die experimentelle Bewertung von verschiedenen Verfahren zur Planbewertung und Planselektion von abstrakten, generalisierten Plänen in dem fallbasiertem Planungssystem PARIS Abstraction and Refinement in an Integrated System), um dieses Problem zu lösen. Dazu werden zwei aufeinander aufbauende Verfahren vorgestellt, die für ein besseres Problemlöseverhalten sorgen, indem sie versuchen, daß oben angesprochene Utility Problem zu beseitigen. Anschließend wurde der Performanzvorteil dieser Verfahren in einer anwendungsorientierten Domäne, in der rotationssymmetrischer Drehteile gefertigt werden, unter

Verwendung verschiedener Abstraktionsabbildung gemessen.
Das Planungssystem PARIS arbeitet mit der Methodik des fallbasierten Planens mit Abstraktion und Verfeinerung von konkreten, bzw. abstrakten Plänen. Die Planabstraktion in diesem System erfolgt nach dem PABS-Verfahren, das in [Bergmann, 1992a] dargestellt ist und in einer Projektarbeit [Wilke, 1993] implementiert wurde. Die Verfeinerung der ausgewählten generalisierten abstrakten Pläne (GAP) ist in [Bergmann, 1993a] beschrieben und wurde neben anderen abstraktionsbasierten Planungsverfahren in [Surmann, 1993] im Rahmen einer Projektarbeit implementiert und untersucht.
Im nächsten Kapitel wird das PARIS-System mit seinen Komponenten beschrieben und ein erster Ansatz zur Planbewertung und Planselektion gegeben. In den zwei folgenden Kapiteln werden dann zwei aufeinander aufbauende Bewertungsverfahren und deren Motive vorgestellt. Das darauf folgende Kapitel gliedert sich in zwei Abschnitte, wobei im ersten Abschnitt eine Einführung in die betrachtete Planungsdomäne und die verwendeten Abstraktionsabbildungen gegeben wird und im zweiten Abschnitt die experimentelle Bewertung der Arbeit dargestellt und erläutert wird. Im letzten Kapitel werden die positiven und negativen Aspekte der verschiedenen Bewertungsverfahren diskutiert. Im ersten Teil des Anhangs findet der interessierte Leser eine Beschreibung zur Bedienung des PARIS-Systems. Im zweiten Teil des Anhangs folgt eine Herleitung der Schätzung des Erwartungswerts der Problemlösezeit, der zur Planbewertung im dritten Verfahren benutzt wird. Im dritten Teil des Anhangs befindet sich die zugrundeliegende formale Beschreibung der Planungsdomäne rotationssymmetrischer Drehteile. Zum besseren Verständnis der Semantik dieser Definitionen sei hier auf [Wilke, 1993] verwiesen.
Die Implementierung des PARIS-System und somit auch die der Bewertungs- und Selektionsalgorithmen wurde in Prolog[1] vorgenommen.
Zum Verständnis werden grundlegende Kenntnisse der Prinzipien der Künstlichen Intelligenz und des Planens vorausgesetzt. Zur Einführung in die Künstliche Intelligenz können hier Lehrbücher wie [Richter, 1989] oder [Charniak and McDermott, 1985] dienen. Um sich in den Themenbereich Planen einzuarbeiten, sei hier auf [Hertzberg, 1989] verwiesen, der einen guten Überblick über dieses Thema vermittelt und eine große Auswahl an Planungsalgorithmen darstellt. Der Leser, der nur einen kurzen Einstieg in das Themengebiet der Planung sucht, findet eine gute Darstellung im Kapitel 7.2.3 in [Görtz, G., 1993]. Bei den spezielleren Themen sei auf die angegebene Literatur verwiesen. Grundkenntnisse zum Verständnis der Anwendungsdomäne aus dem Bereich des Maschinenbaus werden vom Leser nicht erwartet.

[1] Es wurde SWI-Prolog [Wielemaker, 1992] mit einer objektorientierten Schnittstelle XPCE [Wielemaker and Anjewierden, 1992] zur Erstellung graphischer Oberflächen unter Open Windows verwendet

Kapitel 2

Das fallbasierte Planen mit Abstraktion im Planungssystem PARIS

PARIS ist ein Planungssystem, in dem das fallbasierte Planen durch Abstraktion und Verfeinerung zur Lösung von Planungsaufgaben benutzt wird. Dieses System besteht im wesentlichen aus einer Lern- und einer Problemlösekomponente. Eine Übersicht über das PARIS-System ist in der Abbildung 2.1 dargestellt. Die Funktionsweise der einzelnen Komponenten wird im folgenden Kapitel erläutert.
Mit der Lernkomponente kann man konkrete Pläne aus einer konkreten Planungswelt auf abstrakte Pläne in einer abstrakten Planungswelt abbilden. Die so gewonnenen abstrakten Pläne werden anschließend generalisiert. Dies geschieht nach dem PABS-Verfahren, welches ausführlich in [Bergmann, 1992b] beschrieben wird und im Rahmen einer Projektarbeit [Wilke, 1993] implementiert wurde.
Abstraktion bedeutet einen expliziten Wechsel der Beschreibungssprache einer Planungswelt. Neben der Abstraktion ist auch die *Generalisierung* eine Möglichkeit, gegebene Pläne für ein größeres Spektrum neuer Situationen anwendbar zu machen, also ihre Wiederverwendbarkeit zu erhöhen. Verfahren zur Plangeneralisierung sind schon lange bekannt (z.B. [Fikes *et al.*, 1972; Kambhampati and Kedar, 1991]) und sind im wesentlichen eine Variante des erklärungsbasierten Lernens [Ellman, 1989]. Der Unterschied zur Abstraktion ist dabei der, daß bei der Generalisierung die Beschreibungssprache nicht gewechselt wird. Vielmehr werden bestimmte Konstanten im Plan durch Variablen ersetzt, so daß das resultierende Planschema korrekt bleibt. Die Generalisierung der abstrakten Pläne im PABS-Verfahren wird nach [Bergmann, 1990] durchgeführt.
Im ersten Abschnitt dieses Kapitels wird eine genaue Definition der dem Planungssystem zu Grunde liegenden Planungswelten gegeben. Die Abstraktion von Plänen der konkreten in die abstrakte Planungswelt und deren Generalisierung erfolgt nach dem PABS-Verfahren, das im zweiten Abschnitt des Kapitels beschrieben wird. Eine Definition der generalisierten abstrakten Pläne befindet sich ebenfalls in diesem Abschnitt.
Mit der Problemlösekomponente können diese generalisierten abstrakten Pläne, die in einer Planbibliothek gespeichert werden, zum Lösen neuer konkreter Planungsprobleme eingesetzt werden. Die Funktionsweise dieser Komponente wird im dritten Abschnitt des Kapitels beschrieben. Abschließend wird auf die Organisation der Planbibliothek eingegangen

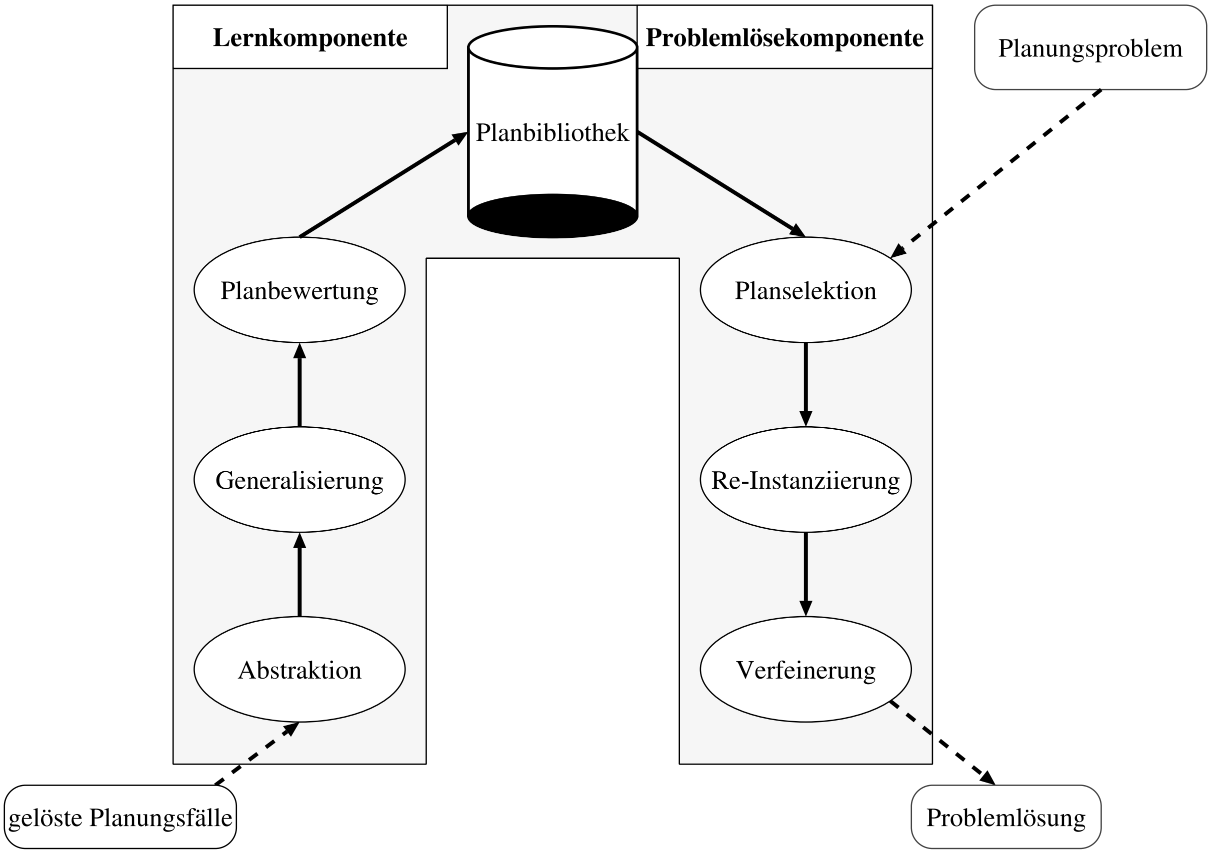

Abbildung 2.1: Eine Übersicht über die Komponenten des PARIS-Systems

und es wird die Notwendigkeit einer effektiven Organisation dieser Planbibliothek motiviert.

2.1 Die Planungswelten im Planungssystem PARIS

Eine Sichtweise des Planens ist es, von einer formalen Beschreibung einer Startsituation ausgehend, eine Folge von Aktionen zu finden, die eine formal beschriebenen Zielsituation erreichen. Die wesentlichen Elemente sind hierbei *Situationen* und *Aktionen*. Eine Situation ist der Schnappschuß des interessierenden Aussschnitts der Welt zu einem Zeitpunkt, beschrieben durch eine Menge logischer Formeln, die in dieser Situation gelten; eine Aktion ist die formale Beschreibung einer Handlung in dieser Welt. Eine Aktion überführt nach einer gegebenen formalen Vorschrift, eine Situation in eine Nachfolgesituation.
Als *Planungswelt* versteht man die formale Umgebung, in der das Planen vollzogen wird. Dazu gehört die Bereitstellung eines Formalismus, der die Formulierung von Aktionen und Situationen erlaubt. Bei der im PARIS-System benutzten Planungswelt handelt es sich um einen linearen Planer, der den Formalismus der STRIPS-Notation [Fikes and Nilsson, 1971; Lifschitz, 1986; Knoblock, 1989] benutzt. Aktionen werden hier mit *Operatoren* und Situationen mit *Zustandsbeschreibungen* bezeichnet. Die Menge der logischen Formeln, die zur Beschreibung der Zustände dienen, gliedert sich in *essentielle Sätze* und *ableitbare Sätze*.

Diese Unterscheidung wurde von Lifschitz [Lifschitz, 1986] eingeführt, als er die theoretischen Grundlagen der STRIPS-Notation untersuchte und Bedingungen für die Korrektheit eines STRIPS-Systems definierte. Dabei werden alle Effekte der Operatoren der Planungswelt, durch essentielle Sätze beschrieben. Die Planungswelt wird um eine *Theorie* erweitert, die es ermöglicht mit den essentiellen Sätzen eines Zustands weitere abgeleitete Sätze herzuleiten. So ist eine Unterscheidung möglich, welche Sätze bei der Formulierung der Operatoren berücksichtigt werden müssen, bzw. verwendet werden dürfen.

Somit besteht eine Planungswelt aus:

Definition 2.1 *Eine Planungswelt W in STRIPS-Notation besteht aus einem Tripel (R, T, Op) über einer prädikatenlogischen Sprache L erster Ordnung, wobei:*

- *$R \subseteq L$ eine Menge von essentiellen Sätzen [Lifschitz, 1986] darstellt, mit denen ein Zustand in der Planungswelt zu einem festen Zeitpunkt beschrieben werden kann.*
- *T ist eine Theorie, definiert durch eine Menge von Regeln r, sodaß:*
  $r = l_1 \wedge l_2 \wedge \ldots \wedge l_n \rightarrow \Phi$, wobei $l_i \in L$ für $i = 1, \ldots, n$ und $\Phi \in L$. Hierbei ist die Konjunktion $l_1 \wedge l_2 \wedge \ldots \wedge l_n$ die Vorbedingung der Regel und Φ der daraus resultierende ableitbare Satz .
- *Op ist eine Menge von Operatoren $o_i \in Op$, die durch ein Tripel $< P_{Oi}, D_{Oi}, A_{Oi} >$ beschrieben werden [Fikes* et al.*, 1972], wobei:*
  - *P_{Oi} eine endliche Konjunktion von Sätzen aus der zugrundeliegenden prädikatenlogischen Sprache L ist, die die Vorbedingungen zur Anwendung des Operators determinieren.*
  - *D_{Oi} eine endliche Menge von essentiellen Sätzen aus R ist, die nach der Anwendung des Operators $o_i \in Op$ nicht mehr gelten, also aus dem Zustandsraum entfernt werden.*
  - *A_{Oi} eine endliche Menge von essentiellen Sätzen aus R ist, die nach der Anwendung des Operators $o_i \in Op$ gelten, also zu dem Zustandsraum hinzugefügt werden.*

Hieraus ergibt sich die Beschreibung eines einzelnen Zustands in der Planungswelt durch:

Definition 2.2 *Ein Zustand $s \in S$ der Planungswelt W ist eine Untermenge der essentiellen Sätze R aus W. Die Menge S ist die Menge aller möglichen Zustände einer Planungswelt mit $S = 2^R$.*

Ein Planungsproblem ist nun durch ein Zustandspaar bestimmt, welches den Startzustand sowie den Zielzustand beschreibt.

Definition 2.3 *Ein Planungsproblem aus der Planungswelt W ist ein Zustandspaar (s_i, s_g), daß den Initialzustand s_i und den Zielzustand s_g, mit einer Zustandsbeschreibung aus der Planungswelt W beschreibt, also $s_i, s_g \in S$*

Darauf aufbauend kann nun ein *Plan* in einer Planungswelt wie folgt definiert werden:

Definition 2.4 *Ein Plan P in einer Planungswelt $W = (R, T, Op)$ ist eine Sequenz von Operatoren $(o_1, \ldots, o_n)$ mit $o_i \in Op$ für $i = 1, \ldots, n$.*

Dieser Plan wird in der Planungswelt nach folgender Vorschrift angewandt:

Definition 2.5 *Ein Plan $P = (o_1, \ldots, o_n)$ in einer Planungswelt $W = (R, T, Op)$ und ein Initialzustand s_0 induzieren eine Sequenz von Zuständen $s_1, \ldots, s_n$ mit $s_i \in S$ für $i = 1, \ldots, n$, derart, daß :*
$s_{i+1} = (s_i \setminus D_{Oi}) \cup A_{Oi}$, falls $s_i \cup T \vdash P_{Oi}$ mit $o_i =< P_{Oi}, D_{Oi}, A_{Oi} >$.
Falls jedoch ein i existiert mit $s_i \cup T \not\vdash P_{Oi}$, so ist der gesamte Plan auf den Initialzustand nicht anwendbar.

Ein *Planungsfall* besteht nun aus einem Planungsproblem und einem Plan, der dieses Problem löst.

Definition 2.6 *Ein Planungsfall $F := (s_i, s_g, P)$ besteht aus einem Planungsproblem (s_i, s_g) zusammen mit einem Plan P, der dieses Problem löst, also den Startzustand s_i des Planungsproblems in den Zielzustand s_g mit dem Plan $P = (o_1, \ldots, o_n)$ gemäß Definition 2.5 transformiert.*

Um die Planbibliothek mit abstrakten, generalisierten Plänen zu füllen, die zur Lösung neuer Planungsprobleme re-instanziiert und verfeinert werden, werden Planungsfälle abstrahiert und generalisiert. Die gelösten Planungsfälle können durch einen Experten vorgegeben werden oder durch einen Problemlöser maschinell akquiriert werden. Die Abstraktion und Generalisierung der konkreten Planungsfälle nach Definition 2.6 wird im folgenden Abschnitt beschrieben wird.

2.2 Das PABS-Verfahren zur Planabstraktion und Plangeneralisierung

Zur Planabstraktion werden zwei verschiedene Planungswelten vorausgesetzt: Eine *konkrete Welt* $W_k = (R_k, T_k, Op_k)$ und eine *abstrakte Welt* $W_a = (R_a, T_a, Op_a)$. *Planabstraktion* heißt nun, einen gegebenen Plan P_k in der konkreten Welt W_k in einen korrespondierenden Plan P_a der abstrakten Welt W_a abzubilden.
Die Planabstraktion und die *Plangeneralisierung* wird nach dem PABS-Verfahren [Bergmann, 1992b] realisiert, daß im nächsten Abschnitt informell beschrieben wird. Eine ausführliche Beschreibung der Implementation des Verfahrens ist in [Wilke, 1993] gegeben. Das Verfahren gliedert sich in fünf Phasen, die im Folgenden erklärt werden.

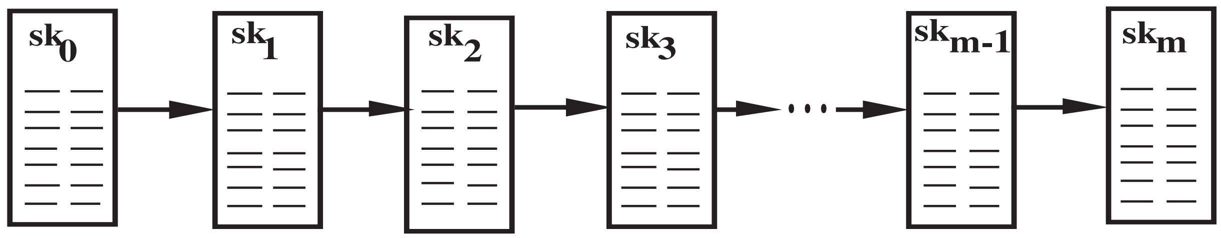

Abbildung 2.2: Die Simulation eines konkreten Plans in der ersten Phase des PABS-Verfahrens

2.2.1 Phase-I: Simulation des konkreten Plans

In der ersten Phase wird die Ausführung des konkreten Plans in der konkreten Planungswelt simuliert.

Dabei werden die Definitionen der konkreten Operatoren in der Reihenfolge, die durch den Plan vorgegeben ist, nach der Definition 2.5 auf den vorgegebenen Startzustand angewandt. Falls ein Operator aufgrund der geltenden Fakten in einem Zustand nicht anwendbar ist, wird mit Hilfe einer konkreten Theorie, die zur Beschreibung der konkreten Planungswelt gehört, versucht, die benötigten Fakten zu beweisen. Dieser Vorgang wird in Abbildung 2.2 verdeutlicht.

2.2.2 Phase-II: Abbildung der konkreten auf die abstrakten Zustände

Die so erhaltenen konkreten Zustandsbeschreibungen werden nun in die abstrakte Planungswelt abgebildet.

Dazu wird eine Beschreibung der Abstraktionsabbildung vorgegeben, die aus einer Menge von Regeln besteht, die definieren, wie sich Fakten aus der konkreten Planungswelt in die abstrakte Planungswelt transformieren lassen. Diese Menge von Regeln wird als *Abstraktionstheorie* [Giordana *et al.*, 1991] bezeichnet und ergibt sich aus:

Definition 2.7 *Sei eine konkrete Welt* $W_k = (R_k, T_k, Op_k)$ *und eine abstrakte Welt* $W_a = (R_a, T_a, Op_a)$ *gegeben. Eine generische Abstraktionstheorie* T_g*, von der konkreten Welt* W_k *in die abstrakte Welt* W_a*, ist gegeben durch eine Menge von Regeln der Form:* $\Psi \Leftarrow K_1 \wedge K_2 \wedge \ldots \wedge K_n$*, wobei* Ψ *ein essentieller Satz aus der abstrakten Welt* W_a *ist* ($\Psi \in R_a$) *und* $K_1 \wedge K_2 \wedge \ldots \wedge K_n$ *eine Konjunktion von Sätzen der konkreten Welt* W_k*,* ($K_i \in L_k$)*, für* $i \in \{1, \ldots, n\}$)*.*

Daraus ergibt sich indirekt die Abstraktion der Operatoren, deren Anwendbarkeit durch die abstrakten Fakten determiniert werden. Die konkreten Operatoren selbst werden jedoch nicht auf abstrakte Operatoren abgebildet.
In der zweiten Phase wird die Abstraktionstheorie angewandt, um die durch die Ausführung des konkreten Plans entstandenen Zustandsbeschreibungen der konkreten Welt auf abstrakte Zustandsbeschreibungen abzubilden. Diese Abbildung stellt die Realisation der

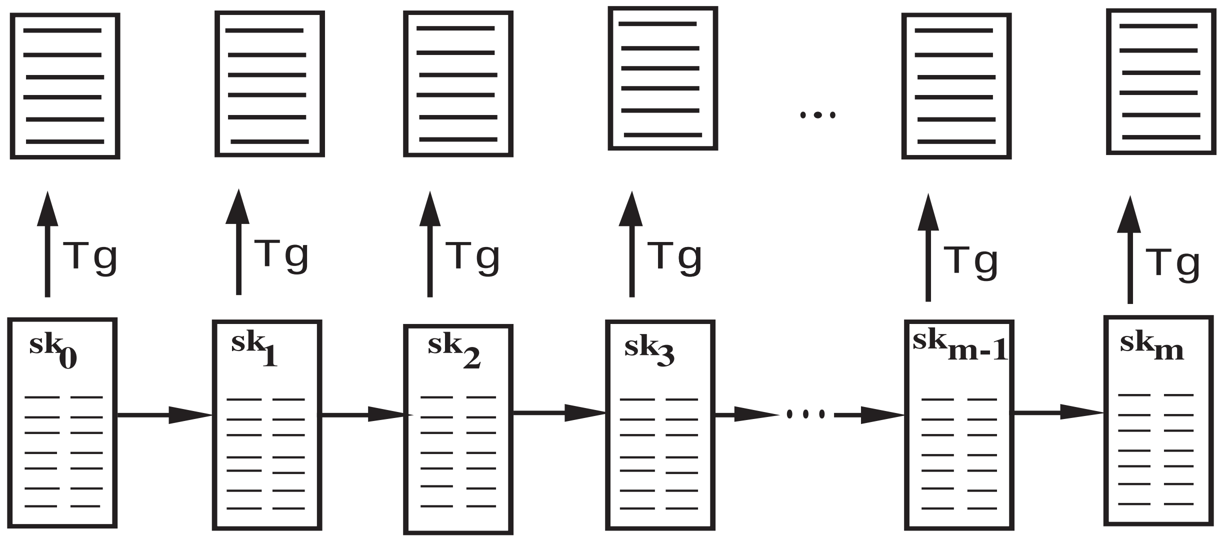

Abbildung 2.3: Die Abstraktion der konkreten Planungswelt in die abstrakte Planungswelt

Abstraktion von der konkreten Planungswelt in die abstrakte Planungswelt dar. Dieser Sachverhalt wird in Abbildung 2.3 illustriert.

2.2.3 Phase-III: Suche nach Zustandsübergängen der abstrakten Operatoren

Das Ziel der dritten Phase ist es, alle abstrakten Operatoren zu finden, die von einem abstrakten Zustand in einen beliebigen Folgezustand führen. Die Zustandsbeschreibungen müssen den STRIPS-Definitionen der abstrakten Operatoren genügen, das heißt, die Vorbedingungen, die ein Operator benötigt, um angewendet zu werden, müssen in dem gewählten Startzustand gelten. In dem Zielzustand dürfen jedoch keine Fakten mehr gelten, die in der Deleteliste des Operators existieren und alle Fakten der Addliste müssen in dem Zielzustand vorhanden sein. So erhält man einen azyklischen, gerichteten Graphen, in dem die Knoten abstrakte Zustände und die Kanten mögliche abstrakte Zustandsübergänge durch abstrakte Operatoren darstellen (Abbildung 2.4).

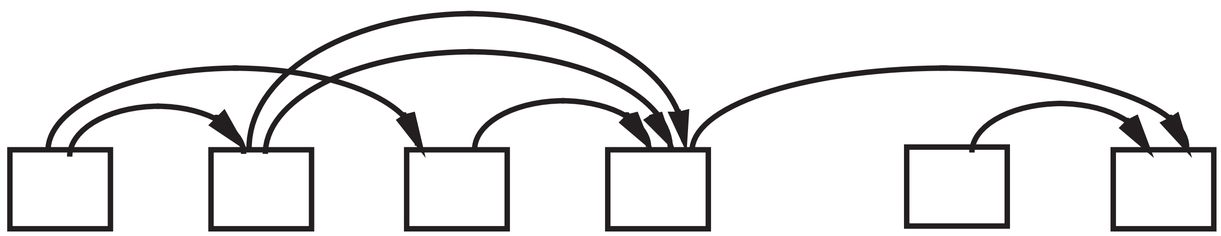

Abbildung 2.4: Der Abhängigkeitsgraph der abstrakten Zustände und der abstrakten Operatoren

2.2.4 Phase-IV: Die Suche nach konsistenten Pfaden vom Start- zum Zielzustand

In der vierten Phase werden alle *konsistenten Pfade* in der abstrakten Planungswelt gesucht, die vom abstrakten Startzustand, das ist der Zustand, der in der Phase-II mit Hilfe der generischen Theorie aus dem konkreten Startzustand hergeleitet wurde, zum abstrakten Zielzustand führen, der aus dem konkreten Zielzustand mit der Abstraktionsabbildung hervorging.
Ein Pfad ist konsistent, falls jeder Fakt der Vorbedingungen eines STRIPS-Operators, der zu diesem Pfad gehört, einer der folgenden Bedingungen genügt:

- Der Fakt galt im abstrakten Startzustand und wurde bis zur Anwendung des Operators, der ihn als Vorbedingung benötigt, nicht gelöscht.
- Der Fakt wurde durch einen vorherigen abstrakten Operator erzeugt und wird nicht wieder gelöscht, bis der Operator, der ihn benötigt, angewandt wird.
- Der Fakt kann aus Fakten, die einer der beiden vorherigen Bedingungen genügen, mit Hilfe der abstrakten Theorie, die zur abstrakten Planungswelt definiert wird, hergeleitet werden.

Abbildung 2.5: Die abstrakten Zustände mit einem konsistenten Pfad von abstrakten Operatorübergängen

So ist die Anwendbarkeit der abstrakten Operatoren durch die Anwendung der abstrakten Operatorsequenz selbst garantiert und begründet sich nicht auf Fakten, die aus der Abstraktionstheorie und den konkreten Zuständen hergeleitet wurden. Die so erhaltenen konsistenten Pfade in der abstrakten Welt, die vom abstrakten Startzustand zum abstrakten Zielzustand führen, sind dann mögliche abstrakte Pläne, die aus dem konkreten Plan hervorgingen (siehe Abbildung 2.5).

2.2.5 Phase-V: Die Generalisierung abstrakter Pläne

In der letzten Phase werden die so gewonnen abstrakten Pläne generalisiert, um eine noch größere Wiederverwendbarkeit der Pläne zu erreichen, als dies schon durch die Abstraktion erzielt wurde. Außerdem werden die Pläne in eine Darstellung transformiert, die eine schnellere Überprüfung des abstrakten, generalisierten Plans auf seine Anwendbarkeit ermöglicht. Diese Plangeneralisierung stellt den zweiten Bearbeitungsschritt in der Lernkomponente nach der Planabstraktion dar (siehe Abbildung 2.1). Diese Repräsentation der abstrakten, generalisierten Pläne (GAP's) gewinnt man durch ein erklärungsbasiertes Lernverfahren [Mitchell *et al.*, 1986; DeJong and Mooney, 1986; Minton *et al.*, 1989b; Knoblock, 1993] für abstrakte Pläne, das ausführlich in [Bergmann, 1990; Wilke, 1993] beschrieben ist. Ein *generalisierter Plan* hat dabei die folgende Gestalt:

Definition 2.8 *Ein generalisierter, abstrakter Plan GAP setzt sich zusammen aus*

$$GAP \equiv (s_i(x), s_g(y), < o_1(z_1), ..., o_n(z_n) >, R(x, y, z_1, ..., z_n))$$

Hierbei bezeichnen $s_i(x) \in S_a$ den Initialzustand und $s_g(y) \in S_a$ den Zielzustand des generalisierten Planes. $< o_1(z_1), ..., o_n(z_n) >$, mit $o_i(z_i) \in Op_a$ für $i = 1, \ldots, n$, ist die Folge von generalisierten Operatoren und $R \subseteq L_a$ eine Menge von Constraints, die die Anwendbarkeit des Plans determinieren. x, y und z_i bezeichnen Variablen, die die Parameter der Operatoren mit Hilfe der Constraints mit dem Initial- und Zielzustand in Beziehung setzen.

Das PABS-Verfahren zum wissensintensiven Lernen von abstrakten Plänen ist in der Lage, für einen gegebenen konkreten Planungsfall die Menge aller gültigen abstrakten, generalisierten Pläne zu generieren, die dann gegebenenfalls zur Lösung neuer Planungsprobleme komplexitätsreduzierend eingesetzt werden können. Dazu werden diese Pläne in einer Planbibliothek gespeichert. Zum Lösen eines neuen Planungsproblems muß ein generalisierter, abstrakter Plan re-instanziiert und so verfeinert werden, daß er eine Lösung des konkreten Planungsproblems darstellt. Dies wird mit der Problemlösekomponente durchgeführt, deren Funktionsweise im folgenden Abschnitt beschrieben wird.

2.3 Die Problemlösekomponente des PARIS - Systems

Um ein neues Planungsproblem zu lösen, wird ein generalisierter, abstrakter Plan gesucht, der auf ein neues konkretes Planungsproblem anwendbar ist. Die Bedingung, ob *ein abstrakter Plan anwendbar ist*, ergibt sich aus:

Definition 2.9 *Ein generalisierter Plan in der abstrakten Welt*

$$GAP \equiv (s_i(x), s_g(y), < O_1(z_1), ..., o_n(z_n) >, R(x, y, z_1, ..., z_n))$$

ist für ein neues konkretes Planungsproblem $(\tilde{s}_i, \tilde{s}_g)$ mit $\tilde{s}_i, \tilde{s}_g \in S_k$ anwendbar, falls

- *es eine Belegung der Variablen $x, y, z_1, ..., z_n$ gibt, sodaß alle Relationen in R gelten.*

und für diese Belegung gilt:

  - *$s_i(x)$ eine Abstraktion von $\tilde{s}_i$ darstellt, also gilt: $\tilde{s}_i \cup T_g \vdash s_i(x)$*

– *$s_g(y)$ eine Abstraktion von $\tilde{s}_g$ darstellt, also gilt:* $\tilde{s}_g \cup T_g \vdash s_g(y)$

Die Überprüfung, ob ein abstrakter Plan anwendbar ist, ist mit einem gewissen Aufwand verbunden. Es muß eine Belegung der Variablen $x, y, z_1, ..., z_n$ gefunden werden, so daß alle Relationen aus R erfüllt sind und die Abstraktionstheorie T_g, die konkreten Start- und Zielzustände des neuen Planungsproblems $(\tilde{s}_i, \tilde{s}_g)$ auf die abstrakten Start- und Zielzustände des generalisierten, abstrakten Plans (GAP), also $s_i(x), s_g(y)$, abbildet. Daraus ergibt sich die Notwendigkeit, bei der Überprüfung der Anwendbarkeitsbedingung nach Definition 2.9, möglichst wenig Pläne zu überprüfen, um die Problemlösezeit zu reduzieren.
Ist so eine *Re-Instanziierung* eines GAP aus der Planbibliothek gefunden, muß der abstrakte Plan zur Lösung des neuen konkreten Planungsproblems verfeinert werden [Friedland and Iwasaki, 1985; Bergmann, 1993a; Surmann, 1993]. Bei der *Verfeinerung* eines

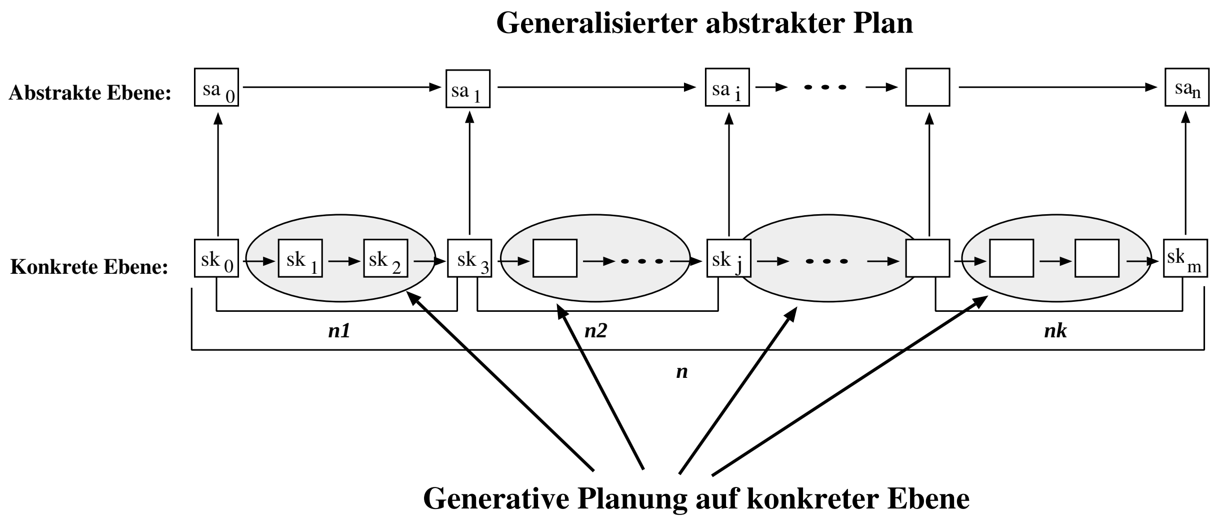

Abbildung 2.6: Die Verfeinerung von generalisierten, abstrakten Plänen mit der Problemlösekomponente

re-instanziierten, abstrakten Plans werden ausgehend vom konkreten Startzustand, Folgezustände mit konkreten Operatoren erzeugt. Ist eine konkrete Zustandsbeschreibung gefunden, die sich mit der Abstraktionstheorie T_g auf die folgende abstrakte Zustandsbeschreibung abbilden läßt, ist ein Teilplan zur Lösung des konkreten Planungsproblems gefunden. Ist so für alle abstrakten Operatorübergänge ein konkreter Plan gefunden, stellen diese Teilpläne eine Lösung des konkreten Planungsproblems dar. Dazu werden die gefundenen Teilpläne in der Reihenfolge konkateniert, die durch die abstrakte Operatorsequenz vorgegeben wird. Dieses Verfahren soll noch einmal durch die Abbildung 2.6 verdeutlicht werden. So ergibt sich beim Planen durch Verfeinerung eine Dekomposition des ursprünglichen Planungsproblems in mehrere kleinere Teilprobleme. Dies gilt natürlich nur, wenn die abstrakten Operatoren für eine Dekomposition des Problems sorgen, sodaß die einzelnen Teilprobleme voneinander unabhängig sind. Dadurch wird vermieden, daß es zu einem Backtracking zwischen den einzelnen Suchräumen bei der Lösung der einzelnen Teilprobleme kommt.
Die Komplexität der Suche nach einem Plan der Länge n bei m möglichen anwendbaren konkreten Operatoren beträgt $O(m^n)$. Durch die Aufteilung des Planes in mehre-

re kürzere Teilpläne beim Planen mit Verfeinerung von *GAP*'s reduziert sich diese auf $O(m^{n_1} + m^{n_2} + \ldots + m^{n_k})$, wobei $n_1 + \ldots + n_k = n$ [Korf, 1987; Knoblock, 1990; Knoblock, 1991]. Die Abbildung 2.6 verdeutlicht diese Dekomposition der Suchräume bei der Problemlösung noch einmal.

