Lernen von Abstraktionshierarchien zur Optimierung der Auswahl von maschinell abstrahierten Plänen

Ralph Bergmann und Wolfgang Wilke
Universität Kaiserslautern
FB Informatik - AG-Richter
Postfach 3049
67653 Kaiserslautern
E-Mail: {bergmann,wilke}@informatik.uni-kl.de

Zusammenfassung

Mit Hilfe von "Multistrategy" Ansätzen, die erklärungsbasiertes und induktives Lernen integrieren, ist es möglich, die Performanz von Planungssystemen signifikant zu verbessern. Dabei können gelöste Planungsprobleme zunächst mit einem wissensintensiven Verfahren abstrahiert und generalisiert werden. Durch den in diesem Beitrag im Vordergrund stehenden induktiven inkrementellen Lernalgorithmus ist es dann weiterhin möglich, die Gesamtheit des deduktiv generierten Wissens in einer Abstraktionshierarchie anzuordnen. Dabei wird die, im allgemeinen unentscheidbare, "spezieller-als-Relation" zwischen generalisierten Plänen, induktiv aus den gegebenen Planungsfällen gelernt. Diese Abstraktionshierarchie dient dann zur Klassifikation neuer Problemstellungen und damit zur Bestimmung einer speziellsten anwendbaren abstrakten Problemlösung.

1 Einleitung und Motivation

Im Bereich des Maschinellen Lernens sind bislang einige Ansätze untersucht worden, um durch Lernen die Leistungsfähigkeit von Planungssystemen zu erhöhen. Da eines der zentralen Probleme bei der Planung die hohe Komplexität der Suchprozesse ist, ist eines der Hauptanliegen, geeignetes Wissen zur gezielten Einschränkung der Suche maschinell zu akquirieren. Da in Planungsdomänen in der Regel hinreichendes Domänenwissen vorhanden ist, werden häufig erklärungsbasierte Verfahren [Mitchell *et al.*, 1986; Ellman, 1989] zum sogenannten "Speedup-Learning" oder zur "Knowledge Compilation" [Fisher *et al.*, 1992] eingesetzt, mit denen Kontrollregeln [Minton *et al.*, 1989], Makro-Operatoren [Fikes *et al.*, 1972; Korf, 1985; Tadepalli, 1991] oder Planskelette [Friedland und Iwasaki, 1985; Bergmann, 1992a] aus Beispielen oder Problemlöseversuchen gelernt werden können. Es hat sich jedoch bei all diesen Ansätzen gezeigt, daß eine pure erklärungsbasierte Lernstrategie häufig daran scheitert, daß die Suche nach anwendbaren Kontrollregeln aufwendiger werden kann als die Suche im Problemraum selbst. Dieser Effekt, wurde von Minton [Minton, 1990] auch als "Utility" Problem bezeichnet.

In jüngster Zeit wurden nun einige "Multistrategy" Ansätze [Michalski und Tecuci, 1991] verfolgt, bei denen verschiedene Lernverfahren geeignet kombiniert werden, so daß sich ein Synergieeffekt zeigt. Desweiteren wurde immer deutlicher, daß auch eine stärkere Integration von Lernverfahren mit den zugehörigen Problemlöseverfahren [Plaza *et al.*, 1993; Aamodt und Plaza, 1994] als Ausweitung das fallbasierten Paradigmas [Kolodner, 1993; Althoff *et al.*, 1992] erforderlich ist.

In diesem Sinne wird das PARIS-System (**P**lan **A**bstraction and **R**efinement in an **I**ntegrated **S**ystem) vorgestellt, in dem erklärungsbasiertes Lernen, Abstraktion, und induktives Lernen kombiniert werden. Hierdurch können aus gelösten Planungsproblemen generalisierte abstrakte Pläne erlernt werden, die die Komplexität zur Lösung neuer Planungsprobleme signifikant reduzieren. Hierbei werden die deduktiv abstrahierten Pläne durch die induktive Komponente in einer Abstraktionshierarchie angeordnet. In dieser Hierarchie sind abstraktere und damit vielseitiger anwendbare Abstraktionen höher angeordnet als speziellere. Speziellere Abstraktionen bewirken jedoch, falls sie für ein gegebenes Problem anwendbar sind, eine stärkere Einschränkung des Suchraumes als Abstraktionen auf einen höheren Niveau. Aus diesem Grund ist es erstrebenswert, immer eine speziellste anwendbare Abstraktion zur Problemlösung heranzuziehen. Ohne die Information über die Ordnung der Abstraktionen bzgl. ihres Allgemeinheits- oder Abstraktionsgrades, müssen immer alle verfügbaren Abstraktionen beim Problemlösen überprüft werden, was bei

großen Mengen das Utility-Problem zur Folge hat. Genau dies wird durch die induzierte Abstraktionshierarchie verhindert, da diese zur Klassifikation der Problemstellung und damit zur effizienten Auswahl eines speziellsten anwendbaren abstrakten Planes herangezogen werden kann.