2.4 Die Selektion abstrakter, generalisierter Pläne

Für ein neues konkretes Planungsproblem sind in der Regel mehrere generalisierte, abstrakte Pläne zur Lösung mit der Problemlösekomponente geeignet. Diese anwendbaren Pläne zeichnen sich jedoch durch verschiedene Re-Instanziierungs- und Verfeinerungszeiten aus. Somit ist die Wahl des abstrakten, generalisierten Plans, der zur Lösung eines konkreten Planungsproblems benutzt wird, für das Auffinden einer Lösung und der Zeit, die dafür benötigt wird, von zentraler Bedeutung. Somit wird durch ein gutes a-priori *Bewertungsverfahren* für die abstrakten, generalisierten Pläne und geeignete *Selektionsstrategien* ein Performanzgewinn des gesamten Planungssystems erzielt. Die Bewertung ist a-priori, da nur Informationen über die bereits bekannten Planungsprobleme zur Verfügung stehen und der generalisierte, abstrakte Plan im bezug auf alle Planungsprobleme der konkreten Planungswelt bewertet werden muß. Weiterhin muß eine Selektionsstrategie möglichst schnell einen guten abstrakten Plan zur Lösung eines konkreten Planungsproblems auffinden. Dies bedeutet bei den Kosten bei der Überprüfung auf Anwendbarkeit eines abstrakten, generalisierten Plans (nach Definition 2.9), daß das Selektionsverfahren möglichst wenig Pläne betrachten darf, bevor es einen Plan findet, der sich auf das konkrete Planungsproblem verfeinern läßt.

2.4.1 Die Anordnung der Pläne in einer linearen Liste

Als ein naiver Ansatz kann man die Pläne der Planbibliothek linear in einer Liste anordnen. Innerhalb dieser Liste werden die Pläne nach der Länge ihrer abstrakten Operatorsequenz sortiert, die den abstrakten Startzustand in den abstrakten Zielzustand überführt. Diese Anordnung versucht die Suchräume beim generischen Planen auf konkreter Ebene möglichst klein zu halten. Bei der Selektion eines geeigneten abstrakten Plans zur Lösung eines konkreten Planungsproblems wird diese Liste beginnend beim längsten abstrakten Plan sequentiell durchlaufen. Führt die Verfeinerung nicht zu einer Lösung des konkreten Planungsproblems, wird die Liste weiter durchsucht und es wird der nächst mögliche Plan zur Problemlösung benutzt, der auf das konkrete Planungsproblem anwendbar ist. So wird garantiert, daß der längste abstrakte Plan, der auf das konkrete Planungsproblem anwendbar ist und der sich auch verfeinern läßt, zum Problemlösen benutzt wird. Dieser Plan verfügt wegen seiner vielen abstrakten Operatoren über die meisten abstrakten Zustandsbeschreibungen. So wird das konkrete Planungsproblem beim Verfeinern in möglichst viele einzelne Teilprobleme zerlegt. Die Suchräume dieser Teilprobleme werden so möglichst klein gehalten, da eine kürzere Operatorsequenz genügt, um ein Teilplan zur Lösung des Planungsproblems zu finden. So findet die generischen Planung auf der konkreten Ebene, bei der eine vollständige Suche nach allen konkreten Operatorsequenzen durchgeführt wird, schneller eine Operatorsequenz, die das Teilproblem löst.
Ist man am Ende der Liste der abstrakten Pläne angekommen und es wurde kein Plan

gefunden, der auf das konkrete Planungsproblem anwendbar und verfeinerbar ist, so kann man das konkrete Planungsproblem mit dem Verfahren der Verfeinerung abstrakter Pläne nicht lösen. Dies deutet auf einen Defizit in der abstrakten Planbibliothek hin. Es besteht jedoch innerhalb des PARIS-Systems die Möglichkeit, das konkrete Planungsproblem mit anderen Planungsverfahren zu lösen. Hat man so eine Lösung des konkrete Planungsproblems gefunden, sollte diese Lösung als Planungsfall der Lernkomponente zur Verfügung gestellt werden, um den oben aufgezeigten Defizit in der Planbibliothek zu beseitigen.
Die Anzahl der betrachteten Pläne bei der Selektion und die Zeiten zur Überprüfung auf Anwendbarkeit durch Re-Instanziierung werden bei diesem Verfahren vernachlässigt. Dies hat zur Folge, daß ab einer gewissen Anzahl an Plänen in der Planbibliothek mit der Suche nach einem anwendbaren Plan soviel Zeit aufgewendet wird, daß sich kein Vorteil mehr durch das Problemlösen mit Re-Instanziierung und Verfeinerung ergibt.

Kapitel 3

Das inkrementelle Lernen von Abstraktionshierarchien

Beim Anordnen der abstrakten Pläne in einer linearen Liste wurde versucht, die Verfeinerungszeit beim Problemlösen zu verringern. Die Anzahl der auf Anwendbarkeit überprüften Pläne und die Zeiten, die dafür zum Re-Instanziieren benötigt werden, wurden jedoch nicht betrachtet. Will man jedoch die Gesamtproblemlösezeit beim fallbasierten Planen mit der Verfeinerung abstrakter Pläne minimieren, sollte man alle drei verschiedenen Suchräume unterscheiden, die die benötigte Zeit zum Planen beeinflussen, und zwar:

- Den Suchraum bei der Wahl eines abstrakten, generalisierten Plans
- Den Suchraum bei der Re-Instanziierung des abstrakten Plans
- Den Suchraum der konkreten Planungsfunktion bei der Verfeinerung eines abstrakten Plans durch die generative Planung auf konkreter Ebene

Die Gesamtproblemlösezeit ergibt sich aus der Addition der Zeiten, die für die einzelnen oben aufgeführten Suchen verbraucht werden. Somit ist ein gutes Bewertungsverfahren für die Güte eines abstrakten Plans[1] dadurch gekennzeichnet, daß bei der Planselektion:

- möglichst wenig abstrakte Pläne auf Anwendbarkeit untersucht werden,
- ein Plan gewählt wird, der sich auf das neue Planungsproblem schnell re-instanziieren läßt, und
- dessen Re-Instanziierung schnell zu einer Lösung des neuen konkreten Planungsproblems durch die Verfeinerung führt.

Bei ersten Experimenten mit der Anordnung der abstrakten Pläne in einer linearen Liste, sortiert nach der Länge ihrer abstrakten Operatorsequenz, zeigte sich, daß insbesondere lange abstrakte Pläne hohe Re-Instanziierungszeiten bei der Problemlösung verursachten.

[1] Im weiteren Verlauf wird, außer es ist erforderlich, nur noch von abstrakten Plänen anstatt von generalisierten, abstraken Plänen gesprochen.

Außerdem werden bei diesem Verfahren sehr viele Pläne betrachtet, bis ein Plan zur Lösung eines konkreten Planungsproblems gefunden wurde. Aufgrund dieser beiden Beobachtungen sollte bei einer besseren Planbewertung, bzw. Planselektion sowohl die Anzahl der auf Anwendbarkeit getesteten Pläne bei der Selektion minimiert werden, als auch eine gezieltere Auswahl der Pläne erfolgen. Des weiteren ist die abstrakte Planlänge ein ungenaues Maß, um etwas über die Nützlichkeit der Anwendung eines abstrakten Plans auszusagen.
Als ein Ansatz zur Lösung des Utility-Problems wurde von [Yoo and Fisher, 1991] vorgeschlagen, gelernte Regeln in einer Klassifikationshierarchie anzuordnen. In einer solchen Hierarchie sind die Regeln an einem Knoten immer genereller als die Regeln an dessen Nachfolgern. Wenn also eine Regel an einem Knoten anwendbar ist, so ist folglich auch die Regel an dem übergeordneten Knoten anwendbar. Umgekehrt folgt daraus, falls eine Regel an einem Knoten nicht anwendbar ist, daß auch keine der Regeln an den Nachfolgerknoten anwendbar sind. Zum Finden der speziellsten anwendbaren Regel kann nun der Baum solange durchlaufen werden, wie es noch Nachfolgerknoten gibt, die anwendbar sind.
Um dieses Prinzip zum Planen mit abstrakten Plänen einzusetzen, müssen die Pläne nach ihrem Abstraktionsgrad angeordnet werden, so daß eine Abstraktionshierarchie zur Klassifikation entsteht. Auch hierbei gilt: Ist ein abstrakter Plan nicht anwendbar, so sind auch alle seine Spezialisierungen nicht mehr anwendbar. Werden nun die abstrakteren Pläne höher in einer Klassifikationshierarchie angeordnet als die spezielleren Pläne, so kann man, falls man beim Durchlaufen dieses Baumes auf einen nicht anwendbaren Plan stößt, alle weiteren Pläne im nachfolgenden Teilbaum ignorieren, da sie keinesfalls mehr anwendbar sein können. Hierdurch kann der Suchaufwand im Vergleich zu einer linearen Anordnung erheblich reduziert werden, da weniger abstrakte Pläne auf Anwendbarkeit getestet werden. Das ist das zentrale Anliegen dieses Indexierungsansatzes, der in diesem Kapitel vorgestellt wird.
In ersten Abschnitt dieses Kapitels werden die Grundlagen des Lernalgorithmus zur Bildung einer solchen Abstraktionshierarchie erläutert. Nach einer informellen Beschreibung des Algorithmus wird dieser ausführlich beschrieben. Im letzten Abschnitt des Kapitels wird die Funktionsweise des Algorithmus noch einmal an einem Beispiel erläutert.

3.1 Die Grundlagen zur Beschreibung des Algorithmus

Als Eingabe erhält die Lernkomponente des PARIS-Systems ein gelöstes Planungsproblem, also ein Planungsfall $F_j = (si_j, sg_j, P_j)$ aus der konkreten Planungswelt und generiert hieraus eine Menge von generalisierten, abstrakten Plänen aus der abstrakten Planungswelt. Um eine Abstraktionshierarchie zu erhalten, muß der Abstraktionsgrad der so entstandenen abstrakten Pläne bestimmt werden. Hierbei konkretisiert sich die Bedingung, wann *ein abstrakter Plan* GAP_1 *spezieller ist als ein anderer abstrakter Plan* GAP_2 wie folgt:

Definition 3.1 $GAP_1 \sqsubseteq GAP_2$ *(*GAP_1 *ist spezieller als* GAP_2*) gdw für alle konkreten Zustandspaare* $si, sg \in S_k$ *gilt, daß falls* GAP_1 *für* (si, sg) *anwendbar ist, auch* GAP_2 *für* (si, sg) *anwendbar ist.*

Mit Hilfe dieser Ordnung ist es möglich den Abstraktionsgrad von abstrakten Plänen im bezug auf alle Planungsfälle einer Anwendung zu bestimmen. Diese Information kann dann

dazu benutzt werden, um eine Klassifikationshierarchie zu konstruieren, in der die abstrakten Pläne nach ihrem Anstraktionsgrad angeordnet sind. Dies wird dadurch realisiert, daß man die spezieller-als Relation $\sqsubseteq$ eines abstrakten Plans gegenüber allen übrigen abstrakten Plänen entscheidet. Mit dieser Information ist es dann möglich, den abstrakten Plan korrekt in die Klassifikationshierarchie einzuordnen. Ein Problem, das sich jedoch bei der Konstruktion einer solchen Klassifikationshierarchie stellt, ist, daß die Bedingung, ob ein gelernter abstrakter Plan spezieller ist als ein anderer abstrakter Plan nur schwer zu entscheiden ist. Wahrscheinlich handelt es sich hier sogar um ein im allgemeinen Fall unentscheidbares Problem.
Wir versuchen nun die $\sqsubseteq$-Halbordnung induktiv aus den Planungsfällen zu lernen. Hierzu wird zunächst aus einem neuen konkreten Planungsfall $F_j = (si_j, sg_j, P_j)$ die Menge aller gültigen abstrakten Pläne $B_j = \{GAP_1, ..., GAP_k\}$ mit Hilfe des PABS-Verfahrens generiert. Jeder der abstrakten Pläne in B_j ist nun anwendbar für diesen Planungsfall und umgekehrt ist jeder abstrakte Plan, der eine Abstraktion des voliegenden Planungsfalls F_j ist, auch in B_j enthalten. Die zweite Bedingung wird durch die Vollständigkeit des PABS-Verfahrens im bezug auf die Abstraktion von konkreten Planungsfällen garantiert. Die so gewonnene Menge B_j ist nun *ein* Trainingsbeispiel für einen induktiven Lernalgorithmus, der nun Hypothesen für eine mögliche $\sqsubseteq$-Relation zwischen allen bisher vorgekommenen abstrakten Plänen bildet. Man beachte hierbei, daß die $\sqsubseteq$-Relation selbst für die abstrakten Pläne, die in den einzelnen Trainingsbeispielen B_j vorkommen, nicht bekannt ist. Aufgrund dessen, daß der PABS-Algorithmus jedoch immer alle gültigen, abstrakten Pläne für einen gegebenen konkreten Planungsfall liefert, ist sichergestellt, daß falls $GAP \in B_j$ gilt, daß dann auch für alle GAP' mit $GAP' \sqsupseteq GAP$ gilt, daß $GAP' \in B_j$. Diese wichtige Information kann nun im folgenden ausgenutzt werden, um eine Klassifikationshierarchie induktiv inkrementell zu lernen. Hierbei soll zu jedem Zeitpunkt – also nach jeder Sequenz von Planungsfällen $\mathcal{F} = (F_1, \ldots, F_n)$ und zugehörigen abstrakten Planmengen $\mathcal{B} = (B_1, ..., B_n)$ – ein Klassifikationsbaum entstehen, der alle abstrakten Pläne aus $\mathcal{B}$ einordnet und für alle bislang gesehenen Planungsfälle $\mathcal{F}$ bzgl. der $\sqsubseteq$ - Relation korrekt ist. Der Baum, der als Hypothese für die $\sqsubseteq$-Relation zu einem Zeitpunkt durch den Lernalgorithmus bestimmt wird, soll also korrekt bzgl. der folgenden abgeschwächten $\sqsubseteq_{\mathcal{F}}$ - Relation sein.

Definition 3.2 *$GAP_1 \sqsubseteq_{\mathcal{F}} GAP_2$ (GAP_1 ist für die Menge von Planungsfällen $\mathcal{F}$ spezieller als GAP_2) gdw für alle konkreten Planungsfälle $F_i = (si_i, sg_i, P_i)$ aus $\mathcal{F}$ gilt: falls GAP_1 für (si_i, sg_i, P_i) anwendbar ist, so ist auch GAP_2 für (si_i, sg_i, P_i) anwendbar.*

Mit Hilfe dieser Ordnung ist es möglich den Abstraktionsgrad von abstrakten Plänen im bezug auf alle bisher bekannten Planungsfälle zu bestimmen. Da man sich hier nur auf die bereits bekannten abstrakten Pläne und die bekannten konkreten Planungsfälle beschränkt, ist diese Relation verhältnismäßig einfach zu entscheiden. Die Information, ob ein beliebiger abstrakter Plan spezieller ist als ein anderer, wird nun benutzt, um eine Klassifikationshierarchie inkrementell zu konstruieren, die für die bisher betrachteten Planungsfälle korrekt ist. Hierzu wurde ein Algorithmus entwickelt [Bergmann and W.Wilke, 1993], der versucht, die abgeschwächte „spezieller als“ Ordnung-$\sqsubseteq_{\mathcal{F}}$ und somit eine Klassifikationshierarchie für abstrakte Pläne induktiv aus den Planungsfällen zu lernen. Eine *Klassifikationshierarchie* für abstrakte Pläne und deren Korrektheit läßt sich nun wie folgt definieren:

Definition 3.3 *Eine Klassifikationshierarchie für abstrakte Pläne ist ein Baum mit der Wurzel K_0 und den Knoten K_i $(i = 1, \ldots, m)$. Jedem Knoten ist eine Menge von abstrakten Plänen $\mathcal{GAP}_i$ zugeordnet. Damit die Klassifikationshierarchie bezüglich einer Beispielmenge $\mathcal{B} = \{B_1, ..., B_n\}$ korrekt ist, werden folgende Bedingungen gefordert:*

- $\forall_{j=1..n}\forall_{i=0..m}(\mathcal{GAP}_i \cap B_j = \emptyset$ *oder* $\mathcal{GAP}_i \subseteq B_j)$, *d.h. für alle Trainingsbeispiele und jeden Knoten im Baum sind entweder immer alle abstrakten Pläne für ein Beispiel gültig*[2] *oder aber keiner. Es ist also nicht erlaubt, daß in einem Knoten für ein Beispiel ein abstrakter Plan gilt und ein anderer abstrakter Plan nicht gilt.*

- $\forall_{j=1..n}\forall_{i,l=0..m}$ *wenn* K_l *Nachfolger von* K_i *und* $\mathcal{GAP}_l \subseteq B_j$ *dann auch* $\mathcal{GAP}_i \subseteq B_j$, *d.h. für alle Trainingsbeispiele gilt, daß wenn die Pläne eines Knotens für ein Beispiel gültig sind, dann sind auch die Pläne des Vorgängerknotens gültig. Diese Bedingung fordert die Korrektheit des Baumes bzgl.* $\sqsubseteq_{\mathcal{F}}$.

Da in einem Knoten der Klassifikationshierarchie eine Menge von abstrakten Plänen vorkommen kann, kann so die Gleichheit von GAP's bezüglich der $\sqsubseteq_{\mathcal{F}}$ Relation ausgedrückt werden. Dies ist notwendig, da die Klassifikationshierarchie die abgeschwächte spezieller als Ordnung repräsentiert. Da diese Ordnung nur die bisher bekannten Planungsfälle und die aus diesen Fällen hervorgehenden abstrakten Pläne berücksichtigt, ist es eventuell nicht möglich die Relation zwischen zwei abstrakten Plänen zu entscheiden. Diese Pläne werden dann als gleich angesehen, da sie immer auf die selben Planungsfälle anwendbar sind. Die Gleichheit muß aber auf jeden Fall in einer Klassifikationshierarchie repräsentierbar sein, da es in Anwendungen immer vorkommen kann, daß zwei verschiedene abstrakte Pläne auf exakt die gleichen konkreten Planungsfälle anwendbar sind.
Bei der Suche nach einem abstrakten Plan zur Lösung eines konkreten Planungsproblems wird der Baum in Vorordnung durchlaufen. Dabei wird versucht, zuerst ein Plan zum Lösen des Planungsproblems zu nehmen, der möglichst speziell ist, also tief im Baum steht. Ist eine Verfeinerung eines solchen speziellen Plans nicht möglich, werden die allgemeineren Pläne, die sich auf dem Suchweg im Baum zu diesem Plan auf das Planungsproblem re-instanziieren ließen, versucht zu verfeinern. Die Benutzung des speziellsten anwendbaren Plans soll den benötigten Aufwand bei der Verfeinerung möglichst gering halten.

3.2 Das Lernen von Klassifikationshierarchien: Ein Beispiel

Um die bislang vorgestellten Überlegungen zu verdeutlichen, wird nun ein einfaches Beispiel vorgeführt. Hierbei werden fünf konkrete Planungsfälle $F_1, ..., F_5$ betrachtet. Für jeden dieser Planungsfälle gibt es nun eine Menge von generalisierten abstrakten Plänen, die vom PABS-Verfahren generiert werden. Insgesamt gibt es fünf[3] verschiedene abstrakte Pläne $GAP_1, ..., GAP_5$, die vorkommen können.
Nun gelten natürlich nicht alle abstrakten Pläne für alle konkreten Planungsfälle. Die genaue Zuordnung der gültigen abstrakten Pläne zu den konkreten Fällen ist in Tabelle 3.1 dargestellt.

[2]Gültig heißt hier, daß die Pläne sowohl re-instanziierbar, als auch verfeinerbar sind
[3]Dies sind hier nur zufällig genauso viele wie konkrete Probleme

	GAP_1	GAP_2	GAP_3	GAP_4	GAP_5
F_1	X	X			
F_2	X	X	X		
F_3	X	X	X		
F_4	X			X	
F_5	X			X	X

Tabelle 3.1: Ein Beispiel zur Anwendbarkeit der abstrakten Pläne für die konkreten Planungsfälle

Man sieht hier schon an der Tabelle, daß der abstrakte Plan GAP_1 für alle konkreten Planungsfälle gültig ist. Dieser Plan stellt den allgemeinsten Plan dar und ist somit immer für alle Planungsprobleme anwendbar. Im folgenden soll an einem kleinen Beispiel erläutert werden, welche Hypothesen für eine Klassifikationshierarchie bei einer gegebenen Beispielmenge in Frage kommen. Hierzu wollen wir annehmen, daß die konkreten Planungsfälle F_1, F_2 und F_5 gegeben sind und nach Anwendung des PABS-Verfahrens drei Trainingsbeispiele für den Lernalgorithmus zur Verfügung stehen, nämlich: $B_1 = \{GAP_1, GAP_2\}$, $B_2 = \{GAP_1, GAP_2, GAP_3\}$, $B_3 = \{GAP_1, GAP_4, GAP_5\}$.
Betrachtet man nun, welche Hierarchien gemäß Definition 3.3 für diese Beispiele korrekt sind (bzgl. $\sqsubseteq_{\{F_1,F_2,F_5\}}$), so stellt man fest, daß es insgesamt 6 verschiedene mögliche Hierarchien und damit $\sqsubseteq_{\mathcal{F}}$-Relationen gibt. Hierbei nimmt man an, daß es einen abstraktesten Plan gibt, der für jedes Problem anwendbar ist. Von diesen sechs Hierarchien sind beispielhaft vier in Abbildung 3.1 dargestellt. Baum (a) repräsentiert dabei die allgemeinste Hierarchie,

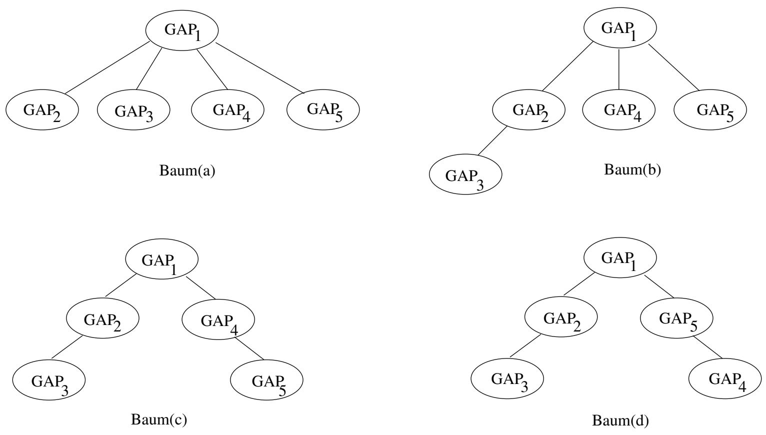

Abbildung 3.1: Eine Auswahl von möglichen Klassifikationsbäumen für F_1, F_2, F_5

da er keine Ordnung postuliert, mit Ausnahme des Bezuges zum abstraktesten Plan. Ein solcher Baum ist im Prinzip für jede Trainingsmenge korrekt, jedoch nicht sinnvoll, da die den Suchraum einschränkende Wirkung des Baumes fast vollständig verloren geht. Aufgrund der Beispiele B_1 und B_2 ist die Anordnung $GAP_3 \sqsubseteq GAP_2 \sqsubseteq GAP_1$ möglich, die in den Bäumen (b), (c) und (d) realisiert ist. Bezüglich GAP_4 und GAP_5 ist aufgrund der bisherigen Beispiele sowohl die Anordnung $GAP_4 \sqsubseteq GAP_5$ als auch $GAP_5 \sqsubseteq GAP_4$ möglich.

Kommt nun jedoch F_4 als weiterer Planungsfall hinzu, so ist die Anordnung $GAP_4 \sqsubseteq GAP_5$ und somit Baum (d) widersprüchlich.
In folgenden Abschnitt wird nun der Lernalgorithmus zum Aufbau von Klassifikationshierarchien erläutert sowie seine Korrektheit begründet. Der Algorithmus arbeitet vollständig inkrementell, d.h. er hat eine aktuelle Hierarchie und bekommt ein neues Trainingsbeispiel, also eine Menge von abstrakten Plänen. Der Algorithmus paßt dann die Hierarchie an das neue Trainingsbeispiel so an, daß die neue Hierarchie sowohl korrekt bzgl. des neuen Beispiels ist, als auch korrekt bzgl. aller vorherigen Beispiele bleibt. Wie bereits zu sehen war, gibt es in der Regel für eine Folge von Trainingsbeispielen mehrere mögliche korrekte Ordnungen und somit korrekte Bäume. Diese können aber aus Komplexitätsgründen nicht alle als Hypothesen[4] mitgeführt werden, da ansonsten die Hypothesenmenge zu groß wäre. Deshalb führt der Algorithmus eine Art Hill-Climbing [Rich, 1988; Pearl, 1984; Langley *et al.*, 1987] durch und hält nur die "vielversprechendsten" Ordnungen fest. Ordnungen, die "tief" sind, wie z.B. Baum (c) in Abbildung 3.1, werden dabei als besser betrachtet als "flache" wie z.B. Baum (a). Der Algorithmus führt nun zu jedem Zeitpunkt immer genau einen Baum mit, der jedoch mehrere Ordnungen dadurch reflektieren kann, daß an einem Knoten mehrere abstrakte Pläne stehen können (siehe Definition 3.3). An einem solchen Mehrfachknoten wird dann keine Aussage über die Ordnung der Pläne in diesem Knoten gemacht. Im Verlauf des Lernprozesses kann und wird sich jedoch ein solcher Mehrfachknoten teilen und sich die Ordnung spezialisieren, sobald dies aufgrund der Beispiele erforderlich ist.

3.3 Die Beschreibung des Algorithmus in Pseudocode

Der Toplevel-Algorithmus `LearnHierarchy(`K_0`,`B_{neu}`)`, erhält als Eingabe die Wurzel des aktuellen Baums, sowie das neue Beispiel.

PROZEDUR LearnHierarchy(K_0, B_{neu})
BEGIN
  Update($K_0, B_{neu}, true, K_0$)
  $B_{Rest} := B_{neu} - \bigcup_{i=0}^{m} GAP_i$
  IF $B_{Rest} \neq \emptyset$ **THEN**
    Insert(K_0, B_{Rest}, B_{neu})
END

Zuerst wird dann mit dem Prozeduraufruf `Update(`$K_0, B_{neu}, true, K_0$`)`[5] die Konsistenz des alten Baumes mit dem neuen Beispiel geprüft und gegebenenfalls wird der Baum so umgestaltet, daß er sowohl mit den alten Beispielen, als auch mit dem neuen Beispiel gemäß

[4]Bei induktiven Lernverfahren wird eine Menge von Hypothesen über das Lernziel gebildet. Aufgrund neuer Beispiele wird versucht, diese Hypothesenmenge suksessive zu verkleinern, um eine Beschreibung des Lernziels zu erlangen.

[5]*true* ist hier der Initialwert für ein Flag, das anzeigt, ob ein vorheriger Knoten noch erfüllt war (sie Beschreibung von `Update`)

Definition 3.3 korrekt ist. Danach werden die neuen abstrakten Pläne, die sich noch nicht im Baum befanden, mit der Prozedur `Insert`(K_0, B_{Rest}, B_{neu}) in den Baum eingefügt.

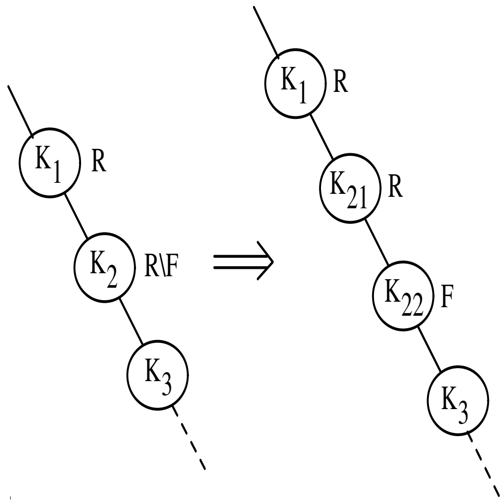

Abbildung 3.2: Das Aufteilen eines Mehrfachknotens in der Abstraktionshierarchie: R bezeichnet einen Knoten mit gültigen Plänen, F einen Knoten mit ungültigen Plänen; R/F ist ein gemischter Knoten

Um beim Eintreffen eines neuen Beispiels zu überprüfen, ob der aktuelle Baum korrekt ist, muß dieser (in Vorordnung) durchlaufen werden. Hierbei wird zunächst geprüft, ob die abstrakten Pläne an einem Knoten entweder alle für das aktuelle Beispiel anwendbar oder aber alle nicht anwendbar sind. Liegt jedoch der Fall vor, daß ein Knoten sowohl gültige als

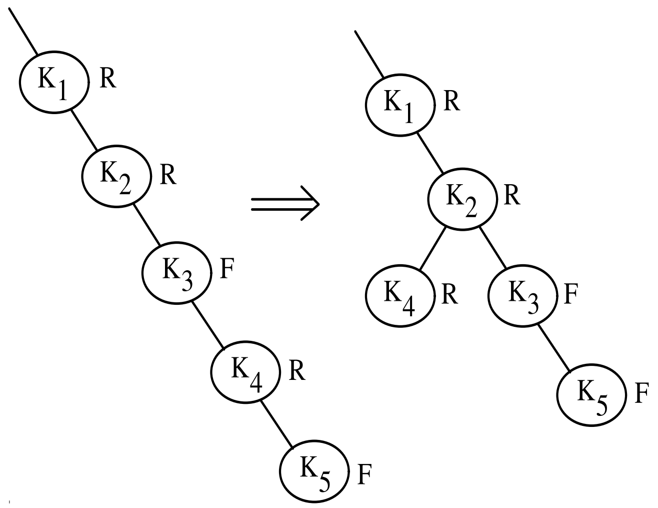

Abbildung 3.3: Das Hochziehen eines (Teil)-Knotens zur Korrektur der Abstraktionshierarchie

auch ungültig Pläne enthält, so muß der Baum an dieser Stelle "repariert" werden, damit die erste Bedingung für die Korrektheit der Klassifikationshierarchie nach der Definition 3.3

wieder erfüllt ist. Hierzu wird der "gemischte" Knoten in zwei neue Knoten geteilt, wie in Abbildung 3.2 verdeutlicht wird. Die abstrakten Pläne, die dem neuen oberen Knoten (K_{21}) zugewiesen werden, sind alle Pläne aus K_2, die für das aktuelle Beispiel gelten, und alle Pläne, die dem neuen unteren Knoten (K_{22}) zugewiesen werden, sind alle die Pläne aus K_2, die für das aktuelle Beispiel nicht gelten. Wie man sich leicht überlegen kann, erzeugt diese Operation einen korrekten Teilpfad K_1, K_{21}, K_{22} in bezug auf das neue Beispiel und erhält die Korrektheit bezüglich aller vorherigen Beispiele. In einem weiteren Schritt muß nun der zweite Teil der Korrektheit der Klassifikationshierarchie nach der Definition 3.3 überprüft werden. Diese erfordert einen Test darauf, ob die abstrakten Pläne an einem Knoten für das aktuelle Beispiel gültig sind, obwohl die abstrakten Pläne an einem Vorgängerknoten ungültig sind. Ist dies der Fall, so ist ebenfalls die Klassifikationshierarchie nicht korrekt und der Baum muß korrigiert werden. Diese Korrektur erfolgt nun dadurch, daß der (fehlerhafte) Knoten, dessen Pläne gültig sind, herausgenommen wird und als Nachfolger des ersten Vorgängers eingefügt wird, dessen Pläne wieder gültig sind. Diese "Hochziehen" des Knotens an die erste passende Stelle ist in Abbildung 3.3 beispielhaft dargestellt. Hier ist der Knoten K_4 gültig, obwohl der Vorgänger K_3 ungültig ist. Folglich wird der Knoten K_4 entfernt und als Nachfolger von K_2 angehängt, da K_2 der erste Vorgängerknoten ist, dessen Pläne wieder gültig sind. Prinzipiell wäre es auch möglich, den Knoten noch weiter oben im Baum anzuhängen, z.B. an der Wurzel. Da wir aber den Baum so verändern möchten, daß er noch möglichst optimal (also tief) ist, wird hier immer die tiefste mögliche Stelle gewählt. Beim Betrachten neuer Trainingsbeispiele ist es jedoch möglich, daß der Knoten trotzdem weiter nach oben wandert. Dieses Hochziehen des Knotens macht nun den Baum für das aktuelle Beispiel wieder korrekt , im bezug auf die zweite Bedingung der Definition 3.3 und erhält ebenfalls die Korrektheit bezüglich aller früheren Beispiele. Im Folgenden ist der vollständige Algorithmus für die beschriebene `Update` Prozedur angegeben. Dieser Algorithmus realisiert im wesentlichen die beiden Operationen zur Erzeugung der Korrektheit des Baumes, die jedoch etwas ineinander verzahnt sind.

PROZEDUR Update($K_i, B_{neu}, Ok_{Flag}, K_{append}$)
BEGIN
IF Ok_{Flag} **THEN** (* Vorheriger Knoten war vollständig erfüllt *)
  IF $GAP_i \subseteq B_{neu}$ **THEN** (* Knoten ist vollständig erfüllt *)
    IF Es existieren keine Nachfolger von K_i **THEN RETURN**
    FOR ALL K_j Nachfolger von K_i **DO**
      Update($K_j, B_{neu}, true, K_i$)
  ELSE IF $GAP_i \cap B_{neu} = \emptyset$ **THEN** (* Knoten K_i ist vollständig unerfüllt *)
    IF Es existieren keine Nachfolger von K_i **THEN RETURN**
    FOR ALL K_j Nachfolger von K_i **DO**
      Update($K_j, B_{neu}, false, K_{append}$)
  ELSE (* Knoten enthält erfüllte und unerfüllte Elemente *)
    Teile Knoten K_i in zwei neue Knoten K_u und K_n mit:
      a) Vorgänger von K_i wird Vorgänger von K_u
      b) K_n wird Nachfolger von K_u
      c) Alle Nachfolger von K_i werden Nachfolger von K_n
      d) $GAP_u := GAP_i \cap B_{neu}$
      e) $GAP_n := GAP_i - GAP_u$
    IF Es existieren keine Nachfolger von K_i **THEN RETURN**

FOR ALL K_j Nachfolger von K_n **DO**
Update($K_j, B_{neu}, false, K_u$)
ELSE (* Vorgängerknoten war vollständig unerfüllt *)
IF $GAP_i \cap B_{neu} = \emptyset$ **THEN** (* Knoten K_i ist vollständig unerfüllt *)
IF Es existieren keine Nachfolger von K_i **THEN RETURN**
FOR ALL K_j Nachfolger von K_i **DO**
Update($K_j, B_{neu}, false, K_{append}$)
ELSE (* K_i enthält erfüllte Elemente *)
1. Erzeuge einen Knoten K_{neu} als Nachfolger von K_{append} mit:
$GAP_{neu} := GAP_i \cap B_{neu}$
2. Setze $GAP_i := GAP_i - B_{neu}$
IF $GAP_i = \emptyset$ **THEN** (* leeren Knoten entfernen *)
1. Entferne K_i
2. Die Nachfolger von K_i werden die Nachfolger von $K_i's$ Vorgänger
3. **FOR ALL** K_j Nachfolger von K_i **DO**
Update($K_j, B_{neu}, false, K_{append}$)
END

Nachdem nun also die Klassifikationshierarchie bezüglich des neuen Beispiels korrekt ist, müssen nun noch alle abstrakten Pläne des neuen Beispiels, die noch nicht im Baum enthalten sind, hinzugefügt werden. Dieses Einfügen kann nun im Prinzip an einer beliebigen Stelle geschehen, solange die Hierarchie danach noch korrekt ist. Die Stelle des Einfügens ist jedoch entscheidend dafür, daß ein guter (möglichst tiefer) Baum entsteht. Zum Bestimmen eines guten Einfügepunktes wird eine Heuristik verwendet, die einen möglichst tiefen Baum erzeugen soll. Hierzu wird, ausgehend von der Wurzel des Baumes, solange im Baum hinabgestiegen bis die abstrakten Pläne eines Knotens für das aktuelle Beispiel nicht mehr anwendbar sind oder ein Blattknoten erreicht wurde. Ist eine solche Stelle im Baum gefunden, wird ein neuer Knoten angehangen und ihm die neuen abstrakten Pläne zugewiesen. Als Kriterium dafür, welcher Pfad weiter verfolgt wird, falls mehrere Nachfolgerknoten anwendbare Plänen besitzen, wird die Anzahl der Pläne an den Nachfolgerknoten herangezogen. Es wird der Nachfolgerknoten gewählt, der die meisten Pläne besitzt, da an solchen Knoten noch der größte Freiraum für die Anordnung der Pläne durch Spaltung des Knotens besteht und somit die Möglichkeit einer ungünstigen Anordnung minimiert wird. Der vollständige Algorithmus für das Einsortieren der neuen Pläne ist im Folgenden dargestellt.

PROZEDUR Insert(K_i, B_{Rest}, B_{neu})
BEGIN
IF Es existieren keine Nachfolger von K_i **THEN**
Erzeuge einen neuen Knoten K_j als Nachfolger von K_i mit:
$GAP_j := GAP_{Rest}$
ELSE
Wähle einen Nachfolger K_j von K_i mit $\mid GAP_j \cap B_{neu} \mid$ maximal
IF $\mid GAP_j \cap B_{neu} \mid > 0$ **THEN** (* Steige tiefer in den Baum ein *)
Insert(K_j, B_{Rest}, B_{neu})
ELSE

Erzeuge einen Knoten K_j als Nachfolger von K_i mit:
$GAP_j := GAP_{Rest}$
END

Die Funktionsweise des Algorithmus zum inkrementellen Lernen von Abstraktionshierarchien wird im folgenden Abschnitt an einem Beispiel verdeutlicht.

3.4 Die Verdeutlichung des Algorithmus an einem Beispiel

Um diesen Lernalgorithmus an einem Beispiel zu verdeutlichen, wird er für die Planungsfälle F_1, F_2, F_3, F_4, F_5 und die daraus resultierenden abstrakten Pläne GAP_1, GAP_2, GAP_3, GAP_4, GAP_5 aus Tabelle 3.1 angewandt. Hierbei wollen wir annehmen, daß die Trainingsbeispiele in der Reihenfolge F_2, F_5, F_1, F_4, F_3 mit den dazugehörenden abstrakten Planmengen vorgegeben werden. Abbildung 3.4 zeigt die resultierende Folge von Klassifikationsbäumen. Nachdem der erste Planungsfall F_2 bearbeitet wurde, wird genau ein Knoten

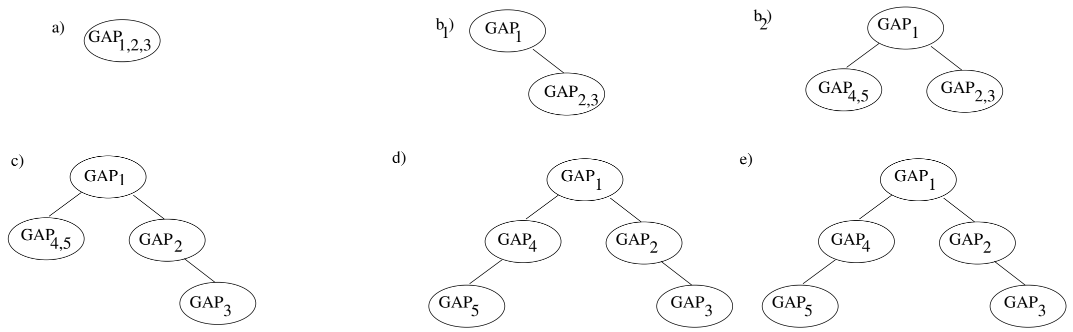

Abbildung 3.4: Eine Beispielfolge von Klassifikationshierarchien für das Beispiel zum inkrementellen Lernen von Abstraktionshierarchien

erzeugt, der alle abstrakten Pläne (GAP_1, GAP_2, GAP_3), die aus diesem Planungsfall hervorgegangen sind, enthält. Bei dem zweiten Planungsfall (F_5) stellt nun die `update` Prozedur fest, daß der bisherige Knoten sowohl anwendbare Pläne (nämlich GAP_1) enthält als auch nicht anwendbare Pläne (GAP_1 und GAP_2). Deshalb wird der Knoten zunächst geteilt (siehe Baum b1). Danach werden beim `Insert` die fehlenden Pläne GAP_4 und GAP_5 zusammen an einem Knoten angehangen (siehe Baum b2). Das Bearbeiten des Planungsfalls F_1 führt nun weiter dazu, daß der Knoten mit GAP_2 und GAP_3 gesplittet wird (Baum c). Weitere Pläne kommen nun nicht mehr hinzu, da bereits alle Pläne aus dem (und allen folgenden) Planungsfällen im Baum vorhanden sind. Der Planungsfall F_4 bewirkt nun lediglich noch ein Splitten eines Knotens (siehe Baum d), während der letzte Planungsfall F_3 den Baum nun überhaupt nicht mehr verändert.
Der Algorithmus liefert für jede Reihenfolge von Trainingsbeispielen, die hier gewählt wurden, letztlich denselben Klassifikationsbaum. Die in diesem Beispiel verwendete Reihenfolge

wurde lediglich deshalb gewählt, um die Operationen des Algorithmus besser erläutern zu können. Es gibt jedoch durchaus auch andere Beispiele, bei denen der resultierende Baum von der Reihenfolge der präsentierten Beispiele stark abhängt.

In diesem Abschnitt wurde der inkrementelle Lernalgorithmus vorgestellt, der als Bestandteil des PARIS-Planungssystems eine Abstraktionshierarchie aus erklärungsbasiert abstrahierten Plänen aufbaut. Hierbei wurde im wesentlichen eine Halbordnung erworben, die a-priori nicht bekannt war und nur indirekt aus der Struktur der Trainingsbeispiele abgeleitet werden konnte. In diesem Sinne ist der vorgestellte Algorithmus durchaus auf ähnliche Situationen übertragbar, in denen nicht bekannte oder unentscheidbare Halbordnungen (z.B. Generalisierungshierarchien) aufgebaut werden müssen. Dies wird zum Beispiel auch mit dem Verfahren der formalen Begriffsanalyse [Wille, 1992; Ganter, 1987] realisiert. Es handelt sich dabei auch um ein algorithmisches Verfahren zur Datenanalyse, mit dem es unter anderem möglich ist, Partialordnungen zu erlernen. Des weiteren gibt es bei dem dargestellten Verfahren starke Bezüge zu Clusterungsalgorithmen für Begriffsbildungsaufgaben [Gennari *et al.*, 1989; Fisher, 1987]. Durch die gelernte Hierarchie werden nämlich auch indirekt konkrete Planungsfälle in hierarchische Cluster angeordnet. So gibt es zu jedem Knoten in der Hierarchie immer auch ein Cluster von Planungsfällen, die in dem Algorithmus lediglich nicht gespeichert werden, da sie für das weitere Problemlösen nicht benötigt werden. Die Menge der Planungsfälle, die dabei einem Knoten zugeordnet ist, ist hierbei immer eine Teilmenge der Planungsfälle des Vorgängerknotens. So ist beispielsweise dem Wurzelknoten der Hierarchie (e) aus Abbildung 3.4 die komplette Menge von Planungsfällen $\{F_1, ..., F_5\}$ zugeordnet und dem Nachfolgerknoten GAP_4 die Planungsfälle $\{F_4, F_5\}$ sowie dem Nachfolgerknoten GAP_2 die Planungsfälle $\{F_1, ..., F_3\}$. Die abstrakten Pläne an den Knoten sind nun immer eine gemeinsame Abstraktion (shared abstraction) [Bergmann, 1993b] der zugehörigen Planungsfälle. Mit anderen Worten, der vorgestellte Algorithmus clustert konkrete Planungsfälle nach den Kriterium, ob es gemeinsame Abstraktionen gibt.

Kapitel 4

Der Algorithmus zur Optimierung der Abstraktionshierarchie

Durch die Konstruktion von Abstraktionshierarchien, wie sie im vorherigen Kapitel erklärt wurden, konnte die Anzahl der zu betrachteten abstrakten Pläne gegenüber der linearen Anordnung verkleinert werden.[1] Es zeigte sich auch, daß ein speziellerer re-instanziierter Plan schneller zu verfeinern war, als ein abstrakterer re-instanziierter Plan. Man konnte aber feststellen, daß ein speziellerer Plan tendenziell durch mehr Relationen charakterisiert wurde als ein abstrakterer Plan. Dies führte dazu, daß die speziellen Pläne zwar schnell mit der konkreten Planungsfunktion zu verfeinern waren, die Problemlösezeit sich in der Summe nicht verbesserte, da für das Re-Instanziieren eines Plans mehr Zeit benötigt wurde. Da immer der speziellste Plan gesucht wird, mußten ab einer gewissen Tiefe des Baumes noch immer viele Pläne re-instanziiert werden.
Dieses führt zu der Idee, den Baum zur Problemlösung zu optimieren. Dies wurde dadurch erreicht, daß der Baum in den einzelnen Zweigen an der Stelle abgeschnitten wurde, ab der sich aufgrund der betrachteten Beispiele eine Verschlechterung der Problemlösezeit ergab. Beim Durchlaufen des Baumes zur Suche eines abstrakten Plans, der sich auf ein konkretes Planungsproblem verfeinern läßt, werden dann diese abgeschnittenen Zweige nicht mehr betrachtet. So kann erreicht werden, daß zu spezielle Pläne nicht mehr zum Problemlösen eingesetzt werden. Des weiteren wurden sowohl die abstrakten Pläne innerhalb eines Baumknotens als auch die Baumknoten selbst so umsortiert, daß Pläne mit einem schnelleren Problemlöseverhalten in der Vergangenheit zuerst zum Lösen neuer Probleme eingesetzt werden.
Die Optimierung der Abstraktionshierarchie versucht, den Erwartungswert der Problemlösezeit mit dieser Hierarchie zu minimieren. Hierbei wird nicht versucht, die Hierarchie für ein einzelnes Problem zu optimieren, sondern für ein für die Anwendung relevantes Problemspektrum. Da nur Wissen über die bisher bekannten Planungsfälle akquiriert werden kann, muß man die Annahme treffen, daß die bisher gesehenen Fälle repräsentativ für das spätere Problemlösen sind. Es muß also die Verteilung der in der Fallbasis vorkommenden Probleme, im bezug auf die Gesamtheit aller repräsentierbaren Probleme, der Verteilung der in der Zukunft zu erwartenden Probleme entsprechen. Dann ist es möglich, den Erwartungswert für die Problemlösezeit zukünftiger Probleme durch die Problemlösezeit

[1] Auf diesen Sachverhalt wird ausführlich im nächsten Kapitel bei der Bewertung der vorgestellten Verfahren eingegangen.

der bisher gesehenen Fälle abzuschätzen. Diese Übereinstimmung der bekannten Probleme mit den zukünftigen Problemstellungen hat einen wesentlichen Einfluß auf die Güte der Abschätzung des Erwartungswerts der Problemlösezeit einer Abstraktionshierarchie.

Definition 4.1 *Sei $\mathcal{P} = (P_1, \ldots, P_n)$ die Menge aller zu lösenden Planungsprobleme. So läßt sich der Erwartungswert der Problemlösezeit definieren als:*

$$T_{EW}(\mathcal{P}) := \sum_{i=1}^{n} Prob(P_i) * T_L(P_i)$$

Hierbei ist $Prob(P_i)$ die Wahrscheinlichkeit, mit der das Problem P_i gelöst werden soll und $T_L(P_i)$ die Zeit, die dafür benötigt wird.

Nun sind aber nicht alle Probleme bekannt, sondern durch die bekannten Fälle nur eine Stichprobe $\mathcal{P}'$. Somit läßt sich nur *eine Schätzung des exakten Erwartungswerts* der Problemlösezeit auf grund dieser Stichprobe berechnen, die sich ergibt aus:

$$T'_{EW}(\mathcal{P}') := \frac{1}{|\mathcal{P}'|} \sum_{i=1}^{|\mathcal{P}'|} T_L(P_i)$$

Durch die in diesem Kapitel vorgestellte Planbewertung und die damit verbundene Planselektion, ist der Plan zum Lösen eines Planungsproblems eindeutig festgelegt. Die Klassifikationshierarchie wird bei der Planselektion in Vorordnung durchlaufen. Durch die Planbewertung werden die Knoten der Hierarchie eindeutig angeordnet und die Pläne innerhalb eines Knotens ebenfalls. Der erste Plan der sich auf das Planungsproblem nach Definition 2.9 re-instanziieren läßt wird auch zum Verfeinern, also zum Lösen des Planungsproblems benutzt. Mit der oben diskutierten Relevanz der Planungsprobleme der Planbiliothek für die neuen zu lösenden Planungsprobleme, kann man bei der Schätzung des exakten Erwartungswerts nicht mehr die Planungsprobleme und die Wahrscheinlichkeit ihres Auftretens, sondern die einzelnen Pläne und die Wahrscheinlichkeit ihrter Anwendung betrachten. Hieraus resultiert eine modifizierte Schätzung des Erwartungswerts der Problemlösezeit aus:

Definition 4.2 *Sei $\mathcal{PLAN} = (Plan_1, \ldots, Plan_n)$ die Menge der Pläne, die sich in einer Abstraktionshierarchie befinden. Dann ergibt sich eine Schätzung des Erwartungswerts der Problemlösezeit der Abstraktionshierarchie aus:*

$$\widetilde{T}_{EW}(\mathcal{PLAN}) := \sum_{i=1}^{n} Prob(Plan_i) * T_L(Plan_i)$$

Hierbei ist $Prob(Plan_i)$ die Wahrscheinlichkeit, mit der ein Plan $Plan_i$ zum Problemlösen benutzt wird und $T_L(Plan_i)$ die Zeit, die dafür benötigt wird.

Diese Schätzung des exakten Erwartungswerts ist ein geeignetes Maß, um das Problemlöseverhalten einer Abstraktionshierarchie zu bewerten. Um nun ein möglichst gutes Problemlöseverhalten für neue Planungsprobleme zu erreichen, sollte die Schätzung des exakten Erwartungswerts der Problemlösezeit minimiert werden. Diese Minimierung des Erwartungswerts ist das Ziel des in diesem Kapitel vorgestellten Algorithmus.
Welcher Plan der Abstraktionshierarchie zum Problemlösen benutzt wird, ist abhängig von der

- Anordnung der abstrakten Pläne in einem Knoten
- Anordnung der Nachfolgerknoten eines Knotens
- Menge der Knoten in der Abstraktionshierarchie

Diese drei Kriterien haben einen Einfluß auf den Erwartungswert der Problemlösezeit der Abstraktionshierarchie. Dazu wird zuerst eine Ordnung der abstrakten Pläne in einem Knoten definiert, sodaß der angenäherte Erwartungswert der Problemlösezeit einer Abstraktionshierarchie minimal wird. Weiterhin wird eine Bewertung der Knoten gegeben, die es uns später ermöglicht, die Nachfolgerknoten eines Knotens der Abstraktionshierarchie so anzuordnen, daß der Erwartungswert für die Problemlösezeit einer Abstraktionshierarchie minimiert wird. Zur Berechnung des Erwartungswerts eines Knotens der Abstraktionshierarchie wird dann die Wahrscheinlichkeit für die Anwendung eines Knotens definiert. Die Bewertung der Kosten eines Knotens der Abstraktionshierarchie zusammen mit der Wahrscheinlichkeit seiner Anwendung kann dann dazu benutzt werden, die Menge der Knoten in der Abstraktionshierarchie so zu verkleinern, daß daraus ein minimaler Erwartungswert für das Problemlösen mit der Abstraktionshierarchie resultiert. Diese drei Operationen auf der Abstraktionshierarchie sollen für einen minimalen Erwartungswert der Problemlösezeit der Abstraktionshierarchie, im bezug auf die bekannten Planungsfälle sorgen.
Um eine Schätzung des Erwartungswerts der Problemlösezeit einer Abstraktionshierarchie zu berechnen, sind sowohl Informationen über das Problemlöseverhalten eines abstrakten Plans nötig als auch die Häufigkeit seiner Anwendung beim Problemlösen. Im ersten Abschnitt dieses Kapitels werden zuerst die Grundlagen zur Bewertung der Problemlösezeit von einzelnen abstrakten Plänen definiert. Darauf aufbauend wird dann das Problemlöseverhalten von Knoten mit abstrakten Plänen aus der Abstraktionshierarchie bewertet. So können dann Aussagen über die Zeiten beim Problemlösen gemacht werden, die ein Knoten verursacht, der sich in der Abstraktionshierarchie befindet. Anschließend wird die Berechnung für die Wahrscheinlichkeit hergeleitet, mit der ein Knoten zum Problemlösen benutzt wird. Mit Hilfe der so berechneten Kosten für einen Knoten und der Wahrscheinlichkeit seiner Anwendung, kann dann die Schätzung des Erwartungswerts der Problemlösezeit für eine Abstraktionshierarchie berechnet werden.
Im zweiten Abschnitt dieses Kapitels wird die Funktionsweise des Algorithmus zur Optimierung der Abstraktionshierarchie informell beschrieben und in Pseudocode dargestellt. Bei dieser Optimierung wird die Abstraktionshierarchie so umgestaltet, daß sich ein minimaler Erwartungswert für die Problemlösezeit der Abstraktionshierarchie ergibt. Genauer gesagt, wird der Erwartungswert der Problemlösezeit des Wurzelknotens der Abstraktionshierarchie minimiert. Zum Schluß wird der Algorithmus noch einmal durch ein ausführliches Beispiel veranschaulicht.

4.1 Die Bewertung von abstrakten Plänen und Knoten der Abstaraktionshierarchie

Um eine Schätzund des Erwartungswerts der Problemlösezeit zu berechnen, sind sowohl Informationen über das Problemlöseverhalten von abstrakten Plänen nötig als auch über die Kosten, die ein Knoten in der Abstraktionshierarchie verursacht. Um eine Abschätzung des

Aufwands bei der Problemlösung mit einem abstrakten Plan zu erlangen, werden die nach der PABS-Planabstraktionsmethodik gewonnenen abstrakten Pläne dazu benutzt, daß konkrete Planungsproblem zu lösen, aus dem sie hervorgegangen sind. Dabei werden die Zeiten, die die Problemlösekomponente zum Re-Instanziieren und zum Verfeinern mit der konkreten Planungsfunktion benötigt, gemessen. So lassen sich für einen konkreten Planungsfall F_i und für die Menge der daraus entstehenden abstrakten Pläne $B_i = \{GAP_1, \ldots, GAP_n\}$ folgende Zeiten messen, die man gewinnt, falls man mit den einzelnen Plänen GAP_j, $j = 1, \ldots, n$, das Planungsproblem F_i wieder löst.

Definition 4.3 *Sei GAP_j ein generalisierter, abstrakte Plan, der aus dem Planungsproblem F_i hervorging. Weiterhin sei $\mathcal{F} = (F_1, \ldots, F_n)$ eine Menge von Planungsfällen, die ebenfalls GAP_j als Abstraktion haben, dann ist:*

- *Tm_{ij} := die Zeit, die zum Re-Instanziieren des Plans GAP_j für (si_i, sg_i) des Planungsfalls F_i benötigt wird.*
- *Tr_{ij} := die Zeit, die zum Verfeinern des Plans GAP_j auf den Planungsfall F_i mit der konkreten Planungsfunktion benötigt wird.*
- *$Tm_j(\mathcal{F}) := \frac{1}{|n|} \sum_{i=1}^{n} Tm_{ij}$ die mittlere Re-Instanziierungszeit des Plans GAP_j für die Probleme aus den Fallmenge $\mathcal{F} = (F_1, \ldots, F_n)$*
- *$Tr_j(\mathcal{F}) := \frac{1}{|n|} \sum_{i=1}^{n} Tr_{ij}$ die mittlere Verfeinerungszeit des Plans GAP_j für die Problemmenge $\mathcal{F} = (F_1, \ldots, F_n)$*

Diese Zeiten können nun dafür eingesetzt werden, daß Problemlöseverhalten von abstrakten Plänen im bezug auf Planungsfälle oder Mengen von Planungsfällen zu bewerten.
Außerdem kann mit Hilfe der in Definition 4.3 gemessenen Werte die Anordnung der abstrakten Pläne in einem Knoten optimiert werden. Eine optimale Anordnung von abstrakten Plänen in einem Knoten zeichnet sich dadurch aus, daß Probleme mit einem besseren Problemlöseverhalten zuerst zum Problemlösen benutzt werden. Um die Qualität der Pläne bei der Problemlösung zu bewerten, sind die in Definition 4.3 gegebenen Zeiten geeignet, da sie angeben, wie lange die Problemlösung mit einem abstrakten Plan bei den bisher bekannten Planungsproblemen gedauert hat.
Die Idee bei der optimalen Anordnung der abstrakten Pläne ist es, die abstrakten Pläne nicht nach der Anzahl ihrer abtrakten Operatoren zu ordnen, sondern nach der Summe aus den gemessenen Re-Instanziierungs- und Verfeinerungszeiten beim Problemlösen. Die angestellte Überlegung sei hier noch einmal durch ein Beispiel erläutert. Dazu betrachten wir folgende Tabelle mit den aus einem Planungsfall F_1 hervorgehenden abstrakten Plänen, den gemessenen Zeiten gemäß Definition 4.3 und ihren abstrakten Operatorenlängen.

Nach dem Abstrahieren von Planungsfall F_1 würden sich die abstrakten Pläne {1,2,3} in einem Knoten des Baumes befinden. Nach ihrer abstrakten Operatorlänge sortiert würde beim erneuten Problemlösen der abstrakte Plan 2 benutzt. Löst man nun den Planungsfall F_1 erneut, würde man mit Plan 2 nach (4140 + 2080 =) 6220 msec. eine Lösung bekommen. Sortiert man nun die Pläne nach der Summe aus den bisher gemessenen Re-Instanziierungs- und Verfeinerungszeiten, würde der abstrakte Plan 3 zum Problemlösen eingesetzt. Benutzt

abstrakter Plan	T_m	T_r	Anzahl der abstrakten Operatoren
GAP_1	4100	2350	4
GAP_2	4140	2080	5
GAP_3	520	2570	2

Tabelle 4.1: Die zum konkreten Planungsfall F_1 gehörenden abstrakten Pläne und die dazugehörenden Re-Instanziierungs- und Verfeinerungszeiten in CPU-Millisekunden

man nun den Plan 3 zum Problemlösen ergibt sich eine Gesamtproblemlösezeit von (520 + 2570 =) 3090 msec. für den Planungsfall F_1. Da bisher nur der Planungsfall F_1 bekannt ist, kann die Abstraktionshierarchie auch nur auf die Lösung dieses Planungsfalls optimiert werden, da a-priori über das Problemlöseverhalten der abstrakten Pläne im bezug auf neue konkrete Planungsprobleme keine Informationen bekannt sein können. Diese Sortierung der abstrakten Pläne innerhalb eines Knotens muß nach jeder Abstraktion eines konkreten Planungsproblems neu durchgeführt werden, da sich theoretisch für jede Planmenge der Wert für T_m und T_r ändern kann.
Um die Anordnung der abstrakten Pläne in einem Knoten zu optimieren, läßt sich nun eine *Ordnung über den abstrakten Plänen* in einem Knoten definieren durch:

Definition 4.4 *Sei*
K ein Knoten des Abstraktionsbaumes, $GAP(K) = \{GAP_1, \ldots, GAP_m\}$ die Menge der generalisierten, abstrakten Pläne, die sich im Knoten K befinden und $\mathcal{F}_K$ die Menge der konkreten Planungsfälle, aus denen die abstrakten Pläne des Knotens $GAP_i, i = 1, \ldots, m$, entstanden sind. Dann ist eine Ordnung $<_{GAP}$ der GAP_i innerhalb des Knotens definiert durch:

$$GAP_i <_{GAP} GAP_j \ \underline{gdw.} \ Tm_i(\mathcal{F}_K) + Tr_i(\mathcal{F}_K) < Tm_j(\mathcal{F}_K) + Tr_j(\mathcal{F}_K)$$

Werden die abstrakten Pläne in einem Knoten nach dieser Ordnung angeordnet, werden die Pläne mit einer geringeren Summe beobachteter Re-Instanziierungs- und Verfeinerungszeiten, im bezug auf die bereits bekannten Planungsprobleme, zuerst zum Problemlösen eingesetzt. Dies liegt daran, daß, falls sich mehrere Pläne in einem Knoten befinden, bei der Suche nach einem Plan zur Lösung eines neuen konkreten Planungsproblems die Pläne nach ihrer Anordnung im Knoten benutzt werden. Wie diese Sortierung im Baum erzeugt wird, ist im nächsten Abschnitt dieses Kapitels bei der Implementierung des Algorithmus zur Optimierung der Problemlösezeit beschrieben.
Mit Hilfe der in Definition 4.3 gegebenen Zeiten und der Ordnung aus Definition 4.4 von abstrakten Plänen, ist es nun möglich, die Kosten zu definieren, die ein Knoten verursacht, falls er zum Problemlösen herangezogen wird.

Definition 4.5 *Sei K ein Knoten des Abstraktionsbaumes, $GAP(K) = \{GAP_1,\ldots,GAP_m\}$ die Menge der generalisierten, abstrakten Pläne, die sich im Knoten K befinden, $\mathcal{F}_K$ die Menge der konkreten Planungsfälle, die mit den $GAP_i, i = 1, \ldots, m$, gelöst werden können und sei $GAP_{min}, min \in \{1, \ldots, m\}$ minimal bezüglich $<_{GAP}$, dann sind*

- $Tm_K := Tm_{min}(\mathcal{F}_\mathcal{K})$ *die Re-Instanziierungskosten für Knoten K*
- $Tr_K := Tr_{min}(\mathcal{F}_\mathcal{K})$ *die Verfeinerungskosten für Knoten K*

Die in Definition 4.5 gemessenen Werte können nun dazu benutzt werden, die Anordnung der Nachfolgerknoten eines Knotens der Abstraktionshierarchie zu optimieren. Diese Optimierung wird in dem Abschnitt über die Beschreibung der Algorithmen in Pseudocode gegeben, da alle nötigen Voraussetzungen für die Durchführung der Optimierung der Anordnung der Nachfolgerknoten noch nicht gegeben sind.
Somit ist sowohl eine Definition der Kosten für das Problemlösen mit einem bestimmten abstrakten Plan als eine Definition der die Kosten des Problemlösens mit einem bestimmten Knoten der Abstraktionshierarchie gegeben. Dies ist für eine Minimierung des Erwartungswerts der Problemlösezeit jedoch nicht ausreichend. Es ist weiterhin wichtig, mit welcher Häufigkeit ein abstrakter Plan benutzt wird, um einen konkreten Planungsfall zu lösen. Dazu ist es notwendig zu berechnen, mit welcher Häufigkeit ein Knoten der Abstraktionshierarchie zum Problemlösen eingesetzt wird.

4.1.1 Die Wahrscheinlichkeit der Anwendung eines Knotens der Abstraktionshierarchie

Um die Schätzung des Erwartungswerts einer Abstraktionshierarchie zu berechnen, sind sowohl die im vorherigen Abschnitt definierten Kosten eines Knotens zur Problemlösung relevant als auch die Wahrscheinlichkeit, mit der eine Planmenge eines Knotens zum Problemlösen benutzt wird. Der Erwartungswert eines Knotens ergibt sich aus dem Produkt der Wahrscheinlichkeit seiner Anwendbarkweit und den Kosten, die er beim Problemlösen verursacht. Der Erwartungswert der Abstraktionshierarchie ergibt sich aus dem Erwartungswert der Problemlösezeit des Wurzelknotens. Um nun die Menge der Knoten der Abstraktionshierarchie so einzuschränken, daß der Erwartungswert des Wurzelknotens minimal wird, wird der Erwartungswert für die Problemlösezeit der übrigen Knoten berechnet. Die Knoten der Abstraktionshierarchie, die eine Verschlechterung des Erwartungswertes des Wurzelknotens bewirken, werden vom Baum, der die Abstraktionshierarchie repräsentiert, abgeschnitten. Durch diese Beschneidung der Abstraktionshierarchie wird die Menge der Pläne, die zum Problemlösen benutzt werden, so eingeschränkt, daß sich ein minimaler Erwartungswert für den Wurzelknoten und somit für die Problemlösezeit der Abstraktionshierarchie ergibt. Um die nun angestellten Überlegungen zu verdeutlichen, betrachten wir den Baum aus Abbildung 4.1. Für diesen Baum gilt $N_i \sqsubseteq_\mathcal{F} N_0$ für $i = 1, \ldots, r$, d.h. daß die Pläne der Knoten N_i für $i = 1, \ldots, r$ spezieller sind, nach Definition 3.2, als die abstrakten Pläne des Knotens N_0. Bei der Auswahl eines Plans werden nach der Überprüfung der Anwendbarkeit der abstrakten Pläne im Knoten N_0, sequentiell die Pläne der Knoten $N_1, \ldots, N_r$ auf ihre Anwendbarkeit hin überprüft, um zum Problemlösen benutzt zu werden. Seien $\mathcal{F}_i = (F_1, \ldots, F_{ni})$ die Mengen von Planungsproblemen, aus denen die abstrakten Pläne der Knoten N_i für $i = 1, \ldots, r$, hervorgegangen sind. Für die Mengen von Planungsfällen gilt nun $F_i \subseteq F_0$, für $i = 1, \ldots, r$, da gilt $N_i \sqsubseteq_\mathcal{F} N_0$, gemäß der im vorherigen Kapitel beschriebenen Konstruktion der Abstraktionsbäume, da es sich bei den abstrakten Plänen in den Nachfolgerknoten immer um speziellere Pläne handelt. Um die Bedingung für die Beschneidung eines Nachfolgerknotens des Baumes noch einmal zu verdeutlichen,

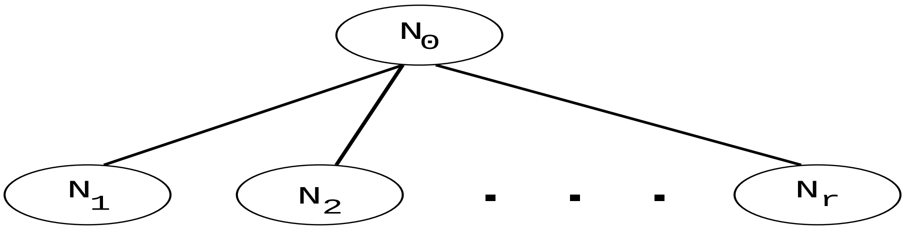

Abbildung 4.1: Die strukturelle Darstellung eines inneren Knotens des Abstraktionsbaums

sei nun folgender Fall angenommen. Um die Anwendbarkeit eines abstrakten Plans für ein Planungsproblem zu testen, muß der abstrakte Plan dazu re-instanziiert werden. Sei nun im Beispielbaum aus Abbildung 4.1 dies für den Knoten N_0 geschehen. So wäre es nur sinnvoll die Anwendbarkeit eines Plans aus dem Knoten N_1 zu überprüfen, falls die Summe der gemessenen Zeiten $Tm_{N_1} + Tr_{N_1}$ kleiner ist als Tr_{N_0}. Es ist also nur sinnvoll, einen weiteren Plan auf seine Anwendbarkeit hin zu überprüfen, falls die Summe der erwarteten Kosten für die Re-Instanziierung und die erwarteten Verfeinerungskosten mit diesem Plan kleiner sind als die Kosten zum Verfeinern mit dem Plan aus dem Knoten N_0, der bereits erfolgreich auf seine Anwendbarkeit hin überprüft wurde. Die Zeit Tm_{N_0} braucht nicht betrachtet zu werden, da sie ja schon verbraucht worden ist, um die Anwendbarkeit des abstrakten Plans im Knoten N_0 zu überprüfen. Dieser Sachverhalt gilt natürlich für jede Folge von Knoten, die bei dem Durchlaufen des Baumes zum Auffinden eines Planes zur Lösung eines neuen Planungsproblems hintereinander betrachtet werden.
Um nun eine Beschneidung des Baumes durchzuführen, ist neben den in der Vergangenheit gemessenen Zeiten zur Problemlösung die bedingte Wahrscheinlichkeit relevant, mit der ein Knoten der Abstraktionshierarchie zur Problemlösung beiträgt unter der Bedingung, daß ein Planungsproblem sich mit seinem Vorgängerknoten[2] lösen läßt. Da bekannt ist, aus welchen konkreten Planungsproblemen die abstrakten Pläne der Abstraktionshierarchie hervorgegangen sind, läßt sich für alle bisher bekannten Pläne diese bedingte Wahrscheinlichkeit berechnen. Für die Berechnung dieser bedingten Wahrscheinlichkeiten werden nun folgende Hilfsmengen $\widetilde{\mathcal{F}}$ konstruiert, die angeben, welche Planungsprobleme mit den einzelnen Knoten gelöst würden, falls man den unbeschnittenen Abstraktionsbaum zum Lösen eines bekannten Planungsproblems einsetzen würde.

Definition 4.6 *$\widetilde{\mathcal{F}}$ ergibt sich aus $\mathcal{F}$ nach folgender Vorschrift:*

- $\widetilde{\mathcal{F}}_0 := \mathcal{F}_0$
- $\widetilde{\mathcal{F}}_i := \left(\mathcal{F}_i \setminus \bigcup_{j=1}^{i-1} \widetilde{\mathcal{F}}_j\right)$, *für* $i \geq 1$

Aus diesen Mengen $\widetilde{\mathcal{F}}$ lassen sich nun die bedingten Wahrscheinlichkeiten berechnen, mit denen ein Planungsfall mit einem bestimmten Knoten des Baumes gelöst wird, unter der Voraussetzung, daß alle seine Vorgängerknoten das gegebene Planungsproblem lösten. Diese Wahrscheinlichkeiten beziehen sich natürlich nur auf die bisher betrachteten Planungsfälle.

[2]Vorgängerknoten sind hier die Knoten die beim Durchlaufen des Baumes vorher betrachtet wurden

Die Wahrscheinlichkeit P_i, daß ein Planungsproblem mit dem Knoten N_i gelöst wird, unter der Bedingung, daß es mit dem Knoten N_0 lösbar ist, ergibt sich aus:

$$P_i := P(N_i \wedge \neg N_{i-1} \wedge, \ldots, \neg N_1 \mid N_0) = \frac{|\widetilde{\mathcal{F}_i}|}{|\widetilde{\mathcal{F}_0}|}$$

Um die Berechnung dieser Wahrscheinlichkeiten noch einmal zu verdeutlichen, betrachten wir den Beispielbaum aus Abbildung 4.1, folgende hypothetische Planungsfallmengen $\mathcal{F}_i$ und die daraus resultierenden Hilfsmengen $\widetilde{\mathcal{F}_i}$ nach der obigen Defintion 4.6 und die daraus resultierenden Wahrscheinlichkeiten, die in der Tabelle 4.2 dargestellt sind: Diese so

Knoten	Planungsfälle $\mathcal{F}$	Hilfsmenge $\widetilde{\mathcal{F}}$	P_i
N_0	$(1,2,3,4,5,6,7,8)$	{1,2,3,4,5,6,7,8 }	1
N_1	$(4,5)$	{4,5 }	$\frac{1}{4}$
N_2	$(2,3,4,5)$	{2,3 }	$\frac{1}{4}$
N_3	$(2,4,6,7,8)$	{6,7,8 }	$\frac{3}{8}$

Tabelle 4.2: Die Knoten einer Abstraktionshierarchie mit ihren Planmengen und den daraus hervorgehenden Wahrscheinlichkeiten ihrer Benutzung bei der Problemlösung

berechneten Wahrscheinlichkeiten geben ein Maß für die Relevanz eines Knotens bei der Problemlösung im bezug auf die bisher betrachteten Planungsprobleme. Zusammen mit den Kosten eines Knotens bei der Problemlösung läßt nun die Wirkung eines Knotens auf den Erwartungswert der Problemlösezeit der Abstraktionshierarchie berechnen. Führt nun ein Knoten der Hierarchie zu einer Verschlechterung des Erwartungswerts, wird dieser Knoten und der sich darunter befindende Teilbaum der Abstraktionshierarchie abgeschnitten. Im folgenden Abschnitt wird nun der Erwartungswert der Problemlösezeit einer Abstraktionshierarchie definiert, an einem Beispiel erläutert und für den allgemeinen Fall definiert.

4.1.2 Die Schätzung des Erwartungswerts der Problemlösezeit einer Abstraktionshierarchie

Es sind nun alle Voraussetzungen erfüllt, um die Schätzung des Erwartungswert EW_K der Problemlösezeit eines Knotens K der Abstraktionshierarchie genau zu definieren, falls ein Planungsproblem mit einem Teilbaum der Abstraktionshierarchie gelöst werden kann. Allgemein ergibt sich der Erwartungswert aus dem Produkt der Zeit der Problemlösung mit einem Knoten und der Wahrscheinlichkeit, daß der Knoten zum Problemlösen benutzt wird.