Im folgenden wird nun der Schwerpunkt dieses Beitrages auf die Darstellung des induktiven Lernalgorithmus zum Aufbauen des Klassifikationsbaums gelegt. Da dieser jedoch nur in Kombination mit der Planabstraktions- und Generalisierungsmethodik verständlich ist, werden die hiervon erforderlichen Bestandteile zuerst eingeführt. Auf die vollständige Darstellung des integrierten Systemes PARIS muß hier jedoch verzichtet werden. Für nähere Information, auch in Bezug auf experimentelle Performanzresultate sei hier auf [Bergmann, 1993] verwiesen.

2 Integration von Abstraktion und Generalisierung

Im folgenden Abschnitt wird nun zunächst die PABS-Planabstraktionsmethodik beschrieben. Dabei handelt es sich um ein wissensintensives Lernverfahren, das Abstraktion [Tenenberg, 1986; Michalski, 1993] mit erklärungsbasiertem Lernen [Mitchell *et al.*, 1986; Ellman, 1989] verbindet. Ziel ist es dabei, aus vorliegenden gelösten Planungsproblemen abstrakte, generalisierte Pläne zu lernen, die dann für die Lösung neuer Planungsprobleme komplexitätsreduzierend eingesetzt werden können.

2.1 Grundbegriffe

Im folgenden soll zunächst die grundlegende Terminologie linearer Planung kurz skizziert werden, da hierauf öfter zurückgegriffen wird.

Eine Planungswelt W ist ein STRIPS-System, bestehend aus einer Menge R eine Menge von essentiellen Sätzen [Lifschitz, 1987], einer statischen Theorie T zur Ableitung zusätzlicher Fakten und einer Menge Op von Operatoren die, wie üblich, durch Vorbedingung, Add- und Deleteliste beschrieben sind. Ein Zustand s der Planungswelt W ist nun, eine Untermenge der essentiellen Sätze R aus W. Die Menge $S = 2^R$ bezeichne den Raum aller möglichen Zustände in W. Ein Planungsproblem ist nun durch ein Zustandspaar (s_i, s_g) bestimmt, welches den Startzustand sowie den Zielzustand beschreibt.
Ein Plan P ist eine Sequenz von Operatoren $(o_1, \ldots, o_n)$ mit $o_i \in Op$ für $i = 1, \ldots, n$. Als Planungsfall wird nun ein Planungsproblem zusammen mit einem Plan angesehen, der dieses Problem löst.

2.2 Modell der Planabstraktion

Im folgenden wird nun zunächst das formale Modell der Planabstraktion nach [Bergmann, 1992b] eingeführt. Hierzu werden zwei disjunkte Planungswelten, sowie eine generische Abstraktionstheorie vorausgesetzt:

Definition 1 *Die konkrete Welt W_k ist eine Planungswelt, die 'reale' Zustände und Operatoren beschreibt.*

Planungsprobleme werden immer in der Terminologie der konkreten Welt definiert und die gewünschten Lösungen sind Pläne der konkreten Welt.

Definition 2 *Die abstrakte Welt W_a ist eine Planungswelt, die 'virtuelle' – wir sagen abstrakte – Zustände und Operatoren beschreibt. Diese Operatoren sind faktisch nicht ausführbar, liefern jedoch nützliche Einheiten zur allgemeinen Beschreibung einer Klasse von möglichen Problemen.*

Wir haben nun also zwei unterschiedliche Ebenen zur Beschreibung von Planungsfällen mit vollkommen getrennter Terminologie. Hierin unterscheidet sich der vorgestellte Ansatz zur Abstraktion deutlich von klassischen Verfahren z.B. im hierarchischen Planen (z.B. [Sacerdoti, 1974; Knoblock, 1993]).

Um einen Bezug zwischen den beiden Beschreibungswelten herzustellen, wird weiterhin eine generische Abstraktionstheorie vorausgesetzt.

Definition 3 *Eine generische Abstraktionstheorie T_g ist 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 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.*

Diese Theorie erlaubt es, aus einem gegebenen konkreten Zustand, die Menge aller potentiell möglichen abstrakten Zustände zu bestimmen.