Bevor nun eine allgemeine Definition der Schätzung des exakten Erwartungswerts für einen beliebigen Knoten der Abstraktionshierarchie gegeben wird, soll die Berechnung dieses Wertes noch einmal durch ein Beispiel verdeutlicht werden. Dazu betrachten wir noch einmal den Beispielbaum aus der Abbildung 4.1. Wir nehmen an, daß der Knoten N_0, dessen Erwartungswert wir berechnen wollen, über drei Nachfolgerknoten N_1, N_2, N_3 verfügt. Aus den dazugehörenden Planungsfälle $\mathcal{F}_{\prime}, \ldots, \mathcal{F}_{\infty}$ ergeben sich die Wahrscheinlichkeiten $P_1, \ldots, P_3$

Knoten	Planungsfälle $\mathcal{F}_i$	Hilfsmenge $\widetilde{\mathcal{F}_i}$	P_i	T_m	T_r
N_0	$(1,2,3,4,5,6,7,8)$	{1,2,3,4,5,6,7,8 }	1	150	160
N_1	$(4,5)$	{4,5 }	$\frac{1}{4}$	170	240
N_2	$(2,3,4,5)$	{2,3 }	$\frac{1}{4}$	180	200
N_3	$(2,4,6,7,8)$	{6,7,8 }	$\frac{3}{8}$	320	80

Tabelle 4.3: Die Knoten einer Abstraktionshierarchie mit ihren Planmengen und den daraus hervorgehenden Wahrscheinlichkeiten ihrer Benutzung und den entstehenden Kosten bei der Problemlösung

und die Re-Instanziierungs- und Verfeinerungszeiten T_m und T_r nach Definition 4.5, die mit der Lernkomponente gemessen worden sind, aus der Tabelle 4.3. Die Schätzung des Erwartungswerts für Knoten N_0 ergibt sich nun aus der Summe der gewichteten Kosten für die Problemlösung der einzelnen Knoten $N_0, \ldots, N_4$ des Beispielbaumes mit der Wahrscheinlichkeit seiner Anwendung bei der Problemlösung. Die einzelnen Summanden für die Knoten ergeben sich aus:

- Die Kosten des Knotens $\underline{N_1:}$
  - Re-Instanziierungskosten für den Knoten N_1: $\underline{170}$
  - Dem Produkt der Wahrscheinlichkeit seiner Anwendung und den damit verbundenen Kosten für das Verfeinern: $\frac{1}{4} * 240 = \underline{60}$
- Die Kosten des Knotens $\underline{N_2:}$
  - Dem Produkt der Wahrscheinlichkeit, daß der Knoten N_2 auf seine Anwendbarkeit hin überprüft wird, also der Knoten N_1 nicht anwendbar war und den Re-Instanziierungskosten für den Knoten N_2: $(1 - \frac{1}{4}) * 180 = \underline{135}$
  - Dem Produkt der Wahrscheinlichkeit der Anwendung des Knotens N_2 und den damit verbundenen Kosten für das Verfeinern: $\frac{1}{4} * 200 = \underline{50}$
- Die Kosten des Knotens $\underline{N_3:}$
  - Dem Produkt der Wahrscheinlichkeit, daß der Knoten N_3 auf seine Anwendbarkeit hin überprüft wird, also die Knoten N_1 und N_2 nicht anwendbar waren, und den Re-Instanziierungskosten für den Knoten N_3: $(1 - \frac{1}{4} - \frac{1}{4}) * 320 = \underline{160}$
  - Dem Produkt der Wahrscheinlichkeit der Anwendung des Knotens N_3 und den damit verbundenen Kosten für das Verfeinern des Knotens N_3: $\frac{3}{8} * 80 = \underline{30}$
- Die Kosten des Knotens $\underline{N_0:}$
  - Da der Knoten N_0 schon auf seine Anwendbarkeit hin überprüft wurde, um überhaupt den Beispielbaum zum Problemlösen benutzen zu können die Re-Instanziierungskosten für den Knoten N_0 bei der Berechnung vernachlässigt werden.

– Dem Produkt der Wahrscheinlichkeit der Anwendung des Knotens N_0, also N_1, N_2 und N_3 nicht anwendbar waren, und den damit verbundenen Kosten für das Verfeinern des Knotens N_0: $(1 - \frac{1}{4} - \frac{1}{4} - \frac{3}{8}) * 160 = \underline{20}$

Der Erwartungswert für die Poblemlösezeit für den Knoten N_0 ergibt sich nun aus der Summe der unterstrichenen Werte, also aus den gewichteten Kosten, die die einzelnen Knoten bei der Problemlösung verursachen:

$$EW_{N_0} := 170 + 60 + 135 + 50 + 160 + 30 + 20 = 625$$

Nun kann eine allgemeine Definition der Schätzung des exakten Erwartungswerts für einen beliebigen Knoten der Abstraktionshierarchie gegeben werden. Dazu wird zwischen Blattknoten und den inneren Knoten eines Baumes unterschieden. Die Kosten zum Verwalten des Baumes sind im Gegensatz dazu so gering, daß sie vernachlässigt werden können. Es ergibt sich so folgende *Schätzung an den Erwartungswert der Problemlösezeit* eines Knotens in der Abstraktionshierarchie:

Definition 4.7 *Seien $Tr, T_m, \mathcal{F}, \widetilde{\mathcal{F}}, P_i$ wie in diesem Abschnitt definiert. Sei N_m ein Knoten einer Abstraktionshierarchie. Dann ergibt sich die Schätzung des Erwartungswerts, EW_{N_m} für die Problemlösezeit des Knotens N_m aus:*

1. *Ist der Knoten N_m ein Blattknoten, dann hat er den Erwartungswert $EW_{N_m} :=$*

$$Tr_m(\widetilde{\mathcal{F}_m})$$

2. *Ein Knoten N_m ist kein Blattknoten und hat somit die allgemeine Gestalt aus Abbildung 4.1, dann hat er den Erwartungswert $EW_{N_m} :=$*

$$\sum_{i=1}^{r} P_r * EW_i + \sum_{i=1}^{r} \left[Tm_i(\widetilde{\mathcal{F}}) * \left(1 - \sum_{j=1}^{i-1} P_j\right)\right] + Tr_m\left(\widetilde{\mathcal{F}_m} \setminus \bigcup_{i=1}^{r} \widetilde{\mathcal{F}_i}\right) * \left(1 - \sum_{i=1}^{r} P_i\right)$$

Handelt es sich also um einen Blattknoten, so ergibt sich der Erwartungswert der Problemlösezeit aus der gemittelten Verfeinerungszeit für den minimalsten Plan des Knotens bezüglich $<_{GAP}$.

Der Erwartungswert der Problemlösezeit bei einem inneren Knoten berechnet sich dagegen etwas komplizierter. Hierzu sei noch einmal auf die Abbildung 4.1 verwiesen. Die Gewichte resultieren aus den bedingten Wahrscheinlichkeiten der bisher beobachteten Anwendungsmöglichkeiten der Pläne aus einem Knoten. Mit diesen Gewichten werden die bisher gemessenen Re-Instanziierungs- und Verfeinerungszeiten bewertet. Die einzelnen Summanden resultieren aus:

- Der erste Summand der Definition des Erwartungswerts repräsentiert die Summe aller Erwartungswerte der Nachfolgerknoten $N_1, \ldots, N_r$, gewichtet mit der Wahrscheinlichkeit ihrer Anwendungsmöglichkeit bei der Problemlösung.

- Im zweiten Summanden gehen die Zeiten ein, die durch das Re-Instanziieren von Plänen verbraucht werden, die für das aktuelle Planungsproblem nicht anwendbar sind und in Nachfolgerknoten $N_1, \ldots, N_r$ des betrachteten Knotens N_0 enthalten sind. Diese Zeiten werden auch wieder durch die Wahrscheinlichkeiten, daß diese Pläne re-instanziiert werden, aber nicht anwendbar sind, gewichtet.

- Der dritte Summand der Definition repräsentiert die gewichteten Kosten, daß alle Nachfolgerknoten nicht zur Problemlösung benutzt werden können, aber betrachtet werden und ein Plan aus dem Knoten N_0 benutzt wird.

Eine ausführliche Herleitung des Erwartungswerts der Problemlösezeit für einen inneren Knoten ist für den interessierten Leser im Anhang dieser Arbeit wiedergegeben. Für das Verständnis der weiteren Arbeit sollten die obigen Informationen jedoch genügen.
Ziel des Algorithmus zur Minimierung des Erwartungswerts der Problemlösezeit einer Abstraktionshierarchie ist es nun, den Erwartungswert des Wurzelknotens einer Hierarchie zu minimieren. Dieser Baum mit der minimalen Problemlösezeit für alle bisher bekannten Planungsprobleme wird dann zum Lösen neuer Planungsprobleme verwandt. Er garantiert im bezug auf neue Planungsprobleme zwar kein optimales Problemlöseverhalten, kann aber als die aus dem vorliegenden Wissen über das Verhalten der abstrakten Pläne bei der Problemlösung , als beste Annäherung an eine solche Lösung verstanden werden, wenn die bisherigen Fälle repräsentativ für die in der Zukunft vorkommenden Probleme sind. Im folgenden Abschnitt wird der Algorithmus vorgestellt, der die Abstraktionshierarchie durch Beschneiden und Umsortieren von Nachfolger-Teilbäumen so abändert, daß der Erwartungswert der Problemlösezeit im Wurzelknoten minimal wird.

4.2 Die Funktionsweise des Algorithmus und seine Realisierung

In diesem Abschnitt wird nun der Algorithmus zur Minimierung des Erwartungswerts der Problemlösezeit einer Abstraktionshierarchie erläutert. Im ersten Teil wird die Idee des Verfahrens und seine Realisierung beschrieben. Im zweiten Teil wird der Algorithmus in Pseudocode vorgestellt. Zum Schluß wird noch einmal ein Beispiel gegeben, daß die Funktionsweise des Verfahrens im PARIS-Planungssystem verdeutlichen soll.

4.2.1 Die Realisierung des Algorithmus zum Minimieren des Erwartungswerts der Problemlösezeit

Das Ziel dieses Verfahrens ist es, den Erwartungswert der Problemlösezeit nach Definition 4.7 des Wurzelknoten des Baumes zu minimieren. Dazu wird eine inkrementell gelernte Abstraktionshierarchie, wie sie im vorherigen Kapitel beschrieben wurde, vorausgesetzt. Nach jeder Abstraktion eines neuen konkreten Plannugsfalles wird diese Abstraktionshierarchie, gemäß dem vorgestellten Algorithmus (Kapitel 4) zum inkrementellen Lernen von Abstraktionshierarchien verändert. Um die nötigen Daten zur Minimierung des Erwartungswerts der neuen Abstraktionshierarchie zu erlangen, wird mit allen neu gewonnenen abstrakten

Plänen das ursprüngliche konkrete Planungsproblem gelöst, aus dem die neuen Pläne hervorgegangen sind. Dabei werden die Zeiten, die zum Re-Instanziieren und zum Verfeinern benötigt werden, gemessen. Nun muß der Algorithmus unter Einbeziehung dieser Daten die neu entstandene Abstraktionshierarchie so optimieren, daß der Erwartungswert der Problemlösezeit minimal wird. Dazu ist es nötig, die

- Anordnung der abstrakten Pläne in einem Knoten zu optimieren
- Anordnung der Nachfolgerknoten eines Knotens zu optimieren
- Abstraktionshierarchie so zu beschneiden, das immer der abstrakte Plan gewählt wird, sodaß die Gesamtproblemlösezeit minimal wird.

Um die Anordnung der Pläne in einem Knoten zu optimieren, müssen in jedem Knoten der Abstraktionshierarchie, die dort enthaltenen Pläne vom minimalen Plan gemäß der Definition 4.4 der $<_{GAP}$-Ordnung aufsteigend angeordnet werden. Dazu werden die Werte T_{m_i} und T_{r_i}, der sich bereits in der Abstraktionshierarchie befindenden Pläne nach Definition 4.3, neu berechnet, da sich ja durch die neuen Messungen die mittlere Re-Instanziierungszeit und die mittlere Verfeinerungszeit dieser Pläne verändert hat. Nachdem diese neuen Werte bekannt sind, werden in den Knoten, die diese Pläne enthalten, die Pläne gemäß der $<_{GAP}$-Ordnung neu angeordnet. So wird eine lokale Optimierung der Anordnung der abstrakten Pläne in einem Knoten erzielt. Alle abstrakten Pläne, die neu akquiriert worden sind, werden in neue Knoten der Abstraktionshierarchie nach der $<_{GAP}$-Ordnung eingeordnet, in die sie nach dem Algorithmus zum inkrementellen Lernen von Abstraktionshierarchien gehören. Somit ist die Anordnung der abstrakten Pläne in aufsteigender Reihenfolge nach der $<_{GAP}$-Ordnung in allen Knoten der Abstraktionshierarchie wieder hergestellt.
Neben der Anordnung der Pläne innerhalb eines Knotens, hat die Anordnung der Nachfolgerknoten eines Baumknotens Einfluß auf die Reihenfolge in der die abstrakten Pläne zum Problemlösen benutzt werden und somit einen Einfluß auf den Erwartungswert der Hierarchie. Es soll also die Anordnung der Nachfolgerknoten eines Knotens lokal optimiert werden, sodaß der Erwartungswert im Wurzelknoten mimimal wird. Wir nehmen an, daß die Planmengen $\mathcal{F}_i$, für $i = 1, \ldots, r$, disjunkt sind, also gilt: $\forall_{i,j \geq 1}$ ist $\mathcal{F}_i \cap \mathcal{F}_j = \emptyset$. Diese Annahme ist notwendig, um die Zeitkomplexität der Beschneidung der Abstraktionshierarchie so zu reduzieren, daß sie noch bei realistischen Anwendungen durchführbar ist. Auf die Notwendigkeit dieser Annahme wird weiter unten genauer eingegangen. Sie kann jedoch auch hier benutzt werden, um die Definition der Anordnung der Nachfolgerknoten eines jeden Knotens zu vereinfachen, sodaß der Erwartungswert des Knotens minimal wird. Um die im folgenden angestellten Überlegungen noch einmal zu verdeutlichen, betrachten wir den Beispielbaum aus Abbildung 4.1. Eine Ordnung der Nachfolgerknoten $N_1, \ldots, N_r$ eines Knotens der Hierarchie N_0 ist gegeben durch:

Definition 4.8 *Sei N_0 ein innerer Knoten der Abstraktionshierarchie und $N_1, \ldots, N_r$ die Menge seiner Nachfolgerknoten mit den Re-Instanziierungszeiten Tm_{N_i}, für $i =, 1, \ldots, r$, nach Definition 4.5, dann gilt:*

$$N_i <_{Node} N_j \ \underline{gdw.} \ Tm_{N_i} < Tm_{N_j}$$

Die Sortierung der Nachfolgerknoten nach aufsteigenden Re-Instanziierungszeiten rechtfertigt sich, wenn man die Definition 4.7 im Zusammenhang mit der obigen Annahme $\forall_{i,j\geq 1}$ $\mathcal{F}_i \cap \mathcal{F}_j = \emptyset$ betrachtet.

$$\underbrace{\sum_{i=1}^{r} P_r * EW_i}_{1.Summand} + \underbrace{\sum_{i=1}^{r}\left[Tm_i(\widetilde{\mathcal{F}}) * \left(1 - \sum_{j=1}^{i-1} P_j\right)\right]}_{2.Summand} + \underbrace{Tr_m\left(\widetilde{\mathcal{F}_m} \setminus \bigcup_{i=1}^{r} \widetilde{\mathcal{F}_i}\right) * \left(1 - \sum_{i=1}^{r} P_i\right)}_{3.Summand}$$

Im ersten Summand ist der Erwartungswert EW_i der Nachfolgerknoten selbst unabhängig von der Reihenfolge der Nachfolgerknoten. Da die konkreten Planmengen $\mathcal{F}_i$ nach der Annahme disjunkt sind, gilt: $\forall_{i,j\geq 1}\mathcal{F}_i = \widetilde{\mathcal{F}_i}$ unter Beachtung der Definition 4.6. Daraus folgt direkt, daß die Wahrscheinlichkeiten P_i für die Anwendbarkeit eines Knotens unabhängig von der Reihenfolge der Nachfolgerknoten ist. Somit ist der erste Summand unabhängig von der Reihenfolge der Nachfolgerknoten und braucht nicht mehr betrachtet zu werden. Die Reihenfolge der Nachfolgerknoten hat auch keinen Einfluß auf den dritten Summanden. Wie zuvor gezeigt, sind die Wahrscheinlichkeiten P_i unabhängig von der Reihenfolge der Nachfolgerknoten und da gilt: $\forall_{i,j\geq 1}\mathcal{F}_i = \widetilde{\mathcal{F}_i}$, ist auch $Tr_m\left(\widetilde{\mathcal{F}_m} \setminus \bigcup_{i=1}^{r} \widetilde{\mathcal{F}_i}\right)$ unabhängig. So hat die Anordnung der Nachfolgerknoten keinen Einfluß auf den dritten Summanden.
Der zweite Summand ist als einziger von der Reihenfolge der Nachfolgerknoten abhängig. Dort werden die Re-Instanziierungszeiten $Tm_i(\widetilde{\mathcal{F}})$ mit dem Faktor $\left(1 - \sum_{j=1}^{i-1} P_j\right) := \alpha_i$ multipliziert. Dieser Faktor fällt monoton, da gilt $\alpha_i \geq \alpha_{i+1}$, für $i = 1, \ldots, r$. Somit wird der zweite Summand minimal, falls die kleinsten Re-Instanziierungskosten $Tm_i(\widetilde{\mathcal{F}})$ zuerst in der Summenformel mit den grösseren α_i multipliziert werden. So genügt es bei der Anordnung der Nachfolgerknoten eines Knotens, sie nach aufsteigenden Re-Instanziierungskosten zu sortieren, damit der Erwartungswert des Vorgängerknotens N_0 minimal wird. Um also den Erwartungswert der inneren Knoten der Abstraktionshierarchie lokal zu optimieren, werden ihre Nachfolgerknoten nach der $<_{Node}$-Ordnung sortiert. Dazu werden die Werte Tm_K nach Definition 4.5 für jeden Knoten K berechnet und die Knoten so umsortiert, daß Knoten, die kleiner sind, bezüglich der $<_{Node}$-Ordnung vor größeren Knoten zum Problemlösen benutzt werden.
Im Gegensatz zu den beiden vorherigen Optimierungen gestaltet sich die Beschneidung der Abstraktionshierarchie etwas schwieriger. Um den Erwartungswert der Problemlösezeit des Wurzelknotens zu berechnen, werden für den Abstraktionsbaum Top-Down die Mengen von Planungsproblemen $\widetilde{\mathcal{F}}$ nach Definition 4.6 konstruiert. Aus diesen Mengen kann man dann die bedingten Wahrscheinlichkeiten P_i für die Knoten des Baum berechnen. Nun können die Werte für $Tr_m\left(\widetilde{\mathcal{F}} \setminus \bigcup_{i=1}^{r} \widetilde{\mathcal{F}_i}\right)$ berechnet werden. Zum Schluß muß man dann Bottom-Up die Erwartungswerte EW für die Knoten des Baumes nach Definition 4.7 berechnen, bis man den Erwartungswert der Problemlösezeit für den Wurzelknoten erhält. Dieser Wert stellt nun das Optimierungsmaß dar und ist zu minimieren.
Um den Erwartungswert der Problemlösezeit für den Wurzelknoten zu verändern, muß man nun an allen inneren Knoten die Nachfolger des Knotens so ein- und ausblenden, daß der Erwartungswert der Wurzel minimal wird. Als Problem erweist sich jedoch hier die Größe des Suchraums, um dieses Minimum zu finden. Bei einem Baum mit Verzweigungsgrad g und k inneren Knoten, muß der oben beschriebene Algorithmus $(g!)^k$ mal ausgeführt werden. Dies ist natürlich für reale Anwendungen mit großen Abstraktionshierarchien nicht

durchführbar.
Um diese Komplexität zu reduzieren, kann man nun die bereits oben benutzte Annahme treffen, daß die konkreten Planungsfälle zu den abstrakten Plänen aller Geschwisterknoten in dem Baum disjunkt sind, also gilt: $\forall_{i,j\geq 1}\mathcal{F}_i = \widetilde{\mathcal{F}_i}$. Sind nun die Planungsfälle aller Teilzweige des Baumes disjunkt, so ist die Optimierung der Teilbäume, die als Nachfolger eines inneren Knoten auftreten, voneinander unabhängig. Somit kann jeder Teilzweig des Abstraktionsbaumes lokal optimiert werden. Führt man diese Optimierung nun Bottom-Up durch, muß man für jeden inneren Knoten N nur noch das Element der Potenzmenge der Nachfolger von N finden, so daß der Erwartungswert der Problemlösezeit für den Knoten N minimal wird. Die Potenzmenge der Nachfolgerknoten repräsentiert alle möglichen Beschneidungen des Knotens N. Durch die obige Annahme wird dann garantiert, daß dadurch auch der Erwartungswert der Problemlösezeit der Wurzel minimal wird. Somit muß für k innere Knoten bei einem durchschnittlichen Verzweigungsfaktor g ein leicht modifizierter Algorithmus $k * 2^g$-mal durchgeführt werden. Bei einer Verletzung der Annahme kann nicht mehr garantiert werden, durch die lokale Optimierung die beste Annährung an die Schätzung des Erwartungswerts zu finden. Es ergibt sich so ein indirekter „trade-off" zwischen der Komplexität bei der Beschneidung des Baumes und der Güte der Annährung an die Schätzung des minimalen Erwartungswerts. In der gewählten Implementierung wurde mit minimalem Aufwand, also vollständige lokale Optimierung, die Abstraktionshierarchie beschnitten, da in der verwendeten Domäne[3] die getroffene Annahme galt. Es dürfte kein Problem darstellen, in Domänen, in denen diese Annahme nicht gilt, eine dynamische Optimierung durchzuführen, indem, falls zwei Teilzweige der Abstraktionshierarchie abhängig sind, diese gemeinsam optimiert werden und umgekehrt, bei unabhängigen Planmengen diese lokal optimiert werden. So wäre es möglich mit minimalem Aufwand eine möglichst gute Annährung an die Schätzung des Erwartungswerts zu bekommen.
Da die Abstraktionshierarchie in der gewählten Implementierung vollständig lokal optimiert wird, werden die Werte für $\widetilde{\mathcal{F}}$, P_i und $Tr_m\left(\widetilde{\mathcal{F}_m} \setminus \bigcup_{i=1}^{r} \widetilde{\mathcal{F}_i}\right)$ nur noch einmal global Top-Down für den Baum berechnet und später gemäß den aus- und eingeblendeten Nachfolgerknoten lokal korrigiert. Dann werden Bottom-Up für alle inneren Knoten die minimalen Erwartungswerte berechnet und gespeichert. Liegt die Menge der Nachfolgerknoten für einen inneren Knoten fest, wird diese nicht mehr verändert. So erhält man eine lokale Optimierung des Erwartungswerts für die Problemlösezeit für den Wurzelknoten. Die Anordnung der Pläne in den Knoten und die Anordnung der Nachfolgerknoten eines Knotens bleibt von dem Ein- und Ausblenden der einzelnen Teilbäume unverändert. Der so entstandene Baum wird dann in dieser Form für die Problemlösung eingesetzt. Im folgenden Abschnitt wird der Algorithmus noch einmal in Pseudocode dargestellt.

4.2.2 Die Beschreibung des Algorithmus in Pseudocode

Der Algorithmus muß als erstes, nachdem ein neues Planungsproblem abstrahiert wurde, die korrekte Abstraktionshierarchie herstellen. Dies wird durch die Ausführung der in dem vorherigen Kapitel vorgestellten Prozedur `LearnHierarchy(`K_0`,`B_{neu}`)` erreicht. Die Prozedur `Sort_Plans_in_Nodes(`K_0`)` realisiert die $<_{GAP}$-Ordnung in allen Knoten der Abstraktionshierarchie. Als nächstes werden mit dem Aufruf `Partitions(`K_0`)` die Mengen $\widetilde{\mathcal{F}}$

[3] Die untersuchte Planungsdomäne wird im folgenden Kapitel vorgestellt.

und die Wahrscheinlichkeiten P_i berechnet. Die Funktion `Get_all_leaves(`K_0`,`$Leaves$`)` liefert alle Blätter aus dem Baum mit der Wurzel K_0. Zum Schluß wird mit der Prozedur `Local_Optimization(`$Leaves$`)` der Erwartungswert der Problemlösezeit der Teilbäume Bottom-Up minimiert. Diese Funktionen werden durch die folgenden Prozedur `Calculate_Hierarchy(`K_0, B_{neu}`)` realisiert, die als Parameter die Wurzel des Baumes K_0 und die neue Beispielmenge B_{neu} übergeben bekommt.

PROZEDUR Calculate_Hierarchy(K_0, B_{neu})
BEGIN
  Learn_Hierarchy(K_0, B_{neu})
  Sort_Plans_in_Nodes(K_0)
  Partitions($\{K_0\}$)
  Get_all_leaves(K_0,$Leaves$)
  Local_Optimization($Leaves$)
END

Die Prozedur `Partitions(`$NodeSet$`)` bekommt eine Menge von Knoten des Baumes übergeben. Initial ist das die Menge mit dem Wurzelknoten K_0. Die abstrakten Pläne der Knoten aus dem Parameter $NodeSet$ und die abstrakten Pläne der Nachfolgerknoten dieser Knoten werden dann sortiert. Außerdem wird die Reihenfolge der Nachfolger der Knoten aus $NodeSet$ so sortiert, daß die Knoten mit den abstrakten Plänen, die kürzere Re-Instanziierungszeiten besitzen, zuerst überprüft werden, d.h daß sie weiter links stehen als Pläne mit längeren Re-Instanziierungszeiten. Dies wird durch die Prozedur `SortGAPs(`$NodeSet$`)` realisiert. Dann werden für alle Knoten aus $NodeSet$ und deren Nachfolgerknoten, die Mengen $\widetilde{\mathcal{F}}$ und die Werte für P_i berechnet. Dies geschieht mit der Prozedur `Divide(`$Node$`)`, die im Folgenden noch erläutert wird. Zum Schluß wird die Prozedur `Partitions(`$NodeSet$`)` rekursiv mit der nächsten Ebene des Baumes aufgerufen.

PROZEDUR Partitions($NodeSet$)
BEGIN
  SortGAPs($NodeSet$)
  FOR ALL $K_i \in NodeSet$ **DO**
    BEGIN
    Divide(K_i)
    FOR ALL $K_j = \{K_1, \ldots, K_m\}$ Nachfolger von K_i **DO**
      Partition($\{K_1, \ldots, K_m\}$)
  END
END

Die Prozedur `Divide(Node)` bekommt als Parameter $Node$ den aktuell zu bearbeitenden Knoten. Mit der Funktion `GetSuccsessors(`K_0`,`Suc`)` werden alle Nachfolgerknoten von $Node$ aus dem Baum aufgesammelt. Die Funktion `GetF(`K_0`,`$\mathcal{F}_0$`)` liefert alle konkreten Planungsfälle, die zu den abstrakten Plänen im Knoten K_0 gehören. Nun werden für alle

Nachfolgerknoten S_i[4] die konkreten Planungsfälle mit der Funktion `GetF(`S_i`,`$\mathcal{F}_i$`)` aus dem Speicher geholt. Dann werden die Werte für $\widetilde{\mathcal{F}_i}$ und P_i berechnet und die Variable *Occured* wird aktualisiert. Ist dies für alle Nachfolger des Knotens durchgeführt, wird die Prozedur `Divide(Node)` rekursiv mit den Nachfolgern S_i aufgerufen.

PROZEDUR Divide(K_0)
BEGIN
  GetSuccsessors(K_0,Suc)
  GetF(K_0,$\mathcal{F}_0$)
  $Occured := \emptyset$
  FOR ALL $Suc = (S_1, \ldots, S_m,)$ **DO**
  BEGIN
    GetF(S_i,$\mathcal{F}_i$)
    $\widetilde{\mathcal{F}_i} := \mathcal{F}_0 \setminus Occured$
    $Occured := Occured \cup \mathcal{F}_i$
    $P_i := \frac{Card(\mathcal{F}_0)}{Card(\widetilde{\mathcal{F}_i})}$
  END
  FOR ALL $Suc = (S_1, \ldots, S_m,)$ **DO**
    Divide(S_i)
END

Nachdem der Baum mit der obigen Prozedur Top-Down durchlaufen wurde, wurden die Werte für $\widetilde{\mathcal{F}_i}$ und P_i für alle Knoten berechnet. Nun wird der Baum Bottom-Up abgearbeitet und lokal optimiert, d.h. das der Erwartungswert der Problemlösezeit gemäß Definition 4.7 in jedem Knoten minimiert wird. Dies wird durch die Prozedur `Local_Optimization(`$Leaves$`)` realisiert, die eine Menge von Knoten $Nodes$ übergeben bekommt. Falls es sich um den ersten Aufruf der Prozedur handelt, bei dem die Blätter des Baumes übergeben werden, ergibt sich der Erwartungswert aus $Tr_j(\widetilde{\mathcal{F}_m})$, der durch `Calculate_EW_Leaves(`$Nodes$`)` berechnet wird. Sonst werden mit `GetSuccsessors(`L_i`,`Suc`)` die Nachfolger des inneren Knotens ermittelt und mit `Calculate_Pot(`Suc`,`$PotSuc$`)` deren Potenzmenge berechnet. Die Potenzmenge der Nachfolgerknoten $PotSuc$ entspricht allen möglichen Beschneidungen des Abstraktionsbaumes. In der Schleife werden dann für alle Elemente dieser Potenzmenge die Erwartungswerte mit `Calculate_EW(`PS_i`,`Tmp`)` berechnet. Nachdem das Element der Potenzmenge $OptPot$ gefunden wird, das den minimalen Erwartungswert gemäß Definition 4.7 für den inneren Knoten L_i liefert, werden die Nachfolger des zu bearbeitenden Knotens L_i auf diesen Wert gesetzt. So werden die Zweige des Baumes ausgeblendet, die die Problemlösezeit unnötig verlängern und die Zweige, die zu einer Verkürzung der Problemlösezeit beitragen, bleiben erhalten. Dies wird nun mit allen Elementen L_i, $i = \{1, \ldots, m\}$, aus $Nodes$ durchgeführt. Danach werden mit `Get_All_Pred(`$Nodes$`,`$Pred$`)` alle Vorgängerknoten ermittelt, um die nächste Ebene des Baumes abzuarbeiten. Der Algorithmus bricht ab, falls die Menge der Vorgänger leer ist, also der Wurzelknoten erreicht wurde.

[4] Die Menge der Nachfolger $(S_1, \ldots, S_m,)$ stellt ihre Anordnung im Baum dar und muß auch in dieser Reihenfolge abgearbeitet werden

```
PROZEDUR Local_Optimization(Nodes)
BEGIN
IF Nodes ≠ ∅ THEN
BEGIN
IF Nodes are the Leaves of the Tree THEN
   Calculate_EW_Leaves(Nodes)
   ELSE
   BEGIN
      FOR ALL Nodes = {L_1, ..., L_m,} DO
      BEGIN
         GetSuccsessors(L_i,Suc)
         Calculate_Pot(Suc,PotSuc)
         Min := MAXINT
         OptPot := ∅
         FOR ALL Ps_i ∈ PotSuc DO
         BEGIN
            Calculate_EW(PS_i,Tmp)
            IF Tmp < MIN THEN
            BEGIN
               Min := Tmp
               OptPot := PS_i
            END
         Set_PSuc(L_i,OptPot)
         END
      END
      Get_All_Pred(Nodes,Pred)
      Local_Optimization(Pred)
   END
END
END
```

Da es sich hier nicht um einen ausgeglichenen Baum handelt, können verschiedene Teilbäume ausgehend von der Wurzel verschiedene Tiefen haben. Bei der obigen Darstellung wurde davon abstrahiert, um die Darstellung des Algorithmus nicht unnötig kompliziert zu machen. In der realen Implementierung werden die Knoten nicht nach ihrer Ebene, sondern in der Reihenfolge ihrer Pfadlänge, ausgehend von der Wurzel des Baumes, bearbeitet. So ist garantiert, daß bei dem Aufruf `Calculate_EW(`PS_i`,`Tmp`)` alle benötigten Werte vorhanden sind. Außerdem ist zu beachten, daß die Werte für die Mengen $\widetilde{\mathcal{F}}_i$ und die Wahrscheinlichkeiten P_i, je nach dem gewählten Element der Potenzmenge, in der Prozedur `Calculate_EW(`PS_i`,`Tmp`)` lokal korrigiert werden müssen. Im folgenden letzten Abschnitt dieses Kapitels ist ein Beispiel zu dem Algorithmus beschrieben, das seine Funktionsweise noch einmal verdeutlichen soll.

4.3 Die Verdeutlichung des Algorithmus an einem Beispiel

In diesem Abschnitts wird der soeben beschriebene Algorithmus noch einmal an einem Beispiel erläutert. Dazu betrachten wir vier konkrete Planungsfälle, F_1, F_2, F_3, F_4, die auch in dieser Reihenfolge mit dem Planungssystem PARIS abstrahiert werden. Nachdem ein Planungsfall abstrahiert wurde, wird er mit den aus ihm resultierenden abstrakten Plänen wieder gelöst. Die so gemessenen Re-Instanziierungs- und Verfeinerungszeiten, sowie die abstrakten Pläne, die aus den konkreten Planungsfällen hervorgingen, sind der Tabelle 4.4 zu entnehmen.

Planungsproblem	GAP	T_m	T_r
F_1	1	3960	2290
F_2	2	4520	2570
	3	4730	1998
F_3	1	3880	2320
	4	12400	8570
F_4	2	4100	2350
	3	4140	2080
	5	520	570

Tabelle 4.4: Eine Menge von konkreten Planungsfällen mit ihren abstrakten Plänen und ihren Re-Instanziierungs- und Verfeinerungszeiten

Die Werte für T_m, T_r sind in CPU-Millisekunden angegeben. Wir betrachten nun die Abstraktionsbäume, die nach dem Abstrahieren der einzelnen konkreten Planungsfälle entstehen. Dabei wird gemäß des Algorithmus zur Minimierung des Erwartungswerts zuerst der vollständige Abstraktionsbaum nach dem Clusterungsalgorithmus erzeugt. Dann wird dieser Baum so beschnitten, daß im bezug auf die bisher betrachteten konkreten Planungsfälle die Problemlösezeit minimal wird.
Zuerst wird der Planungsfall $F_1 = (si_1, sg_1, P_1)$ abstrahiert. Daraus resultiert der abstrakte Plan mit der Nummer 1. Nachdem der abstrakte Plan 1 auf das konkrete Planungsproblem $P_1 = (si_1, sg_1)$ re-instanziiert und verfeinert wurde, ergeben sich die Zeiten aus der ersten Zeile der Tabelle. Es entsteht der Abstraktionsbaum, der in der Abbildung 4.2 dargestellt ist. Dieser Baum besteht nur aus einem Knoten, der den abstrakten Plan 1 und den immer gültigen Plan 0 enthält. Die Re-Instanziierungs- und die Verfeinerungszeit dieses Plans werden auf $MAXINTEGER$ gesetzt, da dieser Plan nicht existiert und auch kein konkretes Planungsproblem mit ihm gelöst werden kann. Wird durch die Planverfeinerungskomponente dieser Plan ausgewählt, so ist mit den Plänen des Abstraktionsbaums das konkrete Planungsproblem, das gelöst werden sollte, nicht zu lösen.
In diesem Knoten ist die Re-Instanziierungszeitzeit T_m des Knotens angegeben, der nach der Ordnung über abstrakten Plänen, gemäß Definition 4.4 der kleinste ist. Da es sich hier um einen Blattknoten handelt, ergibt sich der Erwartungswert des Knotens T_n[5] aus

[5]Innerhalb des Programms wird der Erwartungswert mit T_n für $Time_{node}$ auf englisch bezeichnet

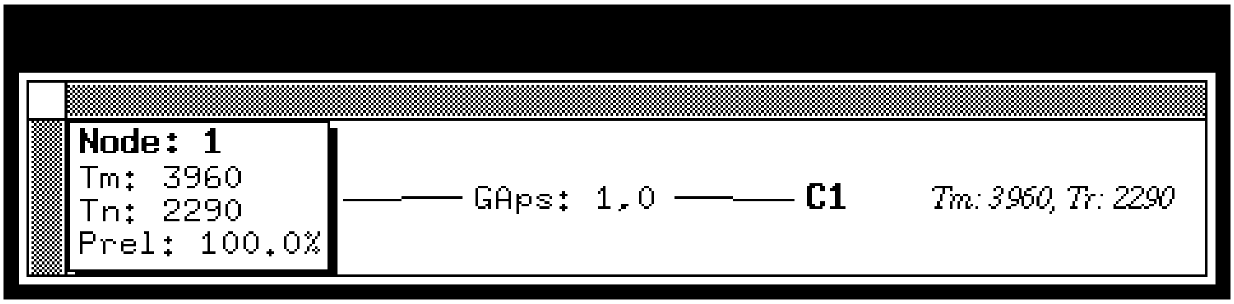

Abbildung 4.2: Der Abstraktionsbaum nach dem Lernen von Planungsfall 1

$Tr_j(\widetilde{\mathcal{F}_m})$ nach der Definition 4.7. Darunter befindet sich die bedingte Wahrscheinlichkeit P_i, daß ein bisher betrachteter Planungsfall sich mit diesem Knoten lösen läßt, hier 100 Prozent. Dahinter befinden sich die abstrakten Pläne $1, 0$ und die Werte für T_m und T_r für den Planungsfall 1.
Als nächstes wird der konkrete Planungsfall 2 abstrahiert. Der daraus resultierende Abstraktionsbaum ist in Abbildung 4.3 dargestellt. Aus diesem Planungsfall ergeben sich die

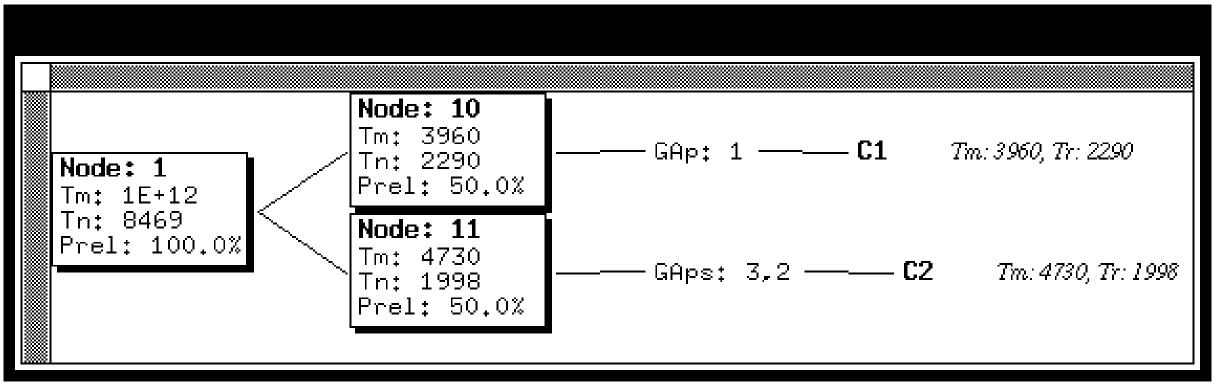

Abbildung 4.3: Der Abstraktionsbaum nach dem Lernen von Planungsfall 1 und 2

abstrakten Pläne 2 und 3. Die daraus resultierenden Zeiten zum Re-Instanziieren und Verfeinern des Plans sind der Tabelle 4.4 zu entnehmen. Der so entstehende Abstraktionsbaum besteht aus drei Knoten. Im Wurzelknoten befindet sich nur noch der abstrakte Plan 0. Die Matchzeit ist aus dem oben geschilderten Grund auf $MAXINTEGER$ gesetzt. Darunter befindet sich der Erwartungswert der Problemlösezeit T_n für den ganzen Abstraktionsbaum, der sich nach der Formel im zweiten Teil der Definition 4.7 errechnet. Da die abstrakten Planmengen vom Planungsfall 1 und 2 disjunkt sind, ergeben sich nach dem Algorithmus zum inkrementellen Lernen von Abstraktionshierarchien zwei Nachfolgerknoten, die die jeweiligen Planmengen enthalten. Die Werte der Knoten für T_m ergeben sich wieder aus dem minimalen Plan des Knotens. Der Wert für den Erwartungswert T_n der Problemlösezeit ergibt sich wieder aus den Verfeinerungskosten des minimalsten abstrakten Plans bezüglich $<_{GAP}$. Die bedingten Wahrscheinlichkeiten mit jeweils 50 Prozent ergeben sich daraus, daß mit jedem der beiden Knoten genau ein konkretes Planungsproblem gelöst werden kann

und insgesamt zwei Planungsprobleme bekannt sind. Die beiden abstrakten Pläne 2 und 3 im Knoten 11 sind wieder nach der $<_{GAP}$ Ordnung sortiert. In dem Baum bleiben die Knoten 10 und 11 erhalten, weil sich der Erwartungswert der Problemlösezeit bei allen anderen Kombinationen von ein- und ausgeblendeten Knoten, $(\{\}, \{10\}, \{11\})$, erheblich erhöht. Dies liegt daran, daß bei diesen Kombinationen die hohen Werte des abstrakten Plans 0 mit in die Berechnung eingehen, der ja dann auch bei 50 Prozent der konkreten Planungsfälle, bzw. immer zum Lösen benutzt werden müßte. Somit ist es klar, daß kein Knoten aus dem Abstraktionsbaum abgeschnitten wird, da jeder Knoten mit seinen abstrakten Plänen zu einer Verbesserung der Problemlösezeit für die bisher betrachteten konkreten Planungsfälle beiträgt.
Nun wird der konkrete Planungsfall 3 nach der PABS-Methode abstrahiert. Aus diesem Planungsfall ergeben sich die abstrakten Pläne 1 und 4. Der neu entstehende Abstraktionsbaum ist in Abbildung 4.4 dargestellt. Der Wurzelknoten hat, bis auf den neuen Erwartungswert

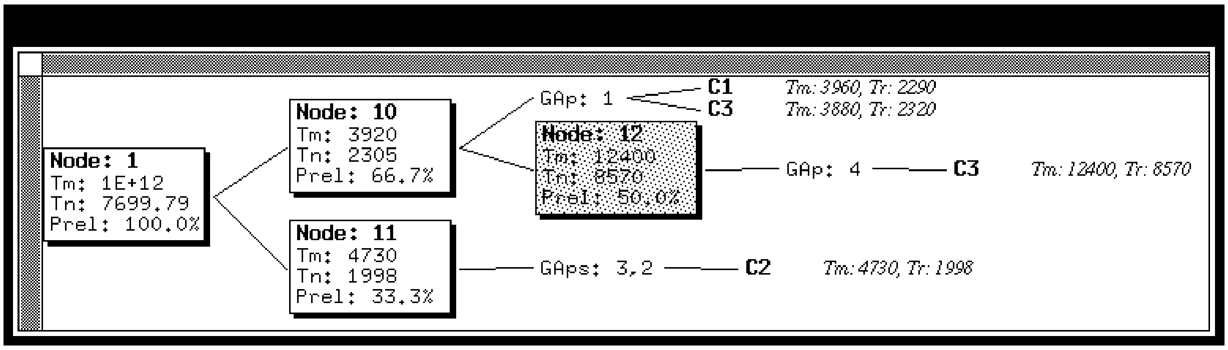

Abbildung 4.4: Der Abstraktionsbaum nach dem Lernen von Planungsfall 1, 2 und 3

des Baumes, wieder die gleiche Gestalt. Im Knoten 11 hat sich bis auf den neuen Wert für die Wahrscheinlichkeit auch nichts geändert. Der interessante Knoten ist jetzt der Knoten 10. Da durch den neuen konkreten Planungsfall die abstrakten Pläne 1 und 4 erzeugt wurden, entsteht nach dem Algorithmus zur Bildung der Abstraktionshierarchie der hier abgebildete Baum, der einen zusätzlichen Knoten 12 enthält. Dieser Knoten ist der Nachfolgerknoten vom Knoten 10 und er enthält den abstrakten Plan 4. Dieser Knoten ist jedoch von dem Algorithmus zum Minimieren der Problemlösezeit ausgeblendet worden. Dies ist durch seine graue Schattierung angedeutet. Dies liegt an den hohen Re-Instanziierungs- und Verfeinerungskosten von GAP_4 für den konkreten Planungsfall F_3, die aus der 5. Zeile der Tabelle 4.4 zu entnehmen sind.
Dies ist auch sofort einzusehen, wenn man sich die Zeiten zum Problemlösen des konkreten Planungsproblems 3 mit dem unbeschnittenen und dem beschnittenen Abstraktionsbaum betrachtet. Bei dem vollständigen Abstraktionsbaum, der nach dem Algorithmus zum Erlernen der Abstraktionshierarchie entsteht, würde der abstrakte Plan 4 zum Lösen des Planungsproblems 4 genommen, da ja hier immer der speziellste Plan zum Problemlösen genommen wird. Somit ergibt sich eine Problemlösezeit von 24850 $msec$ als Addition aus der Zeit zum Re-Instanziieren des Planungsproblems 3 mit dem abstrakten Plan 1 und der beiden Werte für das Re-Instantiieren und Verfeinern des Planungsproblems 3 mit dem abstrakten Plan 4. Wird jedoch der beschnittene Baum zum Problemlösen benutzt, so wird das Planungsproblem mit dem abstrakten Plan 1 gelöst. Somit ergibt sich die Gesamtpro-

blemlösezeit aus der Summe der Re-Instantiierungs- und Verfeinerungszeit für das Lösen des Planungsproblems 3 mit dem abstrakten Plan 1. Diese ist mit 5700 *msec* jedoch geringer als die Zeit, die zum Lösen des Problems mit dem vollständigen Baum benötigt wird.
Diese Betrachtungsweise der Problemlösezeit nur im Bezug auf das konkrete Planungsproblem 3 ist dadurch gerechtfertigt, da durch das PABS-Verfahren immer alle möglichen abstrakten Pläne erzeugt werden. Wäre also der abstrakte Plan 4 in der Lage ein anderes bisher bekanntes Planungsproblem x zu lösen, wäre der abstrakte Plan 4 bei der Planabstraktion auch aus diesem Fall F_x hervorgegangen. Die Zeiten zum Re-Instantiieren und Verfeinern wären somit bei der Berechnung des Erwartungswerts der Problemlösezeit für den Knoten mit dem abstrakten Plan 4 berücksichtigt worden. Es kann somit auch passieren, daß ein Knoten der auf grund seiner schlechten Werte aus dem Baum ausgeblendet war, nach der Betrachtung neuer konkreter Planungsfälle wieder in den Baum eingeblendet wird, da sich der Durchschnitt seiner Werte verbessert und er eine größere Anzahl von konkreten Planungsproblemen löst.
Als das letzte konkrete Planungsproblem wollen wir nun den Planungsfall F_4 betrachten. Der entstehende Abstraktionsbaum wird in der Abbildung 4.5 gezeigt. Aus dem Planungs-

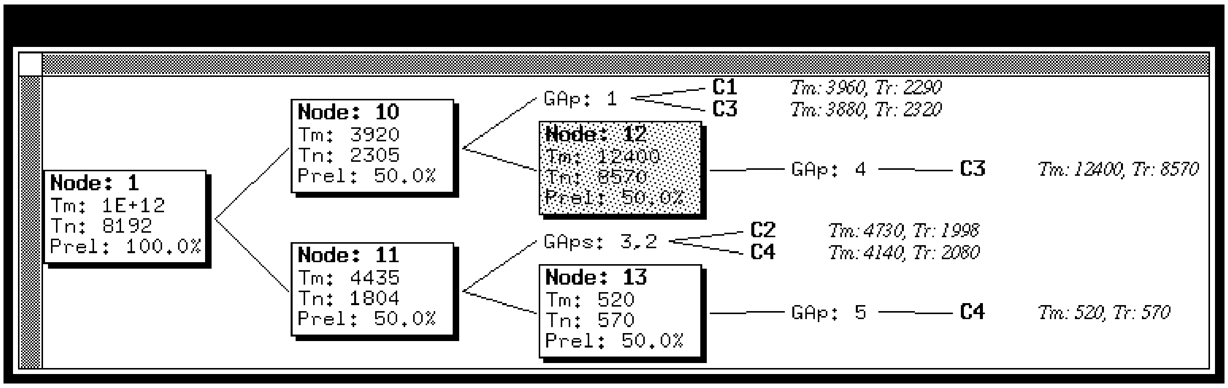

Abbildung 4.5: Der Abstraktionsbaum nach dem Lernen von Planungsfall 1,2,3 und 4

fall F_4 werden die abstrakten Pläne 2, 3 und 5 gewonnen. Somit bleiben die Wurzel und der Knoten 10 nahezu unverändert. Der interessante Knoten ist nun der Knoten 11, an den ein Knoten 13, der auch in der Abbildung sichtbar ist, durch den Algorithmus zur Bildung der Abstraktionshierarchie angehangen wird. Da dieser Knoten die Problemlösezeit des Teilbaums ab Knoten 13 minimiert, verbleibt dieser Knoten im Baum. Dazu betrachten wir wieder das Planungsproblem 4[6]. Würde der Knoten 13 aus dem Abstraktionsbaum ausgeblendet, würde der Fall mit dem abstrakten Plan 2 gelöst. Daraus resultiert eine Problemlösezeit von 6728 *msec*, die sich aus der Addition der Re-Instantiierungs- und Verfeinerungszeit ergibt. Bleibt nun der Knoten 13 eingeblendet, ergibt sich eine Problemlösezeit von 5820 *msec*. Diese Zeit resultiert aus der Re-Instantiierungszeit des abstrakten Plans 2 für den Planungsfall F_4 plus den Re-Instantiierungs- und Verfeinerungskosten für diesen Planungsfall mit dem abstrakten Plan 5. Auch hier kann der umgekehrte Fall passieren, daß später der Knoten 13 oder auch der nachfolgende neu entstehende Teilbaum durch schlechte Werte aus neuen Planungsfällen ausgeblendet wird. Bezüglich der bisher betrachteten

[6]Die Beschränkung auf dieses Planungsproblem ist mit der Argumentation beim vorherigen Planungsproblem 3 gerechtfertigt

Planungsfälle ist die Problemlösezeit jedoch minimal.
Durch dieses Beispiel sollte noch einmal die Funktion des vorgestellten Verfahrens verdeutlicht werden. Im folgenden Kapitel wird im ersten Abschnitt die verwendete anwendungsorientierte Planungsdomäne beschrieben, die für die Laufzeitmessungen verwendet wurde, damit sich der Leser ein Bild über die Art und die Komplexität der Planungsprobleme machen kann. Im zweiten Teil dieses Kapitels werden dann die durchgeführten Laufzeitmessungen in dieser realistischen Anwendungsdomäne beschrieben und deren Ergebnisse diskutiert.

Kapitel 5

Die Planungsdomäne und die Laufzeitmessungen der Indexierungsverfahren

In diesem Kapitel wird im ersten Abschnitt die Planungsdomäne vorgestellt. Da das PARIS-Planungssystem ein domänenunabhängiges Planungssystem ist, wird die Domäne in einer Spezifikationssprache vorgegeben. Die formale Definition dieser Domäne befindet sich im Anhang der Arbeit. Eine Erklärung der Semantik der verwendeten Sprache zur Definition von Planungsdomänen und der Abstraktionstheorie für das PARIS-Planungssystem findet man in [Wilke, 1993]. Im zweiten Abschnitt dieses Kapitels werden dann die durchgeführten Laufzeitmessungen erklärt, ihre Ergebnisse dargestellt und bewertet.

5.1 Die Beschreibung der Planungsdomäne

Bei der verwendeten Domäne handelt es sich um Planungsprobleme aus dem Bereich Maschinenbau. Hierbei geht es um die Produktionsplanung bei der Herstellung von rotationssymmetrischen Drehteilen. Die Planungsdomäne wurde von dem CabPlan-System [Paulokat *et al.*, 1992] übernommen. Dort wird aus einem Rohling dessen geometrische Beschreibung und Beschaffenheit den Startzustand des Planungsproblems darstellt, ein rotationssymetrisches Drehteil zu fertigen, dessen Beschreibung durch den Zielzustand repräsentiert wird. Ein Plan zur Lösung des Planungsproblems besteht nun aus einer Reihenfolge von Operationen, die eine Drehmaschine ausführen muß, um aus dem Rohling das fertige Werkstück herzustellen, also den Startzustand in den Zielzustand transformiert. In der Abbildung 5.1 ist nun so ein recht einfaches Werkstück der Domäne dargestellt. Dieses Werkstück wird aus dem Rohling gefertigt, der durch das äußere Rechteck markiert ist. In der Mitte des Rohlings ist die Rotationsachse des Werkstücks eingezeichnet, um die der Rohling sich in der Werkbank bei der Fertigung dreht. Das Werkstück ist fertiggestellt, sobald die schraffierten Flächen des Rohlings entfernt worden sind. Dazu wird der Rohling in Rotation versetzt und mit einem Schnittmeißel werden die schraffierten Flächen entfernt. Für verschiedene Flächen, Materialien und Schnittrichtungen müssen dabei verschiedene Meißel eingesetzt werden. Um die verschiedenen Positionen des Rohlings zu bezeichnen, wird ein Koordinatensystem über das Werkstück gelegt, welches die markanten Punkte des

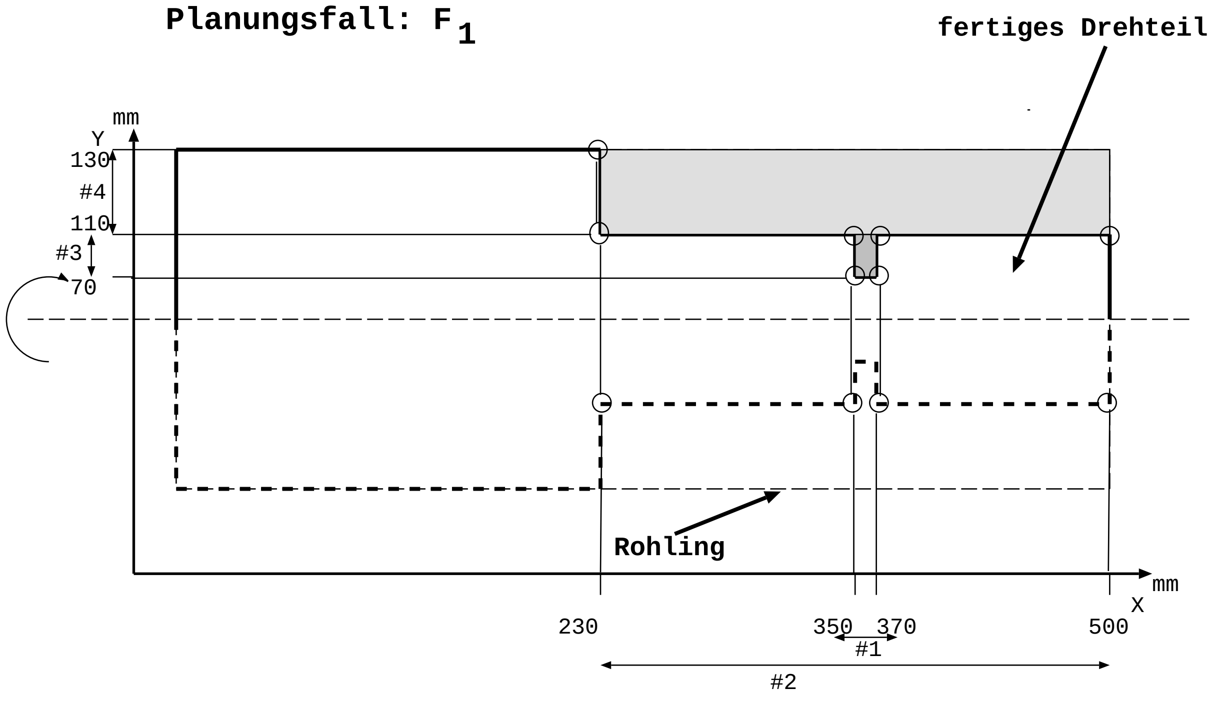

Abbildung 5.1: Ein Planungsfall F_1 aus der Domäne zur Herstellung rotationssymmetrischer Drehteile

Werkstücks enthält. Um das Planungsmodell zu diskretisieren, werden nun die Punkte des Koordinatensystems mit Linien verbunden. Diese Linien können einfach aus den Punkten des Koordinatensystems berechnet werden.
Schon bei einem ersten Blick auf das Werkstück fällt auf, daß die Flächen nicht in beliebiger Reihenfolge entfernt werden können. So muß erst die Fläche entfernt werden, die durch die Linien (#2, #4) umgeben wird, bevor die zweite Fläche entfernt werden kann. Des weiteren kann der Rohling nicht immer eingespannt werden, falls der Teil des Rohlings, der durch das Einspannfutter umschlossen wird, schon bearbeitet ist. Auf diese Fläche, die durch die Einspannung verdeckt ist, kann natürlich mit dem Schnittmeißel nicht zugegriffen werden.
Ich möchte nun die Modellierung der Domäne in einer konkreten und einer abstrakten Planungswelt beschreiben. In der konkreten Planungswelt gibt es nun vier verschiedene Operatoren, die auf ein Werkstück angewendet werden können. Ein Plan, um ein Werkstück zu fertigen, besteht nun aus einer Sequenz von diesen Operatoren. Um das Teil aus der Abbildung 5.1 zu fertigen, muß folgende Operatorsequenz ausgeführt werden:

1. *chuck(left)*
2. *set_tool(right,10,t2)*
3. *cut(#2,#4)*
4. *set_tool(center,10,t2)*
5. *cut(#1,#3)*

6. *unchuck*

Der erste Operator *chuck(left)* spannt das Werkstück an der linken Seite des Rohlings ein. Mit dem Operator *set_tool(right,10,t2)* wird nun ein Schnittmeißel in die Drehbank eingespannt. Der Parameter *right* bezeichnet die Schnittrichtung des Meißels, *10*, die maximal zulässige Drehgeschwindigkeit und *t2* ist ein Schlüssel, um die Art des Meißels zu klassifizeren. Mit dem Operator *cut(#2,#4)* wird dann die Fläche entfernt, die von den beiden Linien (#2, #4) umschlossen wird. Dann wird mit *set_tool(center,10,t2)* der Schnittmeißel gewechselt, um den Einstich in der Mitte des Werkstücks vorzunehmen. Dies wird mit der Operation *cut(#1,#3)* ausgeführt. Zum Schluß wird das Werkstück mit *unchuck(left)* ausgespannt. Somit ist der Zielzustand erreicht und das Drehteil ist fertig.
In der abstrakten Planungswelt gibt es nun vier verschiedene Operatoren. Dort wird vom ein-, Aus- und Umspannen des Werkstücks abstrahiert. Somit gibt es nur noch einen Operator *abs_chuck(Area)*, der diese Funktion ausführt. Neben den Werkzeugwechseln wird auch von den einzelnen Schnittoperationen abstrahiert. So sorgt die Ausführung des Operators *make_ready(Area)*, daß ein ganzer Bereich des Werkstücks fertiggestellt wird. Hierbei kann mit der Ausführung der Operatoren *make_raw(Area)* und *make_fine(Area)* zwischen einer groben und einer endgültigen Bearbeitung des Werkstücks unterschieden werden. Die Wegnahme des ersten Materialteils auf der konkreten Planungsebene ist zum Beispiel mit einer groben Bearbeitung des Werkstücks verbunden, da danach das Werkstück noch eingespannt werden könnte, da das Spannfutter auf der rechten Seite noch genügend Halt hat und durch die Einspannung das Werkstück nicht zerstört wird. Wäre aber in einem Arbeitsgang ein Gewinde auf diese Fläche geschnitten worden, hätte das Werkstück nicht mehr von rechts in die Drehmaschine eingespannt werden dürfen. Dies hätte dann der abstrakten Operation *make_fine(Area)* entsprochen. Auch der Einschnitt, der im zweiten Schnitt auf der konkreten Planungsebene gefertigt wurde ist so eine kritische Operation, da danach beim Einspannen auf der rechten Seite und beim Bearbeiten der linken Seite der Rohling brechen kann. Werden auf einer Seite jedoch nur unkritische Operationen ausgeführt oder der Rohling muß auf der Seite, die fertiggestellt wird, nicht mehr eingespannt werden, kann auch der abstrakte Operator *make_ready(Area)* benutzt werden. Mit *Area* wird immer ein Bereich des Werkstücks wie *left* oder *right* bezeichnet.
Aus dem eben vorgestellten konkreten Plan würde nun mit der Abstraktionsabbildung zum Beispiel der folgende abstrakte Plan hergeleitet:

1. *abs_chuck(left)*
2. *make_raw(right)*
3. *make_fine(right)*
4. *abs_chuck(none)*

Würde auf der konkreten Ebene kein kritischer Schnitt vorgenommen, könnten der zweite und der dritte Operator auch durch *make_ready(right)* ersetzt werden. In diesem speziellen Fall ist dies auch möglich, da das Werkstück nicht mehr auf der rechten Seite eingespannt werden muß, da die linke Seite unbearbeitet bleibt. So wäre ein weiterer abstrakter Plan, der aus dem Beispiel resultiert:

1. *abs_chuck(left)*
2. *make_ready(right)*
3. *abs_chuck(none)*

Diese abstrakten Pläne können nun mit der Verfeinerungskomponente des Planungssystems PARIS wieder benutzt werden, um neue konkrete Planungsprobleme zu lösen. Dieser Sachverhalt ist nun noch einmal in der Abbildung 5.2 dargestellt. Dort ist wieder der konkrete

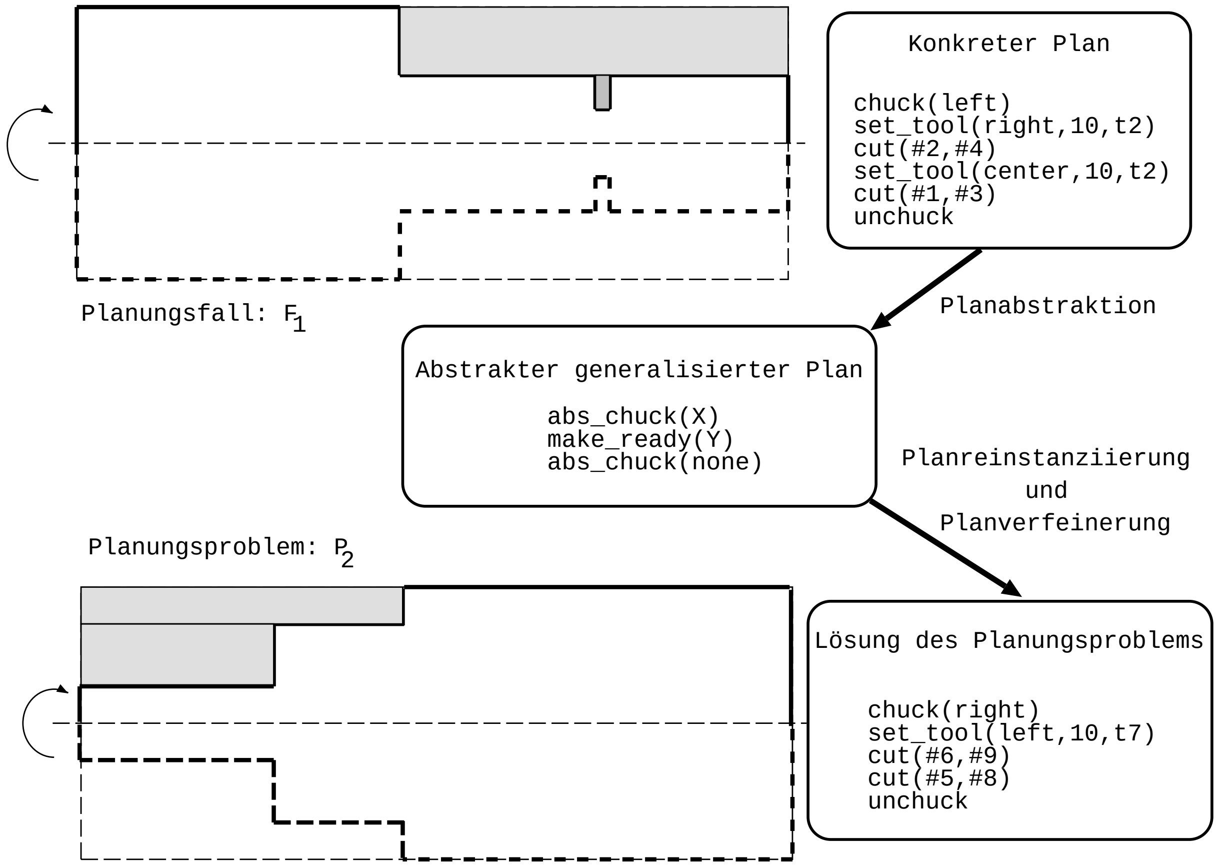

Abbildung 5.2: Die Planabstraktion des Planungsfalls F_1 und Planverfeinerung auf Planungsproblem P_2

Planungsfall F_1 oben im Bild dargestellt. Die dazugehörenden Pläne werden nun nach der PABS-Methode abstrahiert und generalisiert. Einer der aus diesem Planungsfall hervorgegangenen abstrakten, generalisierten Pläne ist zwischen den beiden Werkstücken dargestellt. Wird nun versucht, ein neues konkretes Planungsproblem P_2, welches im unteren Teil der Abbildung dargestellt ist, mit der Verfeinerungskomponente zu lösen, erhält man den dort abgebildeten Plan. Dies sollte noch einmal kurz die Vorgehensweise des Planungssystems PARIS in der Domäne verdeutlichen. Eine Definition der Planungswelten befindet sich im Anhang. In dieser Planungsdomäne wurden nun Planungsprobleme gelöst, um die verschiedenen Indexierungsverfahren der abstrakten Pläne in der Planbibliothek zu vergleichen. Im folgenden Abschnitt werden nun die durchgeführten Laufzeitmessungen vorgestellt und ihre Ergebnisse dargestellt und diskutiert.

5.2 Die Laufzeitmessungen mit den Indexierungsverfahren

In diesem Abschnitt werden die durchgeführten Laufzeitmessungen beschrieben, die in der Domäne der rotationssymmetrischen Drehteile durchgeführt wurden.

5.2.1 Verschiedene Abstraktionen durch verschiedene Domänentheorien

Bei den Laufzeitmessungen der Indexierungsverfahren wurden zwei verschiedene Abstraktionsabbildungen betrachtet, die durch verschiedene abstrakte Weltbeschreibungen und generische Theorien vorgegeben werden.

- Zum einen die im vorherigen Abschnitt beschriebene Abstraktionsabbildung, die eine "echte" Abstraktion von der konkreten Planungswelt in die abstrakte Planungswelt darstellt. Diese wurde bereits im vorherigen Abschnitt ausführlich beschrieben.

- Zum anderen wurde eine Abstraktionabbildung definiert, die die Identität realisiert. Hierbei wurden die konkreten Operatoren mit ihren essentiellen Sätzen und die Regeln der konkreten Planungswelt auf die abstrakte Welt durch einfache Umbenennung übertragen.

Die Eigenschaften der abstrakten Pläne, die aus der „echten" Abstraktionsabbildungen und aus der identischen Abstraktionsabbildung hervorgehen, werden im Folgenden noch einmal kurz charakterisiert.

Die Eigenschaften der Pläne der „echten" Abstraktionsabbildung

Durch die „echte" Abstraktionsabbildung werden abstrakte Pläne erzeugt, die durch wenig Relationen charakterisiert sind. Somit sind sie einfach auf ihre Anwendbarkeit hin überprüfbar. Aus verschiedenen konkreten Planungsfällen entstehen mit dieser Abstraktionsabbildung dieselben abstrakten Pläne. Dies hat zur Folge, daß sich mit diesen generalisierten, abstrakten Pläne mehrere konkrete Planungsprobleme durch Re-Instanziierung und Verfeinerung lösen lassen. Außerdem entstehen mit dieser Abbildung aus einem konkreten Planungsfall nur wenige abstrakte Pläne. So werden wenige abstrakte Pläne in der Planbiliothek gespeichert, die zum Problemlösen von vielen konkreten Planungsproblemen eingesetzt werden können. Die abstrakten Pläne, die aus der „echten" Abstraktionsabbildung aus einem konkreten Planungsfall hervorgehen, haben eine kürzere Planlänge als der konkrete Plan des Planungsfalls aus dem sie hervorgegangen sind. So wird jeder abstrakte Operator durch eine Sequenz von konkreten Operatoren verfeinert. Somit tragen sowohl die Re-Instanziierung als auch die Verfeinerung zur Problemlösung bei, was für eine gute Dekomposition des Aufwands bei der Lösung des Planungsproblems sorgen sollte. Insgesamt sind die Pläne dieser Abstraktionsabbildung gut geeignet, um konkrete Planungsprobleme mit dem PARIS-Planungssystem zu lösen.

Die Eigenschaften der Pläne der identischen Abbildung

Die abstrakten Pläne, die aus der identische Abstraktionsabbildung hervorgehen, werden durch viele Relationen charakterisiert. Die Re-Instanziierung dieser Pläne ist komplexer, als die der Pläne, die aus der „echten“ Abstraktion hervorgehen. Die abstrakte Operatorlänge der abstrakten Pläne entspricht der Anzahl der konkreten Operatoren des Plans aus dem konkreten Planungsproblem. So ist der Verfeinerungsaufwand beim Problemlösen mit diesen Plänen sehr gering. Da bei der identischen Abstraktionsabbildung die konkrete Planungswelt auf die abstrakte Planungswelt übertragen wird, wird aus jedem konkreten Planungsfall ein spezifischer Plan generiert. So werden sehr viele verschiedene Pläne in der Planbiliothek gespeichert. Mit diesen Plänen läßt sich ein konkreter Planungsfall dadurch lösen, daß eine geeignete Re-Instanziierung des abstrakten Plans gefunden wird. Die re-instanziierten, abstrakten Operatoren müssen jedoch nicht mehr durch eine Sequenz von konkreten Operatoren verfeinert werden. Da es sich bei diesen Pläne um sehr spezielle Pläne handelt, die nur re-instanziiert, aber nicht verfeinert werden müssen, sind diese Pläne weniger gut geeignet, um Planungsprobleme mit dem PARIS-Planungssystem zu lösen. Dies liegt daran, daß die zusätzlich aufgewendete Re-Instanziierungszeit größer ist, als der Gewinn durch die geringere Verfeinerungszeit, die aus diesen Plänen resultiert.

5.2.2 Die Laufzeitmessungen

Es wurden zwei Laufzeitmessungen mit 50 Planungsfällen durchgeführt. Einmal wurde nur die „echte“ Abstraktion zum Generieren von abstrakten Plänen benutzt, die wenige Pläne liefert, die aber gut für das Problemlösen mit Re-Instanziierung und Verfeinerung geeignet ist. Bei der zweiten Messung wurde die „echte“ Abstraktion um die identische Abbildung erweitert. So entstehen aus der Planabstraktion der konkreten Planungsfälle sowohl die guten Pläne der ersten Laufzeitmessung als auch die schlechteren Pläne, die aus der identische Abbildung resultieren.

Die Ergebnisse bei der Anwendung der „echten“ Abstraktionsabbildung

Bei den Messungen wurde auf die 50 Planungsfälle die „echte“ Abstraktionstheorie angewandt. Die drei beschriebenen Indexierungsverfahren[1] wurden nun dazu benutzt, mit den so entstandenen abstrakten Plänen die Planbibliothek zum Problemlösen zu indexieren. Danach wurden dieselben 50 Planungsfälle wieder mit den Planbibliotheken gelöst und es wurden die Zeiten gemessen, die für die gesamte Problemlösung benötigt wurde. Für die verschiedenen konkreten Planlängen der konkreten Planungsprobleme wurde dann der Mittelwert der Problemlösezeit berechnet. Die so erhaltenen Werte sind in der Abbildung 5.3 dargestellt. An der x-Achse des Graphen sind die verschiedenen konkreten Planlängen aufgetragen. Auf der y-Achse befinden sich die gemittelte Zeiten, die insgesamt zur Lösung der Planungsprobleme der jeweiligen konkreten Planlänge benötigt wurden. Die verschiedenen Indexierungsverfahren sind durch die verschiedenen Kurven gekennzeichnet. Um das Problemlöseverhalten des gesamten Systems zu bewerten, kann man den Mittelwert der Zeiten

[1] Lineare Anordnung der Pläne, inkrementelles Lernen von Abstraktionshierarchien und die Optimierung der Abstraktionshierarchie.

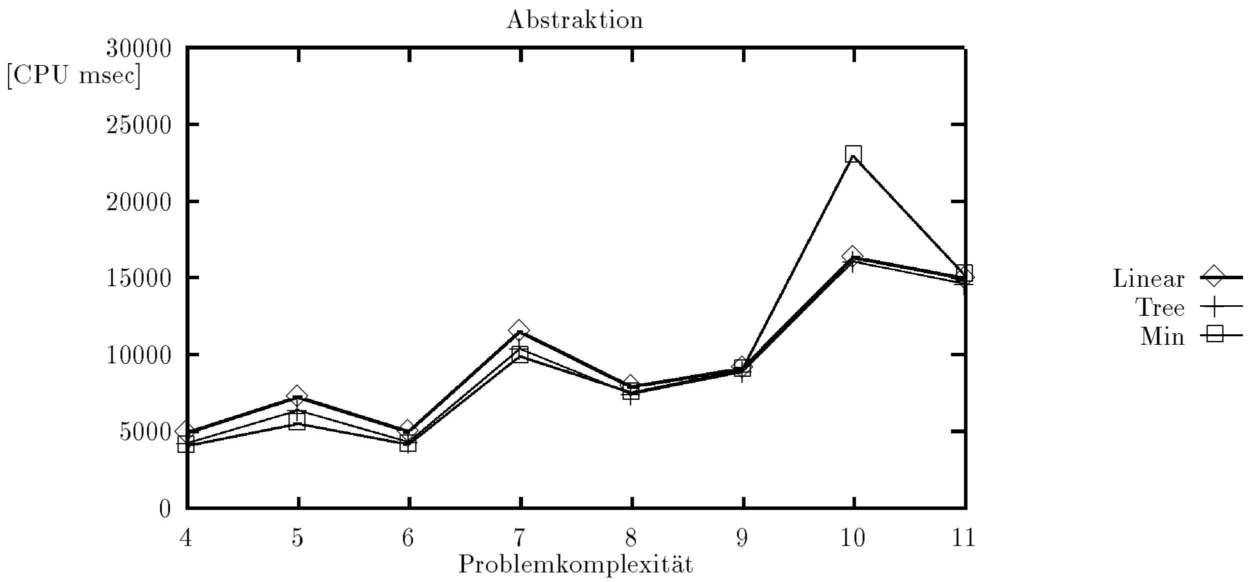

Abbildung 5.3: Die Ergebnisse der Laufzeitmessungen mit der „echten“ Abstraktionstheorie

bilden, die zum Lösen der 50 Planungsfälle benötigt wurden. Die Werte für die verschidenen Indexierungsansätze sind der Tabelle 5.1 zu entnehemen.