Planabstraktion bedeutet nun, einen gegebenen Plan pk in der konkreten Welt W_k in einen korrespondierenden Plan pa der abstrakten Welt W_a abzubilden. Die genaue Definition der Planabstraktion

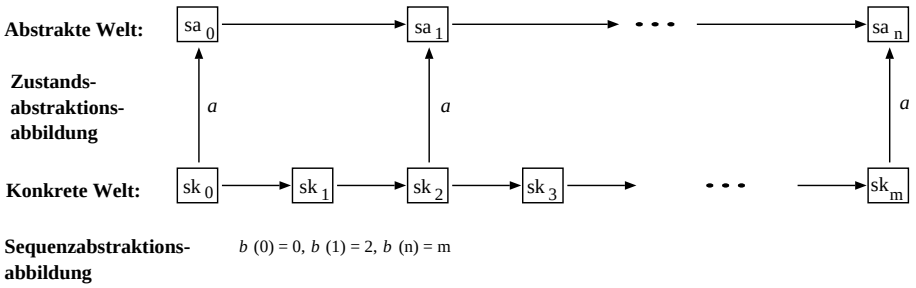

Abbildung 1: *Konkrete und abstrakte Planungswelt mit den Abbildungen* (a,b)

erfolgt mit Hilfe zweier Abbildungen: der Zustandsabstraktionsabbildung und der Sequenzabstraktionsabbildung. Durch die Zustandsabstraktionsabbildung werden die konkreten Zustandsbeschreibungen $s_k \in S_k$ unter Beachtung der generischen Theorie T_g in die abstrakten Zustandsbeschreibungen $s_a \in S_a$ abgebildet. Die Sequenzabstraktionsabbildung hingegen ordnet den abstrakten Zustandsindizes entsprechende konkrete Zustandsindizes zu, wobei der abstrakte Startzustand sa_0 auf den konkreten Startzustand sk_0 und der abstrakte Endzustand sa_n auf den konkreten Endzustand sk_m abgebildet wird. Darüber hinaus ist diese Abbildung ordnungserhaltend.

Dieser Begriff der Planabstraktion wird in Abbildung 1 veranschaulicht: Die Abbildung a bildet die konkreten Zustände der konkreten Planungswelt auf die abstrakten Zustände der abstrakten Planungswelt ab. Die Abbildung b bildet die Indizes der abstrakten Planungszustände auf die Indizes der Zustände der konkreten Planungswelt ab, die durch einen konkreten Plan pk und einen Startzustand sk_0 induziert werden. In der Regel werden dabei von einem abstrakten Operator mehrere konkrete Operatoren überbrückt.

Bei dieser Definition von Planabstraktion ist zu beachten, daß es zu einem gegebenen konkreten Plan in der Regel mehrere abstrakte Pläne gibt. Hierbei stellen verschiedene abstrakte Pläne verschiedene Sichtweisen oder Abstraktionsgrade dar. Die Terminologie dieser Sichten muß jedoch in der vorliegenden abstrakten Planungswelt vorgegeben sein.

Der Wert der abstrakten Pläne für die Planung besteht nun im wesentlichen darin, daß sie eine Suchraumeinschränkung auf der konkreten Ebene darstellen. Im allgemeinen, d.h. für beliebige abstrakte Planungswelten, kann jedoch nicht garantiert werden, daß sich ein abstrakter Plan auch immer verfeinern läßt. Dennoch stellen abstrakte Pläne eine gute Heuristik dar. Eine genauere Analyse dieses Sachverhalts kann in [Knoblock, 1993] gefunden werden.

2.3 Generalisierung von Plänen

Neben der Abstraktion ist auch die Generalisierung eine Möglichkeit gegebene Planungsfälle für ein größeres Spektrum neuer Situationen anwendbar zu machen. Deduktive Lernverfahren zur Plangeneralisierung sind schon lange bekannt (z.B. [Fikes *et al.*, 1972]) und sind im wesentlichen eine Variante des erklärungsbasierten Lernens [Ellman, 1989], wobei die gegebenen Operatorbeschreibungen als Hintergrundtheorie verwendet werden. In [Bergmann, 1992a] ist ein solches Generalisierungsverfahren ausführlich beschrieben. Hierbei wird unter Berücksichtigung der Interaktionen der Operatoren des Plans eine allgemeinste Bedingung bestimmt, die gewährleistet, daß die gegebene Operatorfolge (mit ggf. generalisierten Parametern) weiterhin anwendbar ist. Hierzu werden bestimmte Konstanten in der Problembeschreibung und im Plan durch Variablen ersetzt, wobei zusätzlich gefordert wird, daß diese Variablen in bestimmten Relationen zueinander stehen. Diese Relationenmenge wird durch das Lernverfahren automatisch aus der Beschreibung der Planungswelt und dem Planungsfall generiert. Ein resultierender generalisierter Plan hat dabei die folgende Gestalt:

Definition 4 $GP \equiv (s_i(x), s_g(y), < o_1(z_1), ..., o_n(z_n) >, R(x, y, z_1, ..., z_n))$ *ist ein generalisierter Plan. Hierbei bezeichnen* $s_i(x)$ *den Initialzustand und* $s_g(y)$ *den Zielzustand des generalisierten Planes.*
$< o_1(z_1), ..., o_n(z_n) >$ *ist die Folge von generalisierten Operationen und* R *eine Menge von Relationen.* x, y *und* z_i *bezeichen Variablen, die die Parameter der Operationen mit Hilfe der Relationen mit dem Initial- und Zielzustand in Beziehung setzen.*

Eine wesentliche Eigenschaft dieses deduktiven Generalisierungsprozesses ist es nun, daß die generalisierten Pläne 'korrekt' sind in dem Sinne, daß wenn es eine Substitution σ gibt, die die Variablen x, y und z_i so instantiiert, daß alle Relationen in R erfüllt sind, so löst der Plan $< o_1(\sigma(z_1)), ..., o_n(\sigma(z_n)) >$ das Planungsproblem $(s_i(\sigma(x)), s_g(\sigma(y)))$.

2.4 Das PARIS-System

Das PARIS-System stellt eine Integration der auf Abstraktion und Generalisierung basierenden *Lernkomponente* mit einer entsprechenden *Planungskomponente* dar.

Das Lernverfahren, daß Abstraktion und Generalisierung integriert, ist ausführlich in [Bergmann, 1992b; Wilke, 1993] beschrieben und soll hier nur kurz skizziert werden. Das Verfahren besteht aus fünf aufeinanderfolgenden Phasen. In der ersten Phase wird die Ausführung des konkreten Plans simuliert. In der zweiten Phase wird die generische Theorie angewandt, um die durch die Ausführung des Plans entstandenen Zustandsbeschreibungen der konkreten Welt auf abstrakte Zustandsbeschreibungen abzubilden. Das Ziel der dritten Phase ist es, alle abstrakten Operatoren zu finden, die von einem abstrakten Zustand in einen beliebigen Folgezustand führen. In der vierten Phase werden alle konsistenten Pfade in der abstrakten Planungswelt gesucht, die vom abstrakten Startzustand zum abstrakten Zielzustand führen. In der letzten Phase werden die abstrakten Pläne erklärungsbasiert generalisiert. Als Resultat liefert das Lernverfahren zu einem gegebenen konkreten Plan die Menge aller generalisierten abstrakten Pläne [1] – also generalisierte Pläne in der abstrakten Welt.

In der Planungskomponente können diese Generalisierungen nun zur Lösung eines neuen Planungsproblems herangezogen werden. Hierzu muß zunächst geprüft werden, ob ein generalisierter abstrakter Plan in einer konkreten Situation anwendbar ist. Ein generalisierter abstrakter Plan ist genau dann anwendbar, wenn die folgende Bedingung gilt:

Definition 5 *Ein generalisierter Plan in der abstrakten Welt* $GP \equiv (s_i(x), s_g(y), < O_1(z_1), ..., o_n(z_n) >, R(x, y, z_1, ..., z_n))$ *ist für ein Planungsproblem* $(\tilde{s}_i, \tilde{s}_g)$ *mit* $\tilde{s}_i, \tilde{s}_g \in S_k$ *anwendbar, falls:*

- *es gibt eine Substitution* σ*, die die Variablen* x, y *und* z_i *so belegt, daß alle Relationen in* R *erfüllt sind und*
- *es gibt eine Zustandsabstraktionsabbildung* a*, so daß gilt:* $s_i(\sigma(x)) = a(\tilde{s}_i)$*,* $s_g(\sigma(y)) = a(\tilde{s}_g)$

Das bedeutet, daß eine geeignete Abstraktion des Planungsproblems gefunden werden muß und eine Instantiierung der im abstrakten generalisierten Plan vorkommenden Variablen. Ist dies geschehen, so kann der resultierende abstrakte Plan und zu einem konkreten Plan verfeinert werden [Friedland und Iwasaki, 1985]. Eine ausführliche Beschreibung dieses Prozesses kann in [Bergmann, 1993] gefunden werden.