Indexierungsart	mittlere Problemlösezeit
lineare Anordnung	7777.6 *msec*
Abstraktionshierarchie	7111 *msec*
optimierte Abstraktionshierarchie	7104.4 *msec*

Tabelle 5.1: Die mittleren Problemlösezeiten für die verschidenen Indexierungsansätze bei der Verwendung der „echten“ Abstraktionsabbildung

Die geringen Unterschiede erklären sich durch Pläne, die aus der „echten“ Abstraktion hervorgehen. Es sind zum einen sehr wenig Pläne (7), sodaß der Effekt der Indexierungsverfahren gering ist. Zum anderen sind es sehr gute Pläne, die sich im Problemlöseverhalten nur geringfügig unterscheiden. Insgesamt zeigt sich, daß die lineare Anordnung der Pläne nur geringfügig schlechter ist als das Anordnen der abstrakten Pläne in einem Baum nach ihrem Abstraktionsgrad. Einen noch geringerer Performanzvorteil ergibt sich aus dem Minimieren des Erwartungswerts und der damit verbundenen Beschneidung des Baumes. Das liegt daran, daß bei der Abstraktionsabbildung von unwesentlichen Relationen auf der konkreten Ebene abstrahiert wird. So müssen für das Re-Instanziieren des Plans sehr wenig Relationen (< 10) überprüft werden, um zu entscheiden, ob ein abstrakter Plan anwendbar ist oder nicht. Da durch die Indexierungsverfahren hauptsächlich die Anzahl der Pläne minimiert wird, die auf ihre Anwendbarkeit hin untersucht werden, diese Überprüfung jedoch im Vergleich zur Verfeinerung relativ „billig“ ist, ist der geringe Performanzvorteil durch die Verfahren erklärt.
An den Stellen, wo die Problemlösezeiten der beiden Indexierungsverfahren, die die Pläne in einem Baum anordnen, annährend gleich sind, wird keine Beschneidung des Baumes vorgenommen, da der speziellste Plan im Baum eine minimale Problemlösezeit zur Folge

hat. Ist die Problemlösezeit zwischen der linearen Anordnung der Pläne und den Clusterverfahren der abstrakten Pläne in einem Baum annährend gleich, sind die längeren Pläne, die sich auf das Problem re-instanziieren ließen, auch die, die bei den Clusterverfahren in den Blättern stehen und so zum Problemlösen bei der Verfeinerung genommen werden. Bei den Re-Instanziierungskosten ($<< 1000\ msec$) spielt es kaum eine Rolle, ob zwei oder drei Pläne mehr re-instanziiert werden müssen, um ein konkretes Planungsproblem zu lösen.
Bei den Problemen mit der konkreten Planlänge 10 ist der Modus, der den Erwartungswert der Problemlösezeit minimiert, entgegen der Erwartung erheblich schlechter als die beiden anderen Verfahren. Diese Situation wird im Folgenden analysiert. Es existieren nur zwei Planungsfälle mit der konkreten Planlänge 10. Der eine Planungsfall läßt sich mit vier verschiedenen abstrakten Plänen lösen, die zur Lösung des Falls alle in etwa dieselbe Zeit benötigen ($< 6\ sec$). Somit ist es bei diesem Fall relativ unrelevant, welcher abstrakte Plan zur Lösung benutzt wird. Der zweite Planungsfall verursacht jedoch den schlechten Wert für die Problemlösezeit. Dieser Fall läßt sich mit zwei verschiedenen abstrakten Plänen lösen, von denen der eine ca. 5 sec und der andere ca. 28 sec benötigt. Im linearen Modus ist der abstrakte Plan mit der kürzeren Problemlösezeit vor dem Plan mit der langen Problemlösezeit angeordnet, weil dieser eine längere konkrete Operatorsequenz besitzt. Somit wird dieser Plan zum Problemlösen benutzt und sorgt für den guten Wert bei der Problemlösezeit. Bei dem Plan mit der kurzen Problemlösezeit handelt es sich jedoch um einen sehr speziellen Plan, der nur insgesamt vier der fünfzig Planungsfälle löst. Bei dem Plan mit der für diesen Fall längeren Problemlösezeit handelt es sich jedoch um einen Plan, der bezüglich $\sqsubseteq_{\mathcal{F}}$ allgemeiner als der vorherige Plan ist. Mit ihm lassen sich achtunddreizig der fünfzig Planungsfälle lösen. Beim Clustern der abstrakten Pläne wird nun der speziellere Plan zur Lösung des Problems benutzt, der für die gute Lösungszeit sorgt. Beim Verfahren zur Minimierung der Problemlösezeit wird der Knoten mit dem speziellen Plan jedoch abgeschnitten, weil es sich zum einen um einen sehr speziellen Plan handelt, der nur sehr wenige Fälle löst und zum anderen sich seine Problemlösezeiten bis auf diesen einen speziellen Fall nur unwesentlich unterscheiden. Dieser eine Fall fällt jedoch bei der Planlänge insbesondere auf, da nur zwei Pläne der Länge 10 in der Fallbasis sind und der Unterschied der Problemlösezeiten relativ groß ist. In den 34 Fällen, in denen sich ein konkreter Planungsfall nur mit dem abstrakten Plan des oberen Knoten verfeinern läßt, aber nicht mit dem unteren, wird jedoch ein abstrakter Plan weniger re-instanziiert, was jedoch bei den geringen Zeiten, die dafür benötigt werden, kaum ins Gewicht fällt, aber insgesamt zu einem besseren Problemlöseverhalten des Systems bezüglich aller fünfzig Planungsprobleme zur Folge hat, wie dies aus der Tabelle 5.1 zu entnehmen ist.

Die Ergebnisse bei der Anwendung der „echten“ Abstraktionsabbildung zusammen mit der identischen Abstraktionsabbildung

Um Pläne verschiedener Güte zu erhalten, wurde die „echten“ Abstraktion um die identische Abbildung erweitert. Dort werden die Unterschiede der verschiedenen Indexierungsverfahren offensichtlicher. Dazu wurden die konkreten Fakten, Regeln und Operatoren aus der konkreten Planungswelt einfach in der abstrakten Welt umbenannt. So entsteht eine Abstraktionsabbildung, die die identische Abbildung realisiert. Diese Abstraktionsabbildung wurde mit der „echten“ Abstraktionabbildung vereinigt. Aus dieser so entstandenen Abstraktionsabbildung gehen sowohl die Pläne der ersten Abstraktionsabbildung hervor als auch die Pläne, die aus der identischen Abbildung resultieren. Mit dieser Abstraktionsabbildung wurden die gleichen 50 Fälle mit den drei Verfahren gelöst. Die Ergebnisse sind der Abbildung 5.4 zu entnehmen. Beim Vergleich der Abbildung 5.3 mit der Abbildung 5.4 sind

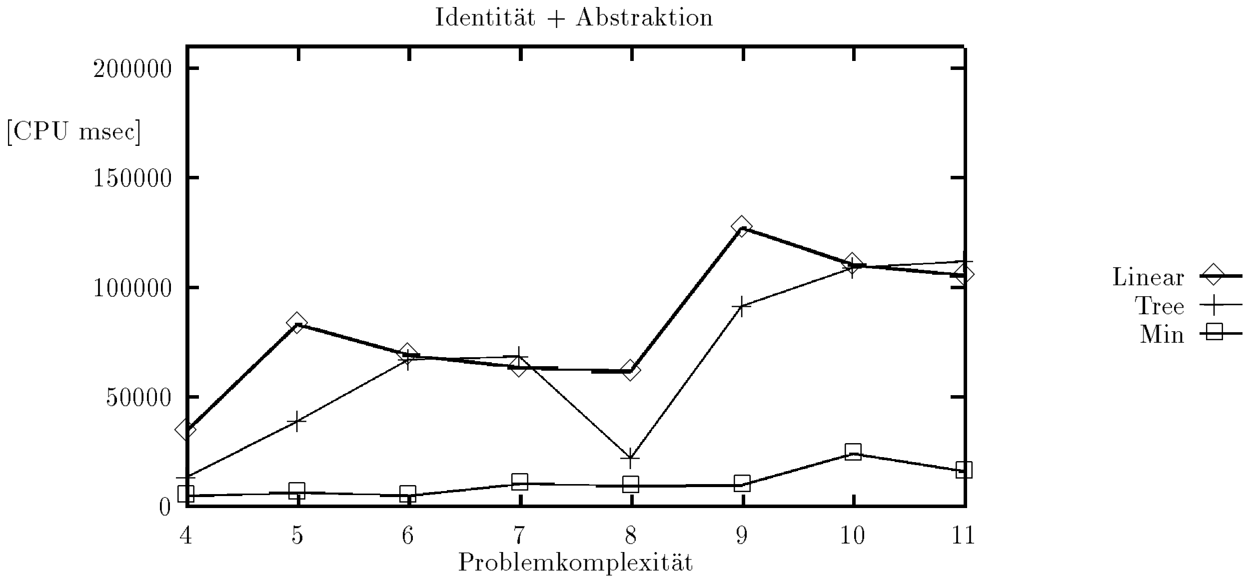

Abbildung 5.4: Die Ergebnisse der Laufzeitmessungen der Identität und der Abstraktion

die unterschiedlichen Maßstäbe auf der y-Achse zu beachten. Um das Problemlöseverhalten des gesamten Systems zu bewerten, kann man wieder den Mittelwert der Zeiten bilden, die zum Lösen der 50 Planungsfälle benötigt wurden. Die Werte für die verschidenen Indexierungsansätze sind der Tabelle 5.2 zu entnehemen. Mit dieser neuen Abstraktionstheorie

Indexierungsart	mittlere Problemlösezeit
lineare Anordnung	65454.8 $msec$
Abstraktionshierarchie	43639 $msec$
optimierte Abstraktionshierarchie	7712.8 $msec$.

Tabelle 5.2: Die mittleren Problemlösezeiten für die verschidenen Indexierungsansätze bei der Verwendung der „echten“ Abstraktionsabbildung zusammen mit der identischen Abbildung

wurden insgesamt 50 abstrakte Pläne aus den konkreten Planungsfällen hergeleitet. Daß diese Anzahl mit der Anzahl der konkreten Planungsfälle übereinstimmt, ist rein zufällig und hat nichts mit der identischen Abbildung in der Abstraktionsabbildung zu tun. Die Pläne aus dieser Abstraktionsabbildung sind, neben den gleichen Plänen wie bei der ersten Messung, Pläne, die höhere Re-Instanziierungskosten haben. Dies liegt an den größeren Relationenmengen (bis zu 45), die die Anwendbarkeit eines abstrakten Plans zur Verfeinerung determinieren. Somit werden die Unterschiede der verschiedenen Verfahren deutlicher, da der Aufwand beim Re-Instanziieren der Pläne stärker ins Gewicht fällt, falls ein oder mehrere Pläne auf ihre Anwendbarkeit überprüft werden müssen. Dies ist immer der Fall, falls sich die abstrakten Pläne in der Anzahl ihrer Relationen stark unterscheiden.

Die Indexierung der abstrakten Pläne nach ihrer Planlänge in einer linearen Liste sorgt hier für die längsten Problemlösezeiten, genau wie bei der vorherigen Messung. Bei den konkreten Planlängen 7 und 10 ist das lineare Verfahren geringfügig besser als die Anordnung der Pläne in einem Abstraktionsbaum. Hier stehen die Pläne zu Lösung der Planungsprobleme relativ weit vorne in der linearen Liste, so daß sehr wenige Pläne re-instanziiert werden müssen bis ein Plan zum Lösen des Planungsproblems gefunden wird. Beim Baumverfahren werden die speziellsten Pläne, die in den Blättern des Abstraktionsbaums stehen, zum Problemlösen benutzt. Stehen nun mehr Pläne, bzw. Pläne mit höheren Re-Instanziierungskosten auf diesem Pfad von der Wurzel des Abstraktionsbaumes zum Blatt, so werden die Zeiten für die Problemlösung schlechter als beim linearen Verfahren. Dies ist natürlich eine Durchschnittsbetrachtung, weil die Werte über die konkreten Planlängen gemittelt werden. Bei dieser Abstraktionsabbildung, die die Identität realisiert, sind die speziellsten Pläne auch die längsten, sodaß dieselben Pläne bei den Verfahren zur Problemlösung benutzt werden. Dies muß jedoch nicht bei jeder Abstraktionsabbildung so sein.
Diese beiden Verfahren sind jedoch deutlich schlechter als das Verfahren zum Minimieren des Erwartungswerts. Dieses Verfahren sichert zu, daß sich zum Problemlösen die Summe der Zeiten zum Re-Instanziieren und Verfeinern des abstrakten Plans, bezüglich aller betrachteten Planungsfällen minimiert. So werden nur abstrakte Pläne re-instanziiert, bis das Re-Instanziieren eines neuen abstrakten Plans eine Verkürzung der gesamten Problemlösezeit durch dessen Verfeinerung zur Folge hat. So werden hier die Pläne benutzt, die aus der „echten" Abstraktion entstanden sind, da die Verfeinerungszeit dieser Pläne weniger Zeit verbraucht als das Re-Instanziieren eines spezielleren Plans, der aus der identischen Abbildung resultiert. Diese spezielleren Pläne haben zwar geringere Verfeinerungszeiten, aber so hohe Re-Instanziierungskosten, daß dieser Zeitvorteil durch das Re-Instanziieren verloren geht. Daß die Pläne der „echten" Abstraktion genommen werden, sieht man auch daran, daß die Problemlösezeiten der ersten und der zweiten Messung beim dritten Verfahren identisch sind[2]. Somit sind zwar durch die neue Abstraktionsabbildung speziellere Pläne hinzugekommen, die jedoch für eine Verlängerung der Problemlösezeit sorgen. Dies ist ein allgemeines Phänomen des dritten Indexierungsansatzes, daß das Hinzufügen von schlechten Plänen zur Planbibliothek keinen negativen Einfluß auf die Problemlösezeit hat.
Insgesamt zeigt sich, daß der Unterschied der verschiedenen Indexierungsverfahren sich erst dann bemerkbar macht, wenn sich in der Planbibliothek Pläne von verschiedener Güte befinden. Außerdem wirkt sich die unterschiedliche Indexierung der Pläne um so mehr auf die Problemlösezeit aus, je mehr abstrakte Pläne sich in der Planbibliothek befinden.

[2] Man beachte die verschiedenen Werte der y-Achse

Kapitel 6

Diskussion

In diesem Kapitel werden zunächst die Ergebnisse der Arbeit noch einmal zusammengefaßt und bewertet. Danach wird der in dieser Arbeit vorgestellte Ansatz mit verwandten Ansätzen verglichen, um einige Unterschiede herauszuarbeiten. Anschließend wird ein Ausblick gegeben, der einige Möglichkeiten für weiterführende Arbeiten aufzeigt.

6.1 Die Ergebnisse der Arbeit

In dieser Diplomarbeit wurden drei verschiedene Verfahren zur Planbewertung und Planselektion im Planungssystem PARIS entworfen, implementiert und experimentell untersucht. Dabei handelt es sich um:

- die lineare Anordnung von abstrakten Planen, wobei die Pläne nach ihrer abstrakten Operatorlänge geordnet werden. Zur Planselektion wird diese lineare Anordnung sequentiell aufsteigend vom längsten abstrakten Plan durchlaufen, um einen möglichst langen Plan zum Problemlösen zu benutzen. Die Idee zu diesem Ansatz resultierte daraus, daß lange abstrakte Pläne für einen möglichst geringen Verfeinerungsaufwand sorgen. Die entstehenden Re-Instanziierungskosten und die Anzahl der Pläne, die bei der Auswahl eines Plans zur Problemlösung betrachtet werden, werden bei diesem Ansatz vernachlässigt.

- das inkrementelle Lernen von Abstraktionshierarchien, wo die abstrakten Pläne nach ihrem Abstraktionsgrad bewertet wurden. Die Pläne wurden dann in einer Abstraktionshierarchie, die durch eine Baumstruktur repräsentiert wird, organisiert. Bei der Planselektion wird diese Abstraktionshierarchie in Vorordnung durchlaufen, um einen möglichst speziellen Plan zur Problemlösung einzusetzen. Bei diesem Ansatz wurde versucht, sowohl die Anzahl der zu betrachteten Pläne zu minimieren, als auch die Verfeinerungskosten zu reduzieren, indem ein möglichst spezieller Plan für das zu lösende Problem eingesetzt wurde.

- das Optimieren der Abstraktionshierarchie, wobei der Erwartungeswert der Problemlösezeit mit der Abstraktionshierarchie aus dem zweiten Ansatz bezüglich der bekannten Planungsprobleme minimiert wird. Dabei werden die Pläne primär nach ihrem Abstraktionsgrad bewertet, um sie korrekt in der Abstraktionshierarchie einzuordnen. Sekundär werden die gemessenen Re-Istanziierungs- und Verfeinerungszeiten der abstrakten Pläne für die bekannten Planungsfälle dazu benutzt, um die so entstandene Abstraktionshierarchie zu optimieren. Es wird bei dieser Planbewertung die Re-Instanziierungszeit, die Verfeinerungszeit und die Anzahl der Pläne zum Auffinden einer Lösung betrachtet. Bei der Planselektion wird die optimierte Abstraktionshierarchie in Vorordnung durchlaufen, um einen möglichst optimalen Plan zum Problemlösen einzusetzten. Optimal heißt hier nicht optimal im bezug auf ein spezielles Planungsproblem, sondern auf das bisher bekannte Problemspektrum der Anwendung, gegeben durch die bisher bekannten Planungsfälle.

Um die Nützlichkeit der drei beschriebenen Ansätze zu beurteilen, wurden sie in dem Planungssystem PARIS implementiert. Anschließend wurden mit diesen drei Ansätzen die Problemlösezeiten für Planungsprobleme in einer anwendungsorientierten Domäne gemessen und bewertet.
Bei der linearen Anordnung der abstrakten Pläne handelt es sich um einen naiven Ansatz zur Planbewertung und Planselektion. Bei einer, für realistische Anwendungen, geringen Anzahl von Plänen resultiert aus diesem Ansatz eine erhebliche Verlängerung der Problemlösezeit. Dafür ist der Aufwand der Planbewertung sehr gering und die Organisation der Planbibliothek recht einfach zu realisieren. Dieses Verfahren ist geeignet, falls in der Anwendung nur sehr wenige abstrakte Pläne die vorkommenden Planungsprobleme lösen oder die abstrakten Pläne über sehr geringe Re-Instanziierungszeiten verfügen, also durch wenig Relationen charakterisiert sind.
Um eine gezieltere Planselektion durchzuführen, wurden die abstrakten Pläne nach ihrem Abstraktionsgrad in einer Klassifikationshierarchie angeordnet. Der Aufwand der hierfür benötigt wird, ist nur geringfügig größer als bei der Organisation der abstrakten Pläne in einer linearen Liste. Dadurch wurden zum einen bei der Planselektion eine geringere Anzahl von Plänen re-instanziiert und zum anderen ein möglichst spezieller Plan für das zu lösende Problem zur Problemlösung benutzt. Es zeigte sich, daß diese Organisation der Planbibliothek für eine erheblich größere Anzahl von abstrakten Pläne effektiv war und das immer ein spezieller Plan für die einzelnen Planungsprobleme gefunden wurde. Ab einer gewissen Tiefe der Abstraktionshierarchie werden jedoch noch immer zu viele abstrakte Pläne re-instanziiert, obwohl schon ein allgemeinerer Plan, der bereits auf seine Anwendbarkeit hin überprüft wurde, für eine Problemlösung in kürzerer Zeit sorgt.
Um die beobachteten Nachteile des zweiten Verfahrens zu beseitgen, wurde der Baum, der die Klassifikationshierarchie für abstrakte Pläne repräsentiert, beschnitten. So kann die Abstraktionshierarchie nur noch so tief werden, daß sie für ein optimales Problemlöseverhalten im bezug auf die bekannten Planungsprobleme sorgt. Der Aufwand für diese Beschneidung ist jedoch um einiges größer als bei den beiden ersten Verfahren, da:

- ein Planungsfall noch einmal mit allen abstrakten Plänen, die aus ihm hervorgehen, gelöst werden muß, um die abstrakten Pläne zu bewerten.
- die Beschneidung der Abstraktionshierarchie an sich ein komplexes Problem darstellt.

Es zeigte sich, das dieses Verfahren für ein besseres Problemlöseverhalten sorgte, als die unbeschnittene Abstraktionshierarchie. Außerdem werden durch dieses Verfahren die Pläne aus der Klassifikationshierarchie entfernt, die für ein schlechteres Problemlöseverhalten im bezug auf die bekannten Planungsprobleme sorgen. Abhängig von der Anzahl der abstrakten Pläne, die aus der Planabstraktion entstehen, kann sich jedoch die Optimierung der Abstraktionshierarchie sehr aufwendig gestalten. Um die Komplexität der Beschneidung der Abstraktionshierarchie zu reduzieren, wurde die bereits diskutierte Annahme getroffen, daß die konkreten Planungsfälle, die zu abstrakten Plänen von Geschwisterknoten der Hierarchie gehören, disjunkt sind. Ist dies nicht der Fall, muß die Beschneidung der Hierarchie global durchgeführt werden, was ebenfalls eine erhebliche Steigerung des Aufwands bei der Organisation der Planbibliothek verursacht.

6.2 Der Vergleich mit anderen Verfahren

6.2.1 Die Formale Begriffsanalyse

Der Algorithmus zum inkrementellen Lernen von Abstraktionshierarchien, erwirbt im wesentlichen eine Partialordnung über abstrakte Pläne aus Beispielen. Die so erworbene Partialordnung repräsentiert die - spezieller als - Relation über den bekannten abstrakten Plänen;sie klassifiziert also die abstrakten Pläne nach ihrem Abstraktionsgrad. Als Beispiele für den Algorithmus werden dazu konkrete Planungsprobleme mit den aus ihnen hervorgehenden abstrakten Planmengen benutzt.
Im Rahmen der *formalen Begriffsanalyse* [Ganter, 1987; Wille, 1992] wird ebenfalls ein Algorithmus vorgestellt, der es ermöglicht diese Partialordnungen über den abstrakten Plänen zu erlernen. Mit diesem Algorithmus ist es möglich, wesentliche Beziehungen zwischen endlich vielen Merkmalen für unendlich viele Objekte eines Bereiches zu extrahieren und in einem endlichen Kontext, dem Begriffsverband, darzustellen. In diesem Verband wird dann eine Ordnungsrelation - Unterbegriff von, bzw. Oberbegriff von - für die Begriffe des gegebenen Kontextes dargestellt. Durch die Assoziation von

- abstrakten Plänen mit Begriffen
- von Merkmalen mit den Relationen, die die Anwendbarkeit eines abstrakten Plans determinieren und
- vom Begriffsverband mit der Klassifikationshierarchie für abstrakte Pläne

wäre dieser Algorithmus dazu geeignet, die - spezieller als - Relation aus Beispielen zu erwerben. Dieses Verfahren arbeitet jedoch nicht inkrementell, sodaß der Algorithmus nach jedem neuen Beispiel auf die ganzen bisher bekannten Beispiele erneut angewendet werden muß. Es können zwar zu bestehenden Konzepten neue Objekte hinzugefügt werden, was dem Einsortieren von neuen abstrakten Plänen in bereits vorhandende Knoten der Abstraktionshierarchie entspricht. Es ist jedoch nicht möglich, den Begriffsverband um ein neues Konzept zu erweitern, was dem Erzeugen neuer Knoten in der Abstraktionshierarchie entspricht.

6.2.2 Das Utility-Maß in PRODIGY

In [Minton *et al.*, 1989a] wird für das Planungssystem *PRODIGY* ein Utility-Maß dargestellt, das es ermöglicht, Aussagen über die Nützlichkeit von Kontrollwissen zu machen. Dort wird die Nützlichkeit von generalisierten Regeln abgeschätzt durch:

$$Utility = (AvrSavings * ApplicFreq) - AvrMatchCost$$

wobei

- $AvrSavings$ die durchschnittliche Kostenersparnis darstellt, falls die Regel angewendet wird
- $ApplicFreq$ die Häufigkeit der Anwendung der Regel darstellt, falls sie auf ihre Anwendbarkeit hin überprüft wurde
- $AvrMatchCost$ die durchschnittlichen Kosten darstellen, die aufgewendet werden, um die Regel zu re-instanziieren, also auf ihre Anwendbarkeit hin zu überprüfen

Hat nun eine Regel ein negatives Utility-Maß, wird sie aus der Wissensbasis entfernt, die das Kontrollwissen zur Planung enthält, da sie für eine Verschlechterung der Zeit zum Lösen von Planungsproblemen sorgen würde. Bei der Optimierung von Abstraktionshierarchien findet ebenfalls eine Bewertung von Kontrollwissen statt. Hier werden die Zeiten zum Re-Instanziieren und Verfeinern abstrakter Pläne auf konkrete Planungsprobleme gemessen, um etwas über die Nützlichkeit dieser Pläne bei der Problemlösung auszusagen. Bei der Optimierung der Abstraktionshierarchie werden dann Pläne, die für ein schlechteres Problemlöseverhalten bezüglich der bekannten Planungsprobleme sorgen, aus der Abstraktionshierarchie entfernt. Im Gegensatz zum Utility-Maß werden die abstrakten Pläne jedoch nicht gelöscht und nie wieder benutzt. Aufgrund neuer Beispiele werden die ausgeblendeten Pläne weiterhin bewertet. Wird durch diese Neubewertung ein abstrakter Plan wieder nützlich, wird er wieder in die Abstraktionshierarchie mit aufgenommen.

6.2.3 Das Diagnosesystem MoCAS/2

In *MoCAS/2* [Pews, 1994] wird mit Hilfe von Abstraktion und Fallübertragung eine Fehlerdiagnostik an technischen Modellen durchgeführt. Dort wird Abstraktion verwendet, um eine grössere Wiederverwendbarkeit von einzelnen bekannten Diagnosefällen zu erzielen. Diese abstrakten Fälle werden dann in einer Falbasis gespeichert. Bei der Diagnose neuer Fehler in dem technischen Modell können dann durch eine Fallübertragung diese Fälle dazu benutzt werden, eine Fehlerdiagnose zu liefern. Dort wird die Ähnlichkeit zwischen Komponenten des technischen Modells durch eine Abstraktionshierarchie repräsentiert. Eine Darstellung über die Abstraktion und die Spezialisierung von Fällen im Diagnosesystems MoCAS/2 und im Planungssystem PARIS befindet sich in [Bergmann *et al.*, 1993].

6.3 Ausblick

Der in dieser Arbeit vorgestellte Ansatz läßt sich sowohl weiterentwickeln als auch auf andere Anwendungsgebiete übertragen. Die in der Modellierung verwendete Abstraktionstheorie

bietet eine guten Ansatzpunkt, um das Planungssystem PARIS zu erweitern und noch leistungfähiger zu gestalten. Die Abstraktionsabbildung kann gezielt dazu eingesetzt werden, um möglichst gute abstrakte Pläne für das Problemlösen mit Re-Instanziierung und Verfeinerung zu erzeugen. Dazu müßte man Kriterien für die Abstraktionsabbildung angeben, die garantieren, solche abstrakten Pläne nach dem Planabstraktionsverfahren zu erzeugen.
Bei dem vorgestellten Planungssystem wird in der konkreten und in der abstrakten Planungswelt ein linearer Planer verwendet. Der vorgestellte Ansatz könnte jedoch auch auf andere Planungsmethoden, wie zum Beispiel der nicht-linearen Planung, übertragen werden. Somit könnte der vorgestellte Ansatz einem allgemeineren Bereich von Planungsproblemen zugänglich gemacht werden.
Es sollten Tests mit anderen Planungssystemen durchgeführt werden, um eine Bewertung des Systems gegenüber anderen Ansätzen zu erlangen.
Durch Abstraktion eine grössere Wiederverwendbarkeit von bekannten Fällen zu bekommen und diese abstrakten Fälle auf neue Probleme zu übertragen, wurde in dieser Arbeit bei der Lösung von Planungsproblemen mit Erfolg durchgeführt. Derselbe Ansatz wurde bei der Diagnostik von Fehlern in technischen Modellen realisiert. Die Technik der Abstraktion und Verfeinerung sollte sich auch auf andere Probleme übertragen lassen, wo fallbasierte Systeme zum Einatz kommen. Dazu brauchte man einen Formalismus, der die Beschreibung eines fallbasierten Systems ermöglicht und in dem sich die Deklaration von Fallabstraktion und Fallübertragung beschreiben läßt. Diese Beschreibung könnte dann einem fallbasiertem System dienen, um Abstraktion und Verfeinerung beim fallbasiertem Schließen einzusetzen.
In dieser Arbeit wurde die Technik der Abstraktion und der Verfeinerung bei der Aktionsplanung betrachtet. Allgemein weisen jedoch viele Begriffswelten eine hierarchische Organisationsstruktur vom allgemeinen zum speziellen auf. Dort hat man dann gewöhnlich mehrere Abstraktionsebenen zur Auswahl, um auf Begriffen in der Begriffswelt zu operieren. Die Festlegung auf eine Abstraktionsebene hat nun Konsequenzen auf die Operationen, die auf diesen Begriffen möglich sind, da Wissen in diesen Begriffswelten oft nur ganz speziellen Ebenen zugeordnet ist. Es gilt also eine Ebene der Beschreibung zu finden, die sowohl alle Begriffe, die für Operationen auf der Begriffswelt benötigt werden hinreichend beschreibt, als auch diese Beschreibungen nicht zu detaliert zu formulieren, um die Operationen auf Begriffen nicht zu komplex werden zu lassen.
Die Auswahl einer geeigneten Beschreibung, wird in diesem Ansatz durch die Planbewertung des zweiten Verfahrens realisiert, indem zu detalierte Beschreibungen, also zu spezielle abstrakte Pläne in der Klassifikationshierarchie, nicht zur Problemlösung verwendet werden. Zu abstrakte Pläne hingegen werden nicht zum Problemlösen benutzt, da bei der Planselektion solange spezieller Pläne zum Problemlösen benutzt werden, bis sich ihr Einsatz nicht mehr lohnt. So wird ein Detaliertheitsgrad der Beschreibung gewählt, der möglichst gut geeignet ist, um das vorhandene Problem zu lösen.
Somit sollten die vorgestellten Ansätze nicht nur auf andere fallbasierte Systeme übertragbar sein, sondern auch eine Anregung darstellen, bei völlig anderen Problemstellungen eine geeignete Beschreibung zu finden, um Probleme effizent zu lösen.

Danksagung

Bedanken möchte ich mich für die freundliche Betreuung bei der Durchführung dieser Diplomarbeit bei Prof. Dr. M. M. Richter sowie ganz besonders bei Dipl.-Inform. Ralph Bergmann für die vielen anregenden und hilfreichen Diskussionen, die zu dieser Arbeit beigetragen haben. Mein Dank gilt ebenfalls Dagmar Surmann und Dipl.-Inform. Stefan Wess für das Korrekturlesen des Manuskripts.

Anhang A

Das PARIS Planungssystem

In diesem Kapitel wird das Planungssystem PARIS vorgestellt. Nach dem Starten von SWI-Prolog im Verzeichnis des Planers werden alle benötigten Files mit dem Programmcode in die Prologumgebung geladen. Anschließend erscheint das folgende Menü:

```
PARIS Toplevel Commands
***********************
------ LEARNING ------
learn_all(CaseFile,GAPFile) :
         Load default theory
         Learn from all cases
         Write GAP-Hierarchy into GAPFile
loadt : Ask for new theories and load them
loadc : Ask for a set of cases and load them
learnc: Ask for ONE case and and learn from it
wrskp : Write GAP-Hierarchy to a new file
ldskp : Load GAP-Hierarchy from a file
delskp: Delete GAP-Hierarchy from the Database
setht : Set the GAP-Hierarchy Type for the GAPs
setsm : Set the Sorting_type of the Nodes in the Tree
delbf : Delete the global concrete branch factor
setbf : Set the global concrete branch factor
deb   : Switch PABS-Debugging mode on/off
------ PLANNING ------
setlimit : Set node and time limit for planning
setrm    : Set the Refinement Backtracking Type
setss    : Set the Search Space for Problem Solving
log      : Ask for print information during planning
solvec   : Ask for ONE problem and solve it
plan_all(ProblemFile,ResultFile) :
         Solve all problems in the file and write ResultFile.
intlearn(CaseFile,ResultFile) :
         Solve all problems and learn afterwards.
help  : This Info
```

Diese Menü kann jeder Zeit wieder mit dem Befehl *help/0* wieder aufgerufen werden. Im Folgenden werde ich nun die einzelnen Befehle des PARIS-Planungssystems erklären. Dabei wird eine Reihenfolge gewählt, die dem Neuling einen möglichst guten Einstieg in das Planungssystem ermöglicht.
Als erste werden die Befehle erklärt, mit den die notwendigen Daten geladen werden können, die das Plannungssystem benötigt. Die Pfadnamen können hierbei absolut oder relativ von dem Punkt angegeben werden, von dem aus das PARIS-System gestartet wurde. Die Pfade müssen in einfachen Hochkommata eingeschlossen werden und die Eingabe muß mit einem Punkt abschließen.

loadt/0: Mit diesem Befehl werden die benötigten Theorien für des Planungssystem geladen. Nacheinander werden die Pfade für die konkrete, generische und die abstrakte Theorie erfragt. Ein Beispiel für diese Theorien für eine Planungsdomäne ist im Anhang wiedergegeben. Bei der Abfrage nach den Pfaden wird ein Defaultpath angegeben. Durch die Eingabe von `.<cr>` wird die Theorie geladen, die diesem Defaultpath entspricht. Will man diese Pfade auf andere Werte setzen, kann man das in dem File theories.pl tun, in dem man das jeweilige erste Argument der Klausel *defaultpath/2* ändert. Dies empfiehlt sich insbesondere bei der Entwicklung einer neuen Domäne.

loadc/0: Mit diesem Befehl wird eine Menge von konkreten Planungsfällen geladen. Diese Planungsfälle können dann abstrahiert werden und so eine Menge von generalisierten, abstrakten Plänen zu erzeugen, die dann später für die Problemlösung eingesetzt werden. Des weiteren beschreiben diese Fälle die konkreten Planungsprobleme, die mit dem Planer gelöst werden können. Der mit dem Fall angegebene Plan wird natürlich nicht bei der Problemlösung betrachtet. Die Beschreibung eines Falles besteht aus dem Prädikat *case(<Nr>,<Init>,<Goal>,<Plan>)*, wobei *<Nr>* eine eindeutige Nummer zur Identifizierung des Falls, *<Init>* und *<Goal>* die Menge von essentiellen Sätzen, die den Start- und Zielzustand beschreiben und *<Plan>* den Plan zur Lösung des Planungsproblems darstellt. Handelt es sich um einen neuen Fall, dessen konkreter Plan nicht bekannt ist, kann dieser durch eine Variable ersetzt werden. Dieser Fall muß dann jedoch mit dem Planungssystem gelöst werden, bevor er zur Planabstraktion eingesetzt werden kann. Ein Beispielfall befindet sich im nächsten Teil des Anhangs.

Wenn man nun mit diesen beiden Befehlen die beschriebenen Daten geladen hat, kann man mit dem PARIS-System arbeiten. Ein weiterer Befehl, um Daten in das Planungssystem zu laden, ist

ldskp/0: Dieser Befehl wird dazu benutzt um eine Menge von generalisierten, abstraken Plänen in das Planungssystem einzuladen. Diese Pläne müssen zuvor mit dem Befehl *wrskp/0* oder *learn_all/2* abgespeichert worden sein. Der Indexierungsmodus, mit dem die abstrakten Pläne gelernt worden sind, wird dabei automatisch gesetzt. Diese Pläne können dann zum Lösen neuer Planungsprobleme benutzt werden. Nach dem Ausführen des Befehls wird der Pfad erfragt, unter dem die abstrakten Pläne abgespeichert wurden. Die alten abstrakten Pläne im Planungssystem werden dabei automatisch gelöscht.

Als nächstes werden die beiden Möglichkeiten beschrieben, um aus konkreten Planungsfällen abstrakte Pläne zu lernen.

learnc/0: Mit diesem Befehl kann die Menge aller abstrakten Pläne aus einem konkreten Planungsproblem erzeugt werden. Diese Pläne werden dann nach dem entsprechenden Modus in die Planbibliotek eingeordnet und werden dann zum Lösen von konkreten Planungsproblemen benutzt. Nach der Eingabe des Befehls wird die Nummer des Falls erfragt, der abstrahiert werden soll. Die Eingabe muß mit einem „." abgeschlossen werden.

learn_all(<Casefile>,<GAPFile>): Mit diesem Befehl wird eine Menge von konkreten Planungsfällen gelernt, die in dem File <*Casefile*> stehen. Danach wird die so entstandene Menge von abstrakten Plänen in das File <GAPFile> geschrieben. Die abstrakten Pläne werden nach dem eingestellten Modus indexiert. Die Abstraktion wird nach der geladenen, bzw. nach der Theorie im Defaultpfad durchgeführt.

Eine weitere Möglichkeit ein Menge von abstrakten Plänen zu speichern, ist der Befehl

wrskp/0: Nachdem dieser Befehl aufgerufen wird, wird vom Benutzer ein Pfad erfragt, unter dem die abstrakten Pläne, die sich im Moment im Planungssystem befinden, abgespeichert werden sollen. Der eingestellte Indexierungsmodus und eventuell der Branchingfaktor werden zu diesen Plänen hinzugespeichert. Die verwendeten Theorien und die konkreten Planungsfälle werden, genauso wie bei *learn_all(<Casefile>,<GAPFile>)* nicht mit abgespeichert.

Mit den beiden folgenden Befehlen, kann der Indexierungsmodus für die abstrakten Pläne eingestellt werden.

setht/0: Nach der Eingabe diese Befehls kann der Benutzer die Art angeben, wie die abstrakten Pläne gespeichert werden. Der Benutzer kann dann *l.* für das Speichern der Pläne in einer linearen Liste eingeben. Wird ein *t.* angegeben, werden die Pläne in einem Baum abgespeichert.

setsm/0: Mit diesem Befehl wird angegeben, ob der Abstraktionsbaum minimiert werden soll oder ob der ganze Baum zum Problemlösen konstruiert werden soll. Nach der Eingabe von *n.* wird der vollständige Baum erzeugt. Falls jedoch *o.* eingegeben wird, wird nach dem Abstrahieren eines Planungsfalls der Erwartungswert des Baumes minimiert.

Im linearen Modus unter *setht/0* darf der Modus für den Baum mit *setsm/0* nicht auf das Minimieren des Erwartungswerts gesetzt werden. Außerdem dürfen die Indexierungsverfahren nicht vermischt werden. Es müssen erst die abstrakten Pläne gespeichert werden, damit sie und ihre Indexierung nicht verloren gehen, bevor man in einem anderen Indexierungsmodus konkrete Planungsprobleme abstrahiert.
Mit den beiden folgenden Befehlen kann der Branchingfaktor gesetzt werden, bzw. gelöscht werden. Dieser Verzweigungsfaktor wird im Modus Minimierung des Erwartungswerts für das Abschätzen der Problemlösezeit benutzt, falls innerhalb der gesetzten Grenzen mit *setlimit/0* der konkrete Planungsfall mit dem abstrakten Planungsproblem nicht lösbar war. Dieser Branchingfaktor bezieht sich global auf die gesamte konkrete Planungswelt. Dieser Faktor wird, falls er nicht eingegeben wurde, bei Bedarf, vom Benutzer erfragt.

delbf/0: Dieser Befehl löscht den Verzweigungsfaktor, der sich im Planungssystem befindet. Dieser Verzweigungsfaktor wird für den Benutzer noch einmal ausgedruckt, um mit verschiedene Faktoren arbeiten zu können.

setbf/0: Hiermit kann ein neuer Verzweigungsfaktor gesetzt, bzw. errechnet werden. Nach der Eingabe dieses Befehls, kann der Benutzer mit *s.* einen neuen Verzweigungsfaktor direkt eingegeben oder mit *c.* kann ein Faktor berechnet werden. Man gibt danach eine Nummer eines konkreten Planungsfalls ein. Dieser Fall wird dann versucht in Breitensuche zu lösen. Aus dem so entstandenen Suchbaum wird dann der Verzweigungsfaktor ermittelt.

Der noch zu verbleibende Befehl der Abstraktionskomponente ist der Befehl

deb/0: Mit diesem Befehl werden nach jeder Phase der Planabstraktion nach dem PABS-Verfahren die jeweils relevanten Werte aus den Planungswelten ausgegeben. Dies ist besonders wichtig bei der Formulierung einer neuen Planungsdomäne.

Nach der Beschreibung der Abstraktionskomponente folgt nun die Beschreibung der Verfeinerungskomponente. Insgesamt gibt es drei Möglichkeiten, ein neues Planungsproblem oder eine Menge von Planungsproblemen zu lösen. Und zwar

solvec/0: Mit diesem Befehl wird ein einzelner Planungsfall gelöst. Nach der Eingabe wird die Nummer des konkreten Planungsproblems erfragt, das gelöst werden soll. Danach kann der Benutzer wählen, mit welchem Verfahren das Planungsproblem gelöst werden soll. Dazu stehen folgende Verfahren bereit:

```
Solvemode ?
1 breadth first search
2 hierarchical planning
3 refinement planning
```

Bei dem ersten Verfahren wird versucht den konkreten Planungsfall auf konkreter Ebene mit Breitensuche zu lösen. Beim zweiten Verfahren wird mit hierarchischem Planen nach [Knoblock, 1989] probiert, eine Lösung für das konkrete Planungsproblem zu finden. Beim refinement planning werden die abstrakten Pläne nach der eingestellten Indexierungsstruktur nach einem abstrakte Plan durchsucht, der sich auf das konkrete Planungsproblem Re-Instanziieren läßt. Dieser abstrakte Plan wird dann versucht zu verfeinern, um so eine Lösung des Planungsproblems zu erhalten.

plan_all(<Problemfile>,<Resultfile>): Mit diesem Befehl wird eine Menge von Planungsproblemen aus dem File *<Problemfile>* mit den sich im System befindenden abstrakten Plänen gelöst. Die Zeiten, die dazu mit den verschiedenen Verfahren benötigt werden, und die Pläne zur Lösung des Planungsproblems werden dabei in das File *<Resultfile>* geschrieben. Soll dies nicht mit allen Verfahren durchgeführt werden, empfiehlt sich bei *setlimit/0* die jeweilige Menge der zulässigen Suchtiefe auf konkreter Ebene auf *0.* zu setzen.

intlearn(<Problemfile>,<Resultfile>): Mit diesem Befehl wird eine Menge von Planungsproblemen aus dem File *<Problemfile>* mit den sich im System befindenden abstrakten Plänen gelöst. Nach dem Lösen eines Problems wird dann die konkrete Lösung wieder abstrahiert, um alle abstrakten Pläne aus diesem gelösten konkreten Planungsproblem zu finden. Diese Pläne werden dann gemäß der eingestellten Indexierungsart in die Bibliothek mit den abstrakten Plänen einsortiert. Somit werden diese Pläne auch beim Lösen der folgenden Planungsprobleme mitbenutzt. Die Zeiten, die dazu mit den verschiedenen Verfahren benötigt werden, und die Pläne zur Lösung des Planungsproblems werden dabei wieder in das File <*Resultfile*> geschrieben.

Im folgenden wird der Befehl *setlimit/0* beschrieben, der es ermöglicht, den Suchraum des Planers zu begrenzen.

setlimit/0: Nach der Eingabe dieses Befehls werden vom Benutzer sechs verschiedene Werte erfragt, die den Suchraum der Verfeinerungskomponente begrenzen. Diese sind

1. Nodelimit: die maximale Anzahl der Knoten, die insgesamt erzeugt werden dürfen
2. Timelimit: die maximale Zeit, in CPU-Millisekunden, die die Verfeinerungskomponente dazu verwenden darf, um ein Planungsproblem zu lösen
3. Max deep for concrete search: Die maximale Tiefe des Suchbaums bei der Breitensuche
4. Max abstract deep for hierarchical planning: Die maximale Tiefe beim hierarchischen Planen auf der abstrakten Ebene
5. Max concrete deep for hierarchical planning: Die maximale Tiefe beim hierarchischen Planen auf der konkreter Ebene
6. Max concrete deep for refinement planning: Die maximale Tiefe für die Planungsfunktion auf der konkreten Ebene beim Planen mit Verfeinerung

Die anfangs eingestellten Werte werden in dem Prolog-File theories.pl gesetzt und können dort geändert werden.

Mit den zwei folgenden Befehlen kann die Funktionsweise des Planers und der Suchraum beim Lösen von Planungsproblemen modifiziert werden.

setrm/0: Mit diesem Befehl kann das Zurücknehmen von Operatoren begrenzt werden. Ist das Zurücknehmen von konkreten Operatoren über die abstrakten Übergänge hinaus erlaubt, können alle festgelegten Operatoren bis hin zum Startzustand hin zurückgenommen werden. Ist es jedoch nicht erlaubt, kann eine einmal festgelegte konkrete Operatorsequenz, die einen abstrakten Operator auf der konkreten Ebene ersetzt, nicht mehr zurückgenommen werden. So ist das Zurücknehmen eines konkreten Operators nur noch innerhalb eines Übergangs auf abstrakter Ebene möglich.

setss/0: Mit diesem Befehl kann die Menge der Pläne, die bei der Verfeinerung betrachtet werden, in den beiden Indexierungsmodi, die die Baumstruktur verwenden, verändert werden. Ist der Modus auf *first* gestellt, wird nur der jeweilige erste abstrakte Plan in dem Knoten mit den Plänen betrachtet. Dem unterliegt die Annahme, daß die Pläne in allen Knoten genügend klassifiziert sind, sodaß sie innerhalb der Domäne entweder immer alle Pläne eines Knotens ein konkretes Planungsproblem lösen oder aber keiner der Pläne. Wird dieser Modus auf *all* gesetzt, werden die Nachfolgepläne in einem Knoten zur Problemlösung herangezogen, falls der erste Plan des Knotens das Planungsproblem nicht lösen konnte.

Der letzte zu erklärende Befehl ist der Befehl:

log/0: Dieser Befehl ist das Äquivalent zu dem Befehl *deb* in der Abstraktionskomponente. Ist dieser Modus eingeschaltet, werden während der Verfeinerung zusätzliche nützliche Informationen ausgegeben, die dem Benutzer hilfreich sein können herauszufinden, warum sich ein Planungsproblem nicht lösen läßt.

Anhang B

Die Herleitung der Definition des Erwartungswerts der Problemlösezeit

In diesem Kapitel des Anhangs wird die Herleitung der Definition der besten Schätzung des Erwartungswertes der Problemlösezeit eines inneren Knotens einer Abstraktionshierarchie gegeben. Diese Schätzung des Erwartungswerts EW_N wurde in Definition 4.7 für einen Knoten N gegeben durch:

$$\sum_{i=1}^{r} P_i * EW_i + \sum_{i=1}^{r}\left[Tm_i(\mathcal{F}) * \left(1-\sum_{j=1}^{i-1} P_j\right)\right] + Tr_m\left(\widetilde{\mathcal{F}_m} \setminus \bigcup_{i=1}^{r} \widetilde{\mathcal{F}_i}\right) * \left(1-\sum_{i=1}^{r} P_i\right)$$

Um die Herleitung dieser Gleichung nachzuvollziehen, wollen wir zuerst den Fall betrachten, in dem ein innerer Knoten nur einen Nachfolger besitzt. Wir betrachten also eine Abstraktionshierarchie, wie sie in Abbildung B.1 noch einmal dargestellt ist. N_0 ist der Knoten,

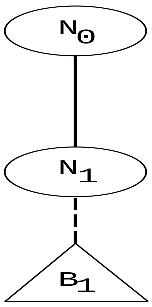

Abbildung B.1: Ein beispielhafter Abstraktionsbaum mit einem Nachfolgerknoten

dessen Erwartungswert wir berechnen wollen. N_1 ist der einzige Nachfolgerknoten diese Knotens und B_1 ist ein beliebiger Restbaum, der sich unterhalb dieses Knotens befindet. Ist dieser Restbaum leer, ist der Knoten N_1 ein Blattknoten. Der Baum unter dem Knoten N_1 kann aber auch ein oder mehrere Nachfolgerknoten von N_1 enthalten. Aus diesem Restbaum B_1 resultiert nun ein Erwartungswert für den Knoten N_1 nach Definition 4.7, den wir mit EW_{N_1} bezeichnen wollen und der als bekannt vorausgesetzt wird[1]. Des weiteren ergibt sich aus den konstruierten Hilfsmengen $\widetilde{\mathcal{F}_{N_0}}, \widetilde{\mathcal{F}_{N_1}}$ nach Definition 4.6 die bedingte Wahrscheinlichkeit, daß ein abstrakter Plan des Knotens N_1 sich auf ein konkretes Planungsproblem anwenden läßt, unter der Bedingung, daß dieses Planungsproblem auch auf einen Plan vom Knoten N_0 anwendbar war:

$$P_1 := P(N_1 \mid N_0) = \frac{|\widetilde{\mathcal{F}_{N_1}}|}{|\widetilde{\mathcal{F}_{N_0}}|}$$

Da $\widetilde{\mathcal{F}_{N_1}} \subseteq \widetilde{\mathcal{F}_{N_0}}$, gilt $P_1 \leq 1$. Die Kosten zum Re-Instanziieren des minimalsten Plans aus N_0 werden bei der Berechnung des Erwartungswerts EW_{N_0} vernachlässigt, da sie, falls der betrachtete Teilbaum aus der Abbildung B.1 zum Problemlösen benutzt wird, immer anfallen, um „in“ den Baum zu gelangen und somit als fixe Kosten auf die Minimierung keinen Einfluß haben. Der Erwartungswert der Problemlösezeit ergibt sich nun aus der Summe:

- Der Re-Instanziierungskosten für die Überprüfung der Anwendbarkeit des minimalen Plans bezüglich $<_{GAP}$ aus Knoten N_1, also $T_{m_{N_1}}$ nach Definition 4.5
- Der bedingten Wahrscheinlichkeit P_1, daß sich ein abstrakter Plan aus dem Knoten N_1 zum Problemlösen verfeinern läßt, multipliziert mit dem Erwartungswert der Problemlösezeit des Knotens EW_{N_1}, also $P_1 * EW_{N_1}$
- Der Wahrscheinlichkeit[2] $(1 - P_1)$, daß sich kein abstrakter Plan aus dem Knoten N_1 zum Problemlösen anwenden läßt, multipliziert mit der Verfeinerungszeit $T_{r_{N_0}}$ des Knotens N_0 nach Definition 4.5, also $(1 - P_1) * T_{r_{N_0}}$

Es ergibt sich also für den Erwartungswert der Problemlösezeit für den Knoten N_0:

$$EW^1_{N_0} := T_{m_{N_1}} + P_1 * EW_{N_1} + (1 - P_1) * T_{r_{N_0}} := EW_{N_0}$$

Somit rechtfertigt sich die Definition der Schätzung des Erwartungswerts der Problemlösezeit für einen inneren Knoten mit einem Nachfolger N_0.

Als nächstes wollen wir den Fall betrachten, daß ein Baum zwei Nachfolgerknoten besitzt. Ein solcher Baum ist noch einmal in der Abbildung B.2 dargestellt. Seien wieder die Erwartungswerte EW_{N_1}, EW_{N_2} der Knoten N_1, N_2 resultierend aus den Unterbäumen B_1, B_2 gegeben. Die bedingten Wahrscheinlichkeiten P_1, P_2 aus den konstruierten Planmengen $\widetilde{\mathcal{F}_0}, \widetilde{\mathcal{F}_1}, \widetilde{\mathcal{F}_2}$ ergeben sich nun aus:

- $P_1 := P(N_1 \mid N_0) = \frac{|\widetilde{\mathcal{F}_{N_1}}|}{|\widetilde{\mathcal{F}_{N_0}}|}$
- $P_2 := P(\neg N_1 \wedge N_2 \mid N_0) = \frac{|\widetilde{\mathcal{F}_{N_2}}|}{|\widetilde{\mathcal{F}_{N_0}}|}$

[1] Gemäß der Definition werden die Erwartungswerte Bottom-Up berechnet und somit existiert dieser Wert EW_{N_1}

[2] Da die Mengen $\widetilde{\mathcal{F}_i}$ nach ihrer Konstruktionsvorschrift disjunkt sind, gelten die Kolmogorov Axiome, die die minimalen Anforderungen an eine Wahrscheinlichkeitsfunktion festlegen, aus denen direkt folgt, daß: $\neg P_1 = (1 - P_1)$ gilt

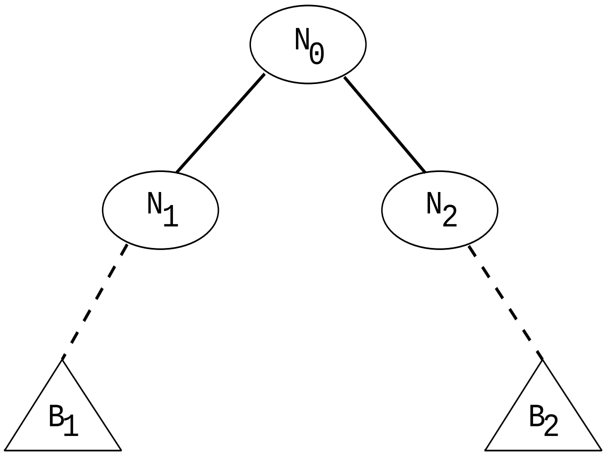

Abbildung B.2: Ein beispielhafter Abstraktionsbaum mit zwei Nachfolgerknoten

Für diese gilt, da $\widetilde{\mathcal{F}_{N_1}} \subseteq \widetilde{\mathcal{F}_{N_0}}$, $\widetilde{\mathcal{F}_{N_2}} \subseteq \widetilde{\mathcal{F}_{N_0}}$ und $\widetilde{\mathcal{F}_{N_1}} \cap \widetilde{\mathcal{F}_{N_2}} = \emptyset$, daß $P_1 + P_2 \leq 1$. Betrachten wir nun wieder die einzelnen Möglichkeiten, mit denen ein konkretes Planungsproblem mit diesem Beispielbaum gelöst werden kann. Die Summe der verschiedenen Zeiten, die in diesen unterschiedlichen Fällen anfallen würden, gewichtet mit der Wahrscheinlichkeit ihres Eintretens, ergeben den Erwartungswert der Problemlösezeit für diesen Beispielbaum. So ergibt sich:

- Die Kosten für die Re-Instanziierung zur Überprüfung der Anwendbarkeit der Pläne des Knotens N_1 und die Kosten, die im Fall der Problemlösung entstehen, durch:

  $T_{m_{N_1}}$: Re-Instanziierung von N_1

  $P_1 * EW_{N_1}$: Erwartungswert des Problemlösen mit Knoten N_1

- Die Kosten für die Re-Instanziierung zur Überprüfung der Anwendbarkeit der Pläne des Knotens N_2 und die Kosten, die im Fall der Problemlösung entstehen, durch:

  $(1 - P_1) * T_{m_{N_2}}$: Re-Instanziierung von N_2

  $P_2 * EW_{N_2}$: Erwartungswert des Problemlösen mit Knoten N_2

- Die Kosten für das Problemlösen mit dem Knoten N_0, falls die abstrakten Pläne der Knoten N_1, N_2 nicht anwendbar waren:

  $(1-(P_1+P_2))*T_{r_{N_0}}$: Erwartungswert der Verfeinerungszeit der Pläne des Knotens N_1.

Durch die Summierung dieser einzelnen Kosten erhält man den Erwartungswert der Problemlösezeit für den Fall, daß ein innerer Knoten zwei Nachfolgerknoten besitzt aus $EW^2_{N_0}$:=:

$$
\begin{array}{ll}
T_{m_{N_1}} & + \; P_1 * EW_{N_1} \; + \\
(1 - P_1) * T_{m_{N_2}} & + \; P_2 * EW_{N_2} \; + \\
 & \qquad\qquad (1 - (P_1 + P_2)) * T_{r_{N_0}}
\end{array}
$$

Durch die Zusammenfassung der drei Spalten der obigen Formel, ergibt sich nun[3] aus $EW^2_{N_0}$:=:

$$\sum_{i=1}^{2} P_i * EW_i + \sum_{i=1}^{2} \left[Tm_i(\mathcal{F}) * \left(1 - \sum_{j=1}^{i-1} P_j\right)\right] + Tr_m \left(\widetilde{\mathcal{F}_m} \setminus \bigcup_{i=1}^{2} \widetilde{\mathcal{F}_i}\right) * \left(1 - \sum_{i=1}^{2} P_i\right)$$

was der Definition 4.7 für den Fall $r = 2$ entspricht.
Nun wollen wir den Fall betrachten, daß ein Baum r Nachfolgerknoten besitzt, wobei r eine natürliche Zahl (> 2) darstellt. Dieser Baum ist noch einmal in einer allgemeinen Form in Abbildung B.3 dargestellt. Die Werte EW_i, P_i für $i = 1, \ldots, r$ und $T_{m_{N_i}}, T_{r_{N_i}}$,

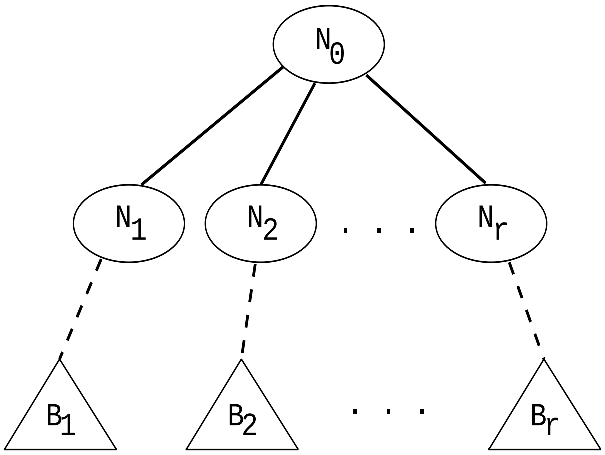

Abbildung B.3: Ein beispielhafter Abstraktionsbaum mit r Nachfolgerknoten

für $i = 0, \ldots, r$ seien wie zuvor definiert, bzw. gegeben. Wir betrachten wieder die sich ergebenden Kosten, die beim Problemlösen mit einzelnen Knoten entstehen, gewichtet mit der Wahrscheinlichkeit ihrer Anwendbarkeit.

- Die Kosten für die Re-Instanziierung zur Überprüfung der Anwendbarkeit der Pläne des Knotens N_i, $i = 1, \ldots, r$, und die Kosten, die im Fall der Problemlösung entstehen, durch:

  $(1 - \sum_{i=1}^{r-1} P_i) * T_{m_{N_i}}$: Re-Instanziierung von N_i

  $P_i * EW_{N_i}$: Erwartungswert des Problemlösen mit Knoten N_i, gewichtet mit der Wahrscheinlichkeit seiner Anwendung P_i

- Die Kosten für das Problemlösen mit dem Knoten N_0, falls die abstrakten Pläne der Knoten N_i, $i = 1, \ldots, r$, nicht anwendbar waren:

  $(1 - \sum_{i=1}^{r} P_i) * T_{r_{N_0}}$: Erwartungswert der Verfeinerungszeit der Pläne des Knotens N_0.

[3]Der zweite Summand resultiert aus der ersten Spalte, der erste Summand aus der zweiten und der dritte Summand geht aus der dritten Spalte hervor

Dann ergibt sich für die Schätzung des Erwartungswerts der Problemlösezeit eines inneren Knotens mit r Nachfolgern, $EW_{N_0}^r :=:$

$$\begin{array}{ll}
T_{m_{N_1}} & + P_1 * EW_{N_1} + \\
(1 - P_1) * T_{m_{N_2}} & + P_2 * EW_{N_2} + \\
(1 - (P_1 + P_2)) * T_{m_{N_3}} & + P_3 * EW_{N_3} + \\
\vdots & \vdots \\
(1 - \sum\limits_{i=1}^{r-1} (P_i * T_{m_{N_r}}) & + P_r * EW_{N_r} + \\
 & \qquad\qquad (1 - \sum\limits_{i=1}^{r} P_i) * T_{r_{N_0}}
\end{array}$$

Durch die Zusammenfassung der einzelnen Spalten wie zuvor ergibt sich nun aus $EW_{N_0}^r :=:$

$$\sum_{i=1}^{r} P_i * EW_i + \sum_{i=1}^{r} \left[Tm_i(\mathcal{F}) * \left(1 - \sum_{j=1}^{i-1} P_j \right) \right] + Tr_m \left(\widetilde{\mathcal{F}_m} \setminus \bigcup_{i=1}^{r} \widetilde{\mathcal{F}_i} \right) * \left(1 - \sum_{i=1}^{r} P_i \right)$$

was der Definition 4.7 der Schätzung des Erwartungswerts der Problemlösezeit eines inneren Knotens einer Abstraktionshierarchie entspricht.

Anhang C

Die Theorien der Planungsdomäne

In diesem Anhang werden die Theorien der konkreten und die der beiden abstrakten Planungswelten dargestellt. Außerdem befinden sich hier die beiden Abstraktionsabbildungen und die Beschreibung eines konkreten Planungsfalls. Für die Erklärung der Semantik der Definitionen dieses Anhangs sei noch einmal auf [Wilke, 1993] verwiesen.

C.1 Die Theorie der konkreten Planungswelt

```
% Planning domain: lpd version 1.0
% Concrete world

essential(grid_xpos(_,_,_),konkret).
essential(grid_ypos(_,_,_),konkret).
essential(mat(_,_,_), konkret).
essential(xpos_max(_),konkret).
essential(ypos_max(_),konkret).
essential(chuck_pos(_),konkret).
essential(covered(_,_),konkret).
essential(cut_direction(_),konkret).
essential(cut_speed(_),konkret).
essential(cut_tool(_),konkret).
essential(control_flag(_),konkret).

% Concrete Theory

regel(1,konkret,
  (not_covered(P) :- covered(Min,Max), P<Min)).
regel(2,konkret,
  (not_covered(P) :- covered(Min,Max), P>Max)).
regel(3,konkret,
  (covered_range(1,1,Max1) :-
     len_wp(Len), ChuckLen is Len / 10,
     grid_xpos(Max,Xstart,Xlen),
```

```
     Xstart > ChuckLen,
     Max1 is Max - 1,
     grid_xpos(Max1,Xstart1,_),
     Xstart1 =< ChuckLen)).
regel(4,konkret,
  (covered_range(r,Min1,Ngrid) :-
     xpos_max(Ngrid),
     len_wp(Len), ChuckLen is Len / 10,
     grid_xpos(Min,Xstart,Xlen),
     Xdiss is Len - (Xstart + Xlen),
     Xdiss > ChuckLen,
     Min1 is Min + 1,
     grid_xpos(Min1,Xstart1,Xlen1),
     Xdiss1 is Len - (Xstart1 + Xlen1),
     Xdiss1 =< ChuckLen)).
regel(5,konkret,
   (len_wp(Len) :- xpos_max(M), grid_xpos(M,XStart,XLen), Len is XStart + XLen)).
regel(6,konkret,
   (block_free(X,Y,r) :-
     k_arround(X,Y,XM1,YM1,XP1,YP1),
mat(X,YP1,n),
mat(XP1,Y,n),
not_covered(X))).
regel(7,konkret,
   (block_free(X,Y,l) :-
     k_arround(X,Y,XM1,YM1,XP1,YP1),
mat(X,YP1,n),
mat(XM1,Y,n),
not_covered(X))).
regel(8,konkret,
   (block_free(X,Y,c) :-
     k_arround(X,Y,XM1,YM1,XP1,YP1),
mat(X,YP1,n),
not_covered(X))).
regel(9,konkret,
   (check_geometry(XPos,YPos,ToolId) :-
k_tool_db(ToolId,Dir,MaxSpeed,MaxDeep,Width),
grid_ypos(YPos,_,Deepness),
grid_xpos(XPos,_,Widthness),
Deepness =< MaxDeep,
Widthness >= Width)).
regel(10,konkret,
    (surface(XPos,YSurface) :-
mat(XPos,YSurface,Mat),k_is_mat(Mat),Y1 is YSurface + 1,
not((mat(XPos,Y1,Mat1),k_is_mat(Mat1))))).
regel(11,konkret,
    (plain(Min,Max,YSurface) :-
```

```
Min > Max)).
regel(12,konkret,
    (plain(Min,Max,YSurface) :-
Min =< Max,
surface(Min,YSurface),
Min1 is Min + 1,
plain(Min1,Max,YSurface))).

% Concrete Operators

operator(chuck(Pos),1,konkret,
   (chuck_pos(none),
    k_chuck_pos_par(Pos),
    covered_range(Pos,Min,Max),
    plain(Min,Max,_)),
   (chuck_pos(none)),
   (chuck_pos(Pos),covered(Min,Max),control_flag(set))).
operator(cut(Xpos,Ypos),4,konkret,
   (cut_direction(D),
    cut_tool(ToolId),
    mat(Xpos,Ypos,r),
    block_free(Xpos,Ypos,D),
    check_geometry(Xpos,Ypos,ToolId)),
   (mat(Xpos,Ypos,r),control_flag(cut),control_flag(set)),
   (mat(Xpos,Ypos,n),control_flag(cut),control_flag(set))).
operator(set_tool(Dir,Speed,ToolId),3,konkret,
   (control_flag(set),
    chuck_pos(ChuckPos),ChuckPos \= none,
    k_set_tool_dir(Dir),
    k_pos_compatible(ChuckPos,Dir),
    k_tool_db(ToolId,Dir,MaxSpeed,MaxDeep,_),
    k_set_tool_speed(Speed), Speed < MaxSpeed),
   (cut_direction(_),cut_speed(_),cut_tool(_),control_flag(set)),
   (cut_direction(Dir), cut_speed(Speed), cut_tool(ToolId))).
operator(unchuck,2,konkret,
   (chuck_pos(P),P \= none,control_flag(cut)),
   (chuck_pos(P),covered(_,_)),
   (chuck_pos(none),control_flag(cut))).
```

C.2 Ein konkreter Planungsfall als Beispiel

Hier ist ein einfacher konkreter Planungsfall in der Case-Beschreibung wiedergegeben. Er besteht aus einer eindeutigen Nummer, dem Start- und Zielzustand und dem Plan zur Lösung des Falls.

```
case(1,
```

```
[chuck_pos(none),xpos_max(18),ypos_max(7),grid_xpos(18, 272.000000, 40.000000),
grid_xpos(17, 247.000000, 25.000000),grid_xpos(16, 228.000000, 19.000000),
grid_xpos(15, 220.000000, 8.000000),grid_xpos(14, 210.000000, 10.000000),
grid_xpos(13, 175.000000, 35.000000),grid_xpos(12, 167.000000, 8.000000),
grid_xpos(11, 147.000000, 20.000000),grid_xpos(10, 136.000000, 11.000000),
grid_xpos(9, 118.000000, 18.000000),grid_xpos(8, 111.000000, 7.000000),
grid_xpos(7, 100.000000, 11.000000),grid_xpos(6, 99.000000, 1.000000),
grid_xpos(5, 85.000000, 14.000000),grid_xpos(4, 57.000000, 28.000000),
grid_xpos(3, 19.000000, 38.000000),grid_xpos(2, 11.000000, 8.000000),
grid_xpos(1, 0, 11.000000),grid_ypos(7, 166.000000, 12.000000),
grid_ypos(6, 134.000000, 32.000000),grid_ypos(5, 111.000000, 23.000000),
grid_ypos(4, 72.000000, 39.000000),grid_ypos(3, 48.000000, 24.000000),
grid_ypos(2, 30.000000, 18.000000),grid_ypos(1, 0, 30.000000),mat(19, 0, n),
mat(18, 0, n),mat(17, 0, n),mat(16, 0, n),mat(15, 0, n),mat(14, 0, n),
mat(13, 0, n),mat(12, 0, n),mat(11, 0, n),mat(10, 0, n),mat(9, 0, n),
mat(8, 0, n),mat(7, 0, n),mat(6, 0, n),mat(5, 0, n),mat(4, 0, n),mat(3, 0, n),
mat(2, 0, n),mat(1, 0, n),mat(0, 0, n),mat(19, 8, n),mat(18, 8, n),
mat(17, 8, n),mat(16, 8, n),mat(15, 8, n),mat(14, 8, n),mat(13, 8, n),
mat(12, 8, n),mat(11, 8, n),mat(10, 8, n),mat(9, 8, n),mat(8, 8, n),
mat(7, 8, n),mat(6, 8, n),mat(5, 8, n),mat(4, 8, n),mat(3, 8, n),mat(2, 8, n),
mat(1, 8, n),mat(0, 8, n),mat(0, 7, n),mat(0, 6, n),mat(0, 5, n),mat(0, 4, n),
mat(0, 3, n),mat(0, 2, n),mat(0, 1, n),mat(19, 7, n),mat(19, 6, n),
mat(19, 5, n),mat(19, 4, n),mat(19, 3, n),mat(19, 2, n),mat(19, 1, n),
mat(18, 7, r),mat(18, 6, w),mat(18, 5, w),mat(18, 4, w),mat(18, 3, w),
mat(18, 2, w),mat(18, 1, w),mat(17, 7, w),mat(17, 6, w),mat(17, 5, w),
mat(17, 4, w),mat(17, 3, w),mat(17, 2, w),mat(17, 1, w),mat(16, 7, w),
mat(16, 6, w),mat(16, 5, w),mat(16, 4, w),mat(16, 3, w),mat(16, 2, w),
mat(16, 1, w),mat(15, 7, w),mat(15, 6, w),mat(15, 5, w),mat(15, 4, w),
mat(15, 3, w),mat(15, 2, w),mat(15, 1, w),mat(14, 7, w),mat(14, 6, w),
mat(14, 5, w),mat(14, 4, w),mat(14, 3, w),mat(14, 2, w),mat(14, 1, w),
mat(13, 7, w),mat(13, 6, w),mat(13, 5, w),mat(13, 4, w),mat(13, 3, w),
mat(13, 2, w),mat(13, 1, w),mat(12, 7, w),mat(12, 6, w),mat(12, 5, w),
mat(12, 4, w),mat(12, 3, w),mat(12, 2, w),mat(12, 1, w),mat(11, 7, w),
mat(11, 6, w),mat(11, 5, w),mat(11, 4, w),mat(11, 3, w),mat(11, 2, w),
mat(11, 1, w),mat(10, 7, w),mat(10, 6, w),mat(10, 5, w),mat(10, 4, w),
mat(10, 3, w),mat(10, 2, w),mat(10, 1, w),mat(9, 7, w),mat(9, 6, w),
mat(9, 5, w),mat(9, 4, w),mat(9, 3, w),mat(9, 2, w),mat(9, 1, w),mat(8, 7, w),
mat(8, 6, w),mat(8, 5, w),mat(8, 4, w),mat(8, 3, w),mat(8, 2, w),mat(8, 1, w),
mat(7, 7, w),mat(7, 6, w),mat(7, 5, w),mat(7, 4, w),mat(7, 3, w),mat(7, 2, w),
mat(7, 1, w),mat(6, 7, w),mat(6, 6, w),mat(6, 5, w),mat(6, 4, w),mat(6, 3, w),
mat(6, 2, w),mat(6, 1, w),mat(5, 7, w),mat(5, 6, w),mat(5, 5, w),mat(5, 4, w),
mat(5, 3, w),mat(5, 2, w),mat(5, 1, w),mat(4, 7, w),mat(4, 6, w),mat(4, 5, w),
mat(4, 4, w),mat(4, 3, w),mat(4, 2, w),mat(4, 1, w),mat(3, 7, w),mat(3, 6, w),
mat(3, 5, w),mat(3, 4, w),mat(3, 3, w),mat(3, 2, w),mat(3, 1, w),mat(2, 7, w),
mat(2, 6, w),mat(2, 5, w),mat(2, 4, w),mat(2, 3, w),mat(2, 2, w),mat(2, 1, w),
mat(1, 7, w),mat(1, 6, w),mat(1, 5, w),mat(1, 4, w),mat(1, 3, w),mat(1, 2, w),
mat(1, 1, w)], [xpos_max(18),ypos_max(7),grid_xpos(18, 272.000000, 40.000000),
```

```
grid_xpos(17, 247.000000, 25.000000),grid_xpos(16, 228.000000, 19.000000),
grid_xpos(15, 220.000000, 8.000000),grid_xpos(14, 210.000000, 10.000000),
grid_xpos(13, 175.000000, 35.000000),grid_xpos(12, 167.000000, 8.000000),
grid_xpos(11, 147.000000, 20.000000),grid_xpos(10, 136.000000, 11.000000),
grid_xpos(9, 118.000000, 18.000000),grid_xpos(8, 111.000000, 7.000000),
grid_xpos(7, 100.000000, 11.000000),grid_xpos(6, 99.000000, 1.000000),
grid_xpos(5, 85.000000, 14.000000),grid_xpos(4, 57.000000, 28.000000),
grid_xpos(3, 19.000000, 38.000000),grid_xpos(2, 11.000000, 8.000000),
grid_xpos(1, 0, 11.000000),grid_ypos(7, 166.000000, 12.000000),
grid_ypos(6, 134.000000, 32.000000),grid_ypos(5, 111.000000, 23.000000),
grid_ypos(4, 72.000000, 39.000000),grid_ypos(3, 48.000000, 24.000000),
grid_ypos(2, 30.000000, 18.000000),grid_ypos(1, 0, 30.000000),mat(19, 0, n),
mat(18, 0, n),mat(17, 0, n),mat(16, 0, n),mat(15, 0, n),mat(14, 0, n),
mat(13, 0, n),mat(12, 0, n),mat(11, 0, n),mat(10, 0, n),mat(9, 0, n),
mat(8, 0, n),mat(7, 0, n),mat(6, 0, n),mat(5, 0, n),mat(4, 0, n),mat(3, 0, n),
mat(2, 0, n),mat(1, 0, n),mat(0, 0, n),mat(19, 8, n),mat(18, 8, n),
mat(17, 8, n),mat(16, 8, n),mat(15, 8, n),mat(14, 8, n),mat(13, 8, n),
mat(12, 8, n),mat(11, 8, n),mat(10, 8, n),mat(9, 8, n),mat(8, 8, n),
mat(7, 8, n),mat(6, 8, n),mat(5, 8, n),mat(4, 8, n),mat(3, 8, n),mat(2, 8, n),
mat(1, 8, n),mat(0, 8, n),mat(0, 7, n),mat(0, 6, n),mat(0, 5, n),mat(0, 4, n),
mat(0, 3, n),mat(0, 2, n),mat(0, 1, n),mat(19, 7, n),mat(19, 6, n),
mat(19, 5, n),mat(19, 4, n),mat(19, 3, n),mat(19, 2, n),mat(19, 1, n),
mat(18, 6, w),mat(18, 5, w),mat(18, 4, w),mat(18, 3, w),mat(18, 2, w),
mat(18, 1, w),mat(17, 7, w),mat(17, 6, w),mat(17, 5, w),mat(17, 4, w),
mat(17, 3, w),mat(17, 2, w),mat(17, 1, w),mat(16, 7, w),mat(16, 6, w),
mat(16, 5, w),mat(16, 4, w),mat(16, 3, w),mat(16, 2, w),mat(16, 1, w),
mat(15, 7, w),mat(15, 6, w),mat(15, 5, w),mat(15, 4, w),mat(15, 3, w),
mat(15, 2, w),mat(15, 1, w),mat(14, 7, w),mat(14, 6, w),mat(14, 5, w),
mat(14, 4, w),mat(14, 3, w),mat(14, 2, w),mat(14, 1, w),mat(13, 7, w),
mat(13, 6, w),mat(13, 5, w),mat(13, 4, w),mat(13, 3, w),mat(13, 2, w),
mat(13, 1, w),mat(12, 7, w),mat(12, 6, w),mat(12, 5, w),mat(12, 4, w),
mat(12, 3, w),mat(12, 2, w),mat(12, 1, w),mat(11, 7, w),mat(11, 6, w),
mat(11, 5, w),mat(11, 4, w),mat(11, 3, w),mat(11, 2, w),mat(11, 1, w),
mat(10, 7, w),mat(10, 6, w),mat(10, 5, w),mat(10, 4, w),mat(10, 3, w),
mat(10, 2, w),mat(10, 1, w),mat(9, 7, w),mat(9, 6, w),mat(9, 5, w),
mat(9, 4, w),mat(9, 3, w),mat(9, 2, w),mat(9, 1, w),mat(8, 7, w),mat(8, 6, w),
mat(8, 5, w),mat(8, 4, w),mat(8, 3, w),mat(8, 2, w),mat(8, 1, w),mat(7, 7, w),
mat(7, 6, w),mat(7, 5, w),mat(7, 4, w),mat(7, 3, w),mat(7, 2, w),mat(7, 1, w),
mat(6, 7, w),mat(6, 6, w),mat(6, 5, w),mat(6, 4, w),mat(6, 3, w),mat(6, 2, w),
mat(6, 1, w),mat(5, 7, w),mat(5, 6, w),mat(5, 5, w),mat(5, 4, w),mat(5, 3, w),
mat(5, 2, w),mat(5, 1, w),mat(4, 7, w),mat(4, 6, w),mat(4, 5, w),mat(4, 4, w),
mat(4, 3, w),mat(4, 2, w),mat(4, 1, w),mat(3, 7, w),mat(3, 6, w),mat(3, 5, w),
mat(3, 4, w),mat(3, 3, w),mat(3, 2, w),mat(3, 1, w),mat(2, 7, w),mat(2, 6, w),
mat(2, 5, w),mat(2, 4, w),mat(2, 3, w),mat(2, 2, w),mat(2, 1, w),mat(1, 7, w),
mat(1, 6, w),mat(1, 5, w),mat(1, 4, w),mat(1, 3, w),mat(1, 2, w),mat(1, 1, w),
cut_direction(r),cut_speed(10),cut_tool(t2),mat(18, 7, n),control_flag(cut),
control_flag(set),chuck_pos(none),control_flag(cut)],
```

```
[chuck(l),set_tool(r, 10, t2),cut(18, 7),unchuck]).
```

C.3 Die abstrakte Planungswelt der „echten“ Abstraktionstheorie

Dies ist die abstrakte Planungswelt die zur „echten“ Abstraktionstheorie gehört.

```
% Real Abstract Theory

essential(abs_area_state(_,_),abstrakt).
essential(abs_chuck_pos(_),abstrakt).
essential(abs_small_parts(_),abstrakt).
essential(abs_area_m_accessable(_),abstrakt).
essential(abs_chuckable_wp(_),abstrakt).

operational(abs_avail_pos(_),abstrakt).
operational(_\=_,abstrakt).
operational(chuck_comp(_,_),abstrakt).
operational(outer_area(_),abstrakt).
operational(abs_oposite(_,_),abstrakt).

operator(make_ready(Area),1,abstrakt,
   ( abs_area_state(Area,todo),
     abs_chuck_pos(P),chuck_comp(P,Area),
     abs_access(Area)),
   (abs_area_state(Area,todo)),
   (abs_area_state(Area,ready))).
operator_protect(make_ready(_),1,abstrakt,
     (abs_chuck_pos(_))).
operator(make_raw(Area),2,abstrakt,
   ( abs_area_state(Area,todo),
     abs_chuck_pos(P),chuck_comp(P,Area),
     abs_access(Area)),
    (abs_area_state(Area,todo)),
   (abs_area_state(Area,raw))).
operator_protect(make_raw(_),2,abstrakt,
     (abs_chuck_pos(_))).
operator(make_fine(Area),3,abstrakt,
   (abs_area_state(Area,raw),abs_small_parts(Area),
     abs_chuck_pos(P),chuck_comp(P,Area)),
   (abs_area_state(Area,raw),abs_small_parts(Area)),
   (abs_area_state(Area,ready))).
operator_protect(make_fine(_),3,abstrakt,
     (abs_chuck_pos(_))).
operator(abs_chuck(Pos),4,abstrakt,
```

```
   (abs_avail_pos(Pos),abs_chuck_pos(Old),Old\=Pos,
    abs_chuckable(Pos)),
   (abs_chuck_pos(Old)),
   (abs_chuck_pos(Pos))).
operator_protect(abs_chuck(_),4,abstrakt,
     (abs_area_state(_,_))).

regel(100,abstrakt,
  (abs_avail_pos(l) :- true)).
regel(101,abstrakt,
  (abs_avail_pos(r) :- true)).
regel(102,abstrakt,
  (abs_avail_pos(none) :- true)).
regel(103,abstrakt,
(chuck_comp(l,r) :- true)).
regel(104,abstrakt,
(chuck_comp(r,l) :- true)).
regel(105,abstrakt,
(chuck_comp(l,m) :- true)).
regel(106,abstrakt,
(chuck_comp(r,m) :- true)).
regel(107,abstrakt,
(abs_access(m) :-
abs_chuck_pos(P),
abs_area_state(A,ready),
abs_oposite(A,P),
abs_area_m_accessable(A))).
regel(108,abstrakt,
(abs_access(m) :-
abs_chuck_pos(P),
abs_area_state(A,raw),
abs_oposite(A,P),
abs_area_m_accessable(A))).
regel(109,abstrakt,
(abs_access(A) :-
outer_area(A))).
regel(110,abstrakt,
(abs_oposite(l,r) :- true)).
regel(111,abstrakt,
(abs_oposite(r,l) :- true)).
regel(112,abstrakt,
(outer_area(l) :- true)).
regel(113,abstrakt,
(outer_area(r) :- true)).
regel(120,abstrakt,
(abs_chuckable(none) :- true)).
regel(121,abstrakt,
```

```
(abs_chuckable(Area) :-
Area \= none,
abs_area_state(Area,todo))).
regel(122,abstrakt,
(abs_chuckable(Area) :-
Area \= none,
abs_area_state(Area,raw))).
regel(123,abstrakt,
(abs_chuckable(Area) :-
Area \= none,
abs_area_state(Area,ready),
abs_chuckable_wp(Area))).
```

C.4 Die abstrakte Planungswelt der „echten“ Abstraktionstheorie mit der Identität

Dies ist die abstrakte Planungswelt, die für Abstraktionsabbildung definiert wurde, die die Identität realisiert zusammen mit der „echten“ Abstraktionstheorie.

```
% Real Abstract Theory with Identity

essential(a_grid_xpos(_,_,_),abstrakt).
essential(a_grid_ypos(_,_,_),abstrakt).
essential(a_mat(_,_,_), abstrakt).
essential(a_xpos_max(_),abstrakt).
essential(a_ypos_max(_),abstrakt).
essential(a_chuck_pos(_),abstrakt).
essential(a_covered(_,_),abstrakt).
essential(a_cut_direction(_),abstrakt).
essential(a_cut_speed(_),abstrakt).
essential(a_cut_tool(_),abstrakt).
essential(abs_area_state(_,_),abstrakt).
essential(abs_chuck_pos(_),abstrakt).
essential(abs_small_parts(_),abstrakt).
essential(abs_area_m_accessable(_),abstrakt).
essential(abs_chuckable_wp(_),abstrakt).

operational(_\==_,abstrakt).
operational(_=_,abstrakt).
operational(_<_,abstrakt).
operational(_>_,abstrakt).
operational(_=<_,abstrakt).
operational(_>=_,abstrakt).
operational(reverse(_,_),abstrakt).
operational(nonmember(_,_),abstrakt).
```

```
operational(_ is _,abstrakt).
operational(k_chuck_pos_par(_),abstrakt).
operational(k_set_tool_dir(_),abstrakt).
operational(k_set_tool_speed(_),abstrakt).
operational(k_pos_compatible(_,_),abstrakt).
operational(k_arround(_,_,_,_,_,_),abstrakt).
operational(k_tool_db(_,_,_,_,_),abstrakt).
operational(k_is_mat(_),abstrakt).
operational(g_small_part(_),abstrakt).
operational(plain(_,_,_),abstrakt).
operational(abs_avail_pos(_),abstrakt).
operational(_\=_,abstrakt).
operational(chuck_comp(_,_),abstrakt).
operational(outer_area(_),abstrakt).
operational(abs_oposite(_,_),abstrakt).

% Concrete Operators

operator(a_chuck(Pos),1,abstrakt,
   (a_chuck_pos(none),
    k_chuck_pos_par(Pos),
    covered_range(Pos,Min,Max)
%    plain(Min,Max,_)),
   (a_chuck_pos(none)),
   (a_chuck_pos(Pos),a_covered(Min,Max))).
operator_protect(a_chuck(_),1,abstrakt,
     (a_mat(_,_,_),a_cut_direction(_),a_cut_speed(_),a_cut_tool(_))).

operator(a_cut(Xpos,Ypos),2,abstrakt,
   (a_cut_direction(D),
    a_cut_tool(ToolId),
    a_mat(Xpos,Ypos,r),
    block_free(Xpos,Ypos,D),
    check_geometry(Xpos,Ypos,ToolId)),
   (a_mat(Xpos,Ypos,r)),
   (a_mat(Xpos,Ypos,n))).
operator_protect(a_cut(_,_),2,abstrakt,
     (a_chuck_pos(_),a_cut_direction(_),a_cut_speed(_),a_cut_tool(_))).

operator(a_set_tool(Dir,Speed,ToolId),3,abstrakt,
   (a_chuck_pos(ChuckPos),ChuckPos \= none,
    k_set_tool_dir(Dir),
    k_pos_compatible(ChuckPos,Dir),
    k_tool_db(ToolId,Dir,MaxSpeed,MaxDeep,_),
    k_set_tool_speed(Speed), Speed < MaxSpeed,
    a_cut_direction(OldD),
    a_cut_speed(OldS),
```

```
    a_cut_tool(OldT)),
   (a_cut_direction(OldD),a_cut_speed(OldS),a_cut_tool(OldT)),
   (a_cut_direction(Dir), a_cut_speed(Speed), a_cut_tool(ToolId))).
operator_protect(a_set_tool(_,_,_),3,abstrakt,
     (a_mat(_,_,_),a_chuck_pos(_))).

operator(a_unchuck,4,abstrakt,
   (a_chuck_pos(P),P \= none),
   (a_chuck_pos(P),a_covered(_,_)),
   (a_chuck_pos(none))).
operator_protect(a_unchuck,4,abstrakt,
     (a_mat(_,_,_),a_cut_direction(_),a_cut_speed(_),a_cut_tool(_))).

operator(make_ready(Area),5,abstrakt,
   ( abs_area_state(Area,todo),
     abs_chuck_pos(P),chuck_comp(P,Area),
     abs_access(Area)),
   (abs_area_state(Area,todo)),
   (abs_area_state(Area,ready))).
operator_protect(make_ready(_),5,abstrakt,
     (abs_chuck_pos(_))).

operator(make_raw(Area),6,abstrakt,
   ( abs_area_state(Area,todo),
     abs_chuck_pos(P),chuck_comp(P,Area),
     abs_access(Area)),
   (abs_area_state(Area,todo)),
   (abs_area_state(Area,raw))).
operator_protect(make_raw(_),6,abstrakt,
     (abs_chuck_pos(_))).

operator(make_fine(Area),7,abstrakt,
   (abs_area_state(Area,raw),abs_small_parts(Area),
     abs_chuck_pos(P),chuck_comp(P,Area)),
   (abs_area_state(Area,raw),abs_small_parts(Area)),
   (abs_area_state(Area,ready))).
operator_protect(make_fine(_),7,abstrakt,
     (abs_chuck_pos(_))).

operator(abs_chuck(Pos),8,abstrakt,
   (abs_avail_pos(Pos),abs_chuck_pos(Old),Old\=Pos,
    abs_chuckable(Pos)),
   (abs_chuck_pos(Old)),
   (abs_chuck_pos(Pos))).
operator_protect(abs_chuck(_),8,abstrakt,
     (abs_area_state(_,_))).
```

```
regel(100,abstrakt,
  (abs_avail_pos(l) :- true)).
regel(101,abstrakt,
  (abs_avail_pos(r) :- true)).
regel(102,abstrakt,
  (abs_avail_pos(none) :- true)).
regel(103,abstrakt,
(chuck_comp(l,r) :- true)).
regel(104,abstrakt,
(chuck_comp(r,l) :- true)).
regel(105,abstrakt,
(chuck_comp(l,m) :- true)).
regel(106,abstrakt,
(chuck_comp(r,m) :- true)).
regel(107,abstrakt,
(abs_access(m) :-
abs_chuck_pos(P),
abs_area_state(A,ready),
abs_oposite(A,P),
abs_area_m_accessable(A))).
regel(108,abstrakt,
(abs_access(m) :-
abs_chuck_pos(P),
abs_area_state(A,raw),
abs_oposite(A,P),
abs_area_m_accessable(A))).
regel(109,abstrakt,
(abs_access(A) :-
outer_area(A))).
regel(110,abstrakt,
(abs_oposite(l,r) :- true)).
regel(111,abstrakt,
(abs_oposite(r,l) :- true)).
regel(112,abstrakt,
(outer_area(l) :- true)).
regel(113,abstrakt,
(outer_area(r) :- true)).
regel(120,abstrakt,
(abs_chuckable(none) :- true)).
regel(121,abstrakt,
(abs_chuckable(Area) :-
Area \= none,
abs_area_state(Area,todo))).
regel(122,abstrakt,
(abs_chuckable(Area) :-
Area \= none,
abs_area_state(Area,raw))).
```

```
regel(123,abstrakt,
(abs_chuckable(Area) :-
Area \= none,
abs_area_state(Area,ready),
abs_chuckable_wp(Area))).
regel(31,abstrakt,
  (not_covered(P) :- a_covered(Min,Max), P<Min)).
regel(32,abstrakt,
  (not_covered(P) :- a_covered(Min,Max), P>Max)).
regel(33,abstrakt,
  (covered_range(l,1,Max1) :-
     len_wp(Len), ChuckLen is Len / 10,
     a_grid_xpos(Max,Xstart,Xlen),
     Xstart > ChuckLen,
     Max1 is Max - 1,
     a_grid_xpos(Max1,Xstart1,_),
     Xstart1 =< ChuckLen)).
regel(34,abstrakt,
  (covered_range(r,Min1,Ngrid) :-
     a_xpos_max(Ngrid),
     len_wp(Len), ChuckLen is Len / 10,
     a_grid_xpos(Min,Xstart,Xlen),
     Xdiss is Len - (Xstart + Xlen),
     Xdiss > ChuckLen,
     Min1 is Min + 1,
     a_grid_xpos(Min1,Xstart1,Xlen1),
     Xdiss1 is Len - (Xstart1 + Xlen1),
     Xdiss1 =< ChuckLen)).
regel(35,abstrakt,
   (len_wp(Len) :- a_xpos_max(M), a_grid_xpos(M,XStart,XLen),
    Len is XStart + XLen)).
regel(36,abstrakt,
   (block_free(X,Y,r) :-
     k_arround(X,Y,XM1,YM1,XP1,YP1),
a_mat(X,YP1,n),
a_mat(XP1,Y,n),
not_covered(X))).
regel(37,abstrakt,
   (block_free(X,Y,l) :-
     k_arround(X,Y,XM1,YM1,XP1,YP1),
a_mat(X,YP1,n),
a_mat(XM1,Y,n),
not_covered(X))).
regel(38,abstrakt,
   (block_free(X,Y,c) :-
     k_arround(X,Y,XM1,YM1,XP1,YP1),
a_mat(X,YP1,n),
```

```
not_covered(X))).
regel(39,abstrakt,
   (check_geometry(XPos,YPos,ToolId) :-
k_tool_db(ToolId,Dir,MaxSpeed,MaxDeep,Width),
a_grid_ypos(YPos,_,Deepness),
a_grid_xpos(XPos,_,Widthness),
Deepness =< MaxDeep,
Widthness >= Width)).
regel(310,abstrakt,
    (surface(XPos,YSurface) :-
a_mat(XPos,YSurface,Mat),k_is_mat(Mat),Y1 is YSurface + 1,
not((a_mat(XPos,Y1,Mat1),k_is_mat(Mat1)))
)).
regel(311,abstrakt,
    (plain(Min,Max,YSurface) :-
Min > Max)).
regel(312,abstrakt,
    (plain(Min,Max,YSurface) :-
Min =< Max,
surface(Min,YSurface),
Min1 is Min + 1,
plain(Min1,Max,YSurface))).
```

C.5 Die Abstraktion, die die „echte“ Abstraktion realisiert

Dies ist die Abstraktionstheorie, die nur die „echte“ Abstraktion durchführt.

```
% Generic Theory

regel(1,generisch,
   (abs_area_state(l,State) :-
covered_range(l,Min,Max),
state_of(Min,Max,State))).
regel(2,generisch,
   (abs_area_state(r,State) :-
covered_range(r,Min,Max),
state_of(Min,Max,State))).
regel(3,generisch,
   (abs_area_state(m,State) :-
covered_range(l,Minl,Maxl),
covered_range(r,Minr,Maxr),
Min is Maxl + 1,
Max is Minr - 1,
state_of(Min,Max,State))).
```

```
regel(4,generisch,
   (state_of(Min,Max,todo) :-
mat(Xpos,_,r),Xpos =< Max,Xpos >= Min,
grid_xpos(Xpos,_,Size), not((g_small_part(Size))))).
regel(5,generisch,
   (state_of(Min,Max,raw) :-
not((state_of(Min,Max,todo))),
mat(Xpos,_,r),Xpos =< Max,Xpos >= Min,
grid_xpos(Xpos,_,Size), g_small_part(Size))).
regel(6,generisch,
   (state_of(Min,Max,ready) :-
not((state_of(Min,Max,todo))),
not((state_of(Min,Max,raw))))).
regel(7,generisch,
    (abs_chuck_pos(P) :- chuck_pos(P))).
regel(8,generisch,
    (abs_small_parts(l) :-
covered_range(l,Min,Max),
mat(XPos,_,r), XPos =< Max,XPos >= Min,grid_xpos(XPos,_,Size),
g_small_part(Size))).
regel(9,generisch,
    (abs_small_parts(r) :-
covered_range(r,Min,Max),
mat(XPos,_,r), XPos =< Max,XPos >= Min,grid_xpos(XPos,_,Size),
g_small_part(Size))).
regel(10,generisch,
    (abs_small_parts(m) :-
covered_range(l,Minl,Maxl),
covered_range(r,Minr,Maxr),
Min is Maxl + 1,
Max is Minr - 1,
mat(XPos,_,r), XPos =< Max,XPos >= Min,grid_xpos(XPos,_,Size),
g_small_part(Size))).
regel(11,generisch,
    (lowest_raw(X,Yret) :-
mat(X,Yret,MAT),
MAT \= w,
not(lowerst_raw1(X,Yret)))).
regel(12,generisch,
    (lowest_raw1(X,Y) :-
mat(X,Y1,MAT),
MAT \= w,
Y1 < Y)).
regel(13,generisch,
    (abs_area_m_accessable(l) :-
covered_range(l,_,LEnd),
MAnf is LEnd + 1,
```

```
lowest_raw(MAnf,YLow),
mat(LEnd,YLow,MAT),
MAT \= w)).
regel(14,generisch,
    (abs_area_m_accessable(r) :-
covered_range(r,RAnf,_),
MEnd is RAnf - 1 ,
lowest_raw(MEnd,YLow),
mat(RAnf,YLow,MAT),
MAT \= w)).
regel(15,generisch,
    (wp_surface(XPos,YSurface) :-
mat(XPos,YSurface,w),Y1 is YSurface + 1,
not(mat(XPos,Y1,w)))).
regel(16,generisch,
    (wp_plain(Min,Max,YSurface) :-
Min > Max)).
regel(17,generisch,
    (wp_plain(Min,Max,YSurface) :-
Min =< Max,
wp_surface(Min,YSurface),
Min1 is Min + 1,
wp_plain(Min1,Max,YSurface))).
regel(18,generisch,
   (abs_chuckable_wp(r) :-
covered_range(r,Min,Max),
wp_plain(Min,Max,_))).
regel(19,generisch,
   (abs_chuckable_wp(l) :-
covered_range(l,Min,Max),
wp_plain(Min,Max,_))).
```

C.6 Die Abstraktionsabbildung die die identischen Abbildung und die „echte“ Abstraktion realisiert

Dies ist die Abstraktionsabbildung, die die Identität und die „echte“ Abstraktion realisiert.

```
% Generic Theory

regel(41,generisch,
   (abs_area_state(l,State) :-
covered_range(l,Min,Max),
state_of(Min,Max,State))).
regel(42,generisch,
```

```
   (abs_area_state(r,State) :-
covered_range(r,Min,Max),
state_of(Min,Max,State))).
regel(43,generisch,
   (abs_area_state(m,State) :-
covered_range(l,Minl,Maxl),
covered_range(r,Minr,Maxr),
Min is Maxl + 1,
Max is Minr - 1,
state_of(Min,Max,State))).
regel(44,generisch,
   (state_of(Min,Max,todo) :-
mat(Xpos,_,r),Xpos =< Max,Xpos >= Min,
grid_xpos(Xpos,_,Size), not((g_small_part(Size))))).
regel(45,generisch,
   (state_of(Min,Max,raw) :-
not((state_of(Min,Max,todo))),
mat(Xpos,_,r),Xpos =< Max,Xpos >= Min,
grid_xpos(Xpos,_,Size), g_small_part(Size))).
regel(46,generisch,
   (state_of(Min,Max,ready) :-
not((state_of(Min,Max,todo))),
not((state_of(Min,Max,raw))))).
regel(47,generisch,
    (abs_chuck_pos(P) :- chuck_pos(P))).
regel(48,generisch,
    (abs_small_parts(l) :-
covered_range(l,Min,Max),
mat(XPos,_,r), XPos =< Max,XPos >= Min,grid_xpos(XPos,_,Size),
g_small_part(Size))).
regel(49,generisch,
    (abs_small_parts(r) :-
covered_range(r,Min,Max),
mat(XPos,_,r), XPos =< Max,XPos >= Min,grid_xpos(XPos,_,Size),
g_small_part(Size))).
regel(410,generisch,
    (abs_small_parts(m) :-
covered_range(l,Minl,Maxl),
covered_range(r,Minr,Maxr),
Min is Maxl + 1,
Max is Minr - 1,
mat(XPos,_,r), XPos =< Max,XPos >= Min,grid_xpos(XPos,_,Size),
g_small_part(Size))).
regel(411,generisch,
    (lowest_raw(X,Yret) :-
mat(X,Yret,MAT),
MAT \= w,
```

```
not(lowerst_raw1(X,Yret)))).
regel(412,generisch,
    (lowest_raw1(X,Y) :-
mat(X,Y1,MAT),
MAT \= w,
Y1 < Y)).
regel(413,generisch,
    (abs_area_m_accessable(l) :-
covered_range(l,_,LEnd),
MAnf is LEnd + 1,
lowest_raw(MAnf,YLow),
mat(LEnd,YLow,MAT),
MAT \= w)).
regel(414,generisch,
    (abs_area_m_accessable(r) :-
covered_range(r,RAnf,_),
MEnd is RAnf - 1 ,
lowest_raw(MEnd,YLow),
mat(RAnf,YLow,MAT),
MAT \= w)).
regel(415,generisch,
    (wp_surface(XPos,YSurface) :-
mat(XPos,YSurface,w),Y1 is YSurface + 1,
not(mat(XPos,Y1,w)))).
regel(416,generisch,
    (wp_plain(Min,Max,YSurface) :-
Min > Max)).
regel(417,generisch,
    (wp_plain(Min,Max,YSurface) :-
Min =< Max,
wp_surface(Min,YSurface),
Min1 is Min + 1,
wp_plain(Min1,Max,YSurface))).
regel(418,generisch,
   (abs_chuckable_wp(r) :-
covered_range(r,Min,Max),
wp_plain(Min,Max,_))).
regel(419,generisch,
   (abs_chuckable_wp(l) :-
covered_range(l,Min,Max),
wp_plain(Min,Max,_))).
regel(21,generisch,
    (a_grid_xpos(A,B,C) :- grid_xpos(A,B,C))).
regel(22,generisch,
    (a_grid_ypos(A,B,C) :- grid_ypos(A,B,C))).
regel(23,generisch,
    (a_mat(A,B,C) :- mat(A,B,C))).
```

```
regel(24,generisch,
    (a_xpos_max(A) :- xpos_max(A))).
regel(25,generisch,
    (a_ypos_max(A) :- ypos_max(A))).
regel(26,generisch,
    (a_chuck_pos(A) :- chuck_pos(A))).
regel(27,generisch,
    (a_covered(A,B) :- covered(A,B))).
regel(28,generisch,
    (a_cut_direction(A) :- cut_direction(A))).
regel(81,generisch,
    (a_cut_direction(none) :- not(cut_direction(_)))).
regel(29,generisch,
    (a_cut_speed(A) :- cut_speed(A))).
regel(91,generisch,
    (a_cut_speed(none) :- not(cut_speed(_)))).
regel(110,generisch,
    (a_cut_tool(A) :- cut_tool(A))).
regel(101,generisch,
    (a_cut_tool(none) :- not(cut_tool(_)))).
```

Index

Literaturverzeichnis

[Bergmann and W.Wilke, 1993] R. Bergmann and W.Wilke. Inkrementelles Lernen von Abstraktionshierarchien aus maschinell abstrahierten Plänen. GI-Fachgruppentreffen: Maschinelles Lernen. (to be published), 1993.

[Bergmann *et al.*, 1993] R. Bergmann, Pews G., and W. Wilke. Explanation based similarity: A unifing apporach for integrating domain knowledge into case-based reasoning for diagnose and planning tasks. In *Selected Papers of the First European Workshop on Case-Based Reasoning EWCBR-93*. Springer-Verlag (in press), 1993.

[Bergmann, 1990] R. Bergmann. Generierung von Skelettplänen als Problem der Wissensakquisition. Master's thesis, Universität Kaiserslautern, Erwin-Schrödinger-Straße, Postfach 3049, W-67653 Kaiserslautern, Germany, 1990.

[Bergmann, 1992a] R. Bergmann. Knowledge acquisition by generating skeletal plans. In F. Schmalhofer, G. Strube, and Th. Wetter, editors, *Contemporary Knowledge Engineering and Cognition*, pages 125–133, Heidelberg, 1992. Springer.

[Bergmann, 1992b] R. Bergmann. Learning plan abstractions. In H.J. Ohlbach, editor, *GWAI-92 16th German Workshop on Artificial Intelligence*, volume 671 of *Springer Lecture Notes on AI*, pages 187–198, 1992.

[Bergmann, 1993a] R. Bergmann. Integrating abstraction, explanation-based learning from multiple examples and hierarchical clustering with a performance component for planning. In Enric Plaza, editor, *Proceedings of the ECML-93 Workshop on Integrated Learning Architectures (ILA-93)*, Vienna, Austria, 1993.

[Bergmann, 1993b] R. Bergmann. Learning Hierarchically Clustered Shared Plan Abstractions as Problem solving Knowledge with High utility for planning. In A. Horz, editor, *Proceedings of GI-Workshop on 'Planung und Konfigurierung' PuK*, number 723 in Arbeitspapiere der GMD, pages 97–108, Bonn, 1993. GMD.

[Charniak and McDermott, 1985] E. Charniak and D. McDermott. *Introduction to artificial intelligence*. Addison-Wesley Publishing, 1985.

[DeJong and Mooney, 1986] G. DeJong and R. Mooney. Explanation-based learning: An alternative view. *Machine Learning*, 1(2):145–176, 1986.

[Ellman, 1989] T. Ellman. Explanation-based learning: A survey of programs and perspectives. *ACM Computing Surveys*, 21(2):163–222, 1989.

[Fikes and Nilsson, 1971] R. E. Fikes and N. J. Nilsson. Strips: A new apporach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[Fikes *et al.*, 1972] R. E. Fikes, P. E. Hart, and N. J. Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence*, 3:251–288, 1972.

[Fisher, 1987] D. H. Fisher. Knowledge aquisition via incremental conceptual clustering. *Machine Learning*, 2(2):139–172, 1987.

[Friedland and Iwasaki, 1985] P. E. Friedland and Y. Iwasaki. The concept and implementation of skeletal plans. *Journal of Automated Reasoning*, pages 161–208, 1985.

[Ganter, 1987] Bernhard Ganter. Algorithmen zur formalen begriffsanalyse. In B. Ganther, R. Wille, and K. E. Wolf, editors, *Beiträge zur Begriffsanalyse*, pages 241 – 254. B.I. - Wissenschaftsverlag, 1987.

[Gennari *et al.*, 1989] J. H. Gennari, P. Langley, and D. Fisher. Models of incremental concept formation. *Artificial Intelligence*, 40:11–61, 1989.

[Giordana *et al.*, 1991] A. Giordana, D. Roverso, and L. Saitta. Abstracting background knowledge for concept learning. In Y. Kodratoff, editor, *Lecture Notes in Artificial Intelligence: Machine Learning-EWSL-91*, volume 482, pages 1–13, Berlin, 1991. Springer.

[Görtz, G., 1993] Görtz, G. *Einführung in die künstliche Intelligenz.* Addision-Wesley, 1 edition, 1993.

[Hertzberg, 1989] Joachim Hertzberg. *Planen: Einführung in die Planerstellungsmethoden der künstlichen Intelligenz.* B-I Wissenschaftsverlag, 1989.

[Kambhampati and Kedar, 1991] S. Kambhampati and S. Kedar. Explanation-based generalization of partially ordered plans. In In MIT Press (Herausgeber), editor, *Proceedings Ninth National Conference on Artificial Intelligence*, volume 2, pages 679–685, Menlo Park, California, 1991. MIT Press.

[Knoblock, 1989] C. A. Knoblock. A theory of abstraction for hierachical planning. In *Proceedings of the Workshop on Change of Representation and Inductive Bias*, pages 81–104, Boston, MA, 1989. Kluwer.

[Knoblock, 1990] C. A. Knoblock. A theory of abstraction for hierachical planning. In *Proceedings of the Workshop on Change of Representation and Inductive Bias*, pages 81–104, Boston, MA, 1990. Kluwer.

[Knoblock, 1991] C. A. Knoblock. Search reduction in hierarchical problem solving. In *Proceedings of the AAAI-1991*, pages 686–691, 1991.

[Knoblock, 1993] C. A. Knoblock. *Machine Learning methods for planning.* Morgan Kaufmann, 1993.

[Korf, 1987] R.E. Korf. Planning as search: A quantitative apporach. *Artifical Inteligence*, 33(1):65–88, 1987.

[Langley *et al.*, 1987] P. Langley, J. Gennari, and W.Iba. Hill climbing theories of learning. In *Proceedings of the Fourth International Workshop on Machine Learning*, pages 312–323, Irvine, CA, 1987.

[Lifschitz, 1986] V. Lifschitz. On the semantics of strips. In *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop*, pages 1–9, Timberline, Oregon, 1986.

[Minton *et al.*, 1989a] S. Minton, J. Carbonell, C. Knoblock, D. Kuokka, O. Etzioni, and Y. Gil. Explanation-based learning: A problem solving perspective. *Atificial Inteligence*, 40:63–119, 1989.

[Minton *et al.*, 1989b] Steven Minton, Jaime G. Carbonell, Craig A. Knoblock, Daniel R. Kuokka, Oren Etzioni, and Yolanda Gil. Explanation-based learning: A problem solving perspective. *Artifical Inteligence*, 40:63–118, 1989.

[Minton, 1990] S. Minton. Quantitativ results concerning the utility of explanation-based learning. *Artifical Intelligence*, 42:363–391, 1990.

[Mitchell *et al.*, 1986] T. M. Mitchell, R. M. Keller, and S. T. Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning*, 1(1):47–80, 1986.

[Paulokat *et al.*, 1992] Jürgen Paulokat, Reinhard Präger, and Stefan Wess. CabPlan - fallbasierte Arbeitsplanung. In T. Messner and A. Winkelhofer, editors, *Beiträge zum 6. Workshop Planen und Konfigurueren*, number 166 in FR-1992-001, page 169, Germany, March 1992. Forwiss.

[Pearl, 1984] J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving.* Addison-Wesley, 1984.

[Pews, 1994] Gerhard Pews. MoCAS/2 - Fallbasierte Diagnose in technischen Domänen: Die Verwendung zur Ähnlieitbestimmung und Fallübertragung durch Integration von Wissen aus technischen Modellen. Diplomarbeit Universität Kaiserslautern, May 1994.

[Rich, 1988] Elaine Rich. *Künstliche Intelligenz: Einführung und Anwendung.* McGraw-Hill, 1988.

[Richter, 1989] Michael M. Richter. *Prinzipien der Künstlichen Intelligenz.* Teubner Verlag, 1989.

[Surmann, 1993] Dagmar Surmann. Implementierung und vergleichende Untersuchung verschiedener Varianten abstraktionsbasierter Planungsverfahren. Projektarbeit, Universität Kaiserslautern, 1993.

[Wielemaker and Anjewierden, 1992] J. Wielemaker and A. Anjewierden. *Programming in PCE/Prolog.* SWI, University of Amsterdam, Roetersstraat 15,1018 WB Amsterdam, The Netherlands, E-mail: {jan,anjo}@swi.psy.uva.nl, 1992.

[Wielemaker, 1992] J. Wielemaker. *SWI-Prolog, Reference Manual.* SWI, University of Amsterdam, Roetersstraat 15,1018 WB Amsterdam, The Netherlands, E-mail: jan@swi.psy.uva.nl, 1992.

[Wilke, 1993] W. Wilke. Entwurf und Implementierung eines Algorithmus zum wissensintensiven Lernen von Planabstraktionen nach der PABS-Methode. Projektarbeit, Universität Kaiserslautern, July 1993.

[Wille, 1992] Rudolf Wille. Concept lattices and conceptual knowledge systems. *Computers Mathematical Applications*, 23(6-9):493–515, 1992.

[Yoo and Fisher, 1991] J. Yoo and D. Fisher. Concept formation over explanations and problem-solving experience. In J. Mylopoulos and R. Reiter, editors, *Proceedings of the Twelfth International Conference on Artificial Intelligence*, volume 2, pages 630–637, San Mateo, CA, 1991. Morgan Kaufmann.