Die Schnittstelle zwischen der Lernkomponente und der Planungskomponente ist also die Menge der abstrakten Pläne. Hierbei besteht jedoch das Problem, aus einer Menge abstrakter Pläne überhaupt einen geeigneten Plan auf effiziente Weise auszuwählen. Hierzu müssen die abstrakten Pläne geeignet organisiert werden. Im folgenden wird nun ein Ansatz vorgestellt mit dem eine sinnvolle Anordnung induktiv gelernt werden kann.

3 Organisation abstrakter Pläne in einer Abstraktionshierarchie

Die bislang vorgestellte Methode zum wissensintensiven Lernen von abstrakten Plänen ist in der Lage, für einen gegebenen Planungsfall eine Menge von gültigen abstrakten Plänen zu generieren, die zur Lösung neuer Planungsprobleme komplexitätsreduzierend eingesetzt werden können. Im folgenden wird nun ein Ansatz zum Lernen einer Klassifikationshierarchie von abstrakten Plänen vorgestellt.

3.1 Eine Klassifikationshierarchie für abstrakte Pläne

Die gelernten abstrakten Pläne können auf verschiedenen Abstraktionsebenen angesiedelt sein. Ein Plan auf einer höheren Abstraktionsebene zerlegt ein gegebenes Planungsproblem in eine geringere Zahl von

[1] Im folgenden wird der Einfachheit halber nur von einem abstrakten Plan gesprochen. Dabei ist jedoch immer ein generalisierter abstrakter Plan gemeint.

Teilproblemen, während ein speziellerer abstrakter Plan mehrere, aber weniger komplexe Teilprobleme erzeugt. Insgesamt läßt sich jedoch sagen, daß die Komplexität zur Verfeinerung eines spezielleren abstrakten Plans niedriger ist, als die Komplexität zur Verfeinerung eines Plans auf einer höheren Abstraktionsebene.

Desweiteren ist es erforderlich, die Gesamtmenge der gelernten Abstraktionen in einer geeigneten Art und Weise zu organisieren, um den Zugriff während des Problemlösens zu optimieren. Andernfalls läuft man Gefahr, alle gelernten Regeln immer vollständig durchsuchen zu müssen und somit am Utility-Problem [Minton, 1990] zu scheitern. Als ein Ansatz zur Lösung dieses Problems wurde von [Yoo und 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, daß falls eine Regel an einem Knoten nicht anwendbar ist, daß auch keine der Regeln an den Nachfolgerknoten anwendbar ist. Zum Finden der speziellsten anwendbaren Regel kann nun der Baum solange durchlaufen werden, wie es noch Nachfolgerknoten gibt, die anwendbar sind. Hat ein Knoten keine anwendbaren Nachfolger, so ist eine (aber nicht notwendigerweise die einzige) speziellste anwendbare Regel gefunden.

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. Hierbei konkretisiert sich die Bedingung, wann ein abstrakter Plan spezieller ist als ein anderer abstrakter Plan wie folgt:

Definition 6 *$GP_1 \sqsubseteq GP_2$ (GP_1 ist spezieller als GP_2) gdw für alle konkreten Zustandspaare $si, sg \in S_k$ gilt, daß falls GP_1 für (si, sg) anwendbar ist, auch GP_2 für (si, sg) anwendbar ist.*

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. Das ist das zentrale Anliegen dieses Lernansatzes. Desweiteren wird vorausgesetzt, daß in der Beschreibung der abstrakten Planungswelt ein universeller abstrakter Operator vorgegeben ist, der immer anwendbar ist und alle Effekte (auf der höchsten Abstraktionsebene) erreicht. Dies garantiert, daß es immer einem anwendbaren abstrakten Plan gibt, der genau aus diesem einen Operator besteht. Jedoch bewirkt dieser Plan keine Suchraumeinschränkung bei der Planung, garantiert jedoch, daß zumindest prinzipiell alle Planungsprobleme gelöst werden können.

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 im allgemeinen nicht entscheidbar ist, da die Zustandsräume S_k und S_a unendlich sein können. Auch die PABS-Planabstraktionsmethodik kann nicht zur Überprüfung dieser Bedingung herangezogen werden, da sich zum einen die abstrakten Pläne in derselben Welt W_a befinden und zum anderen diese Pläne bereits generalisiert sind.

3.2 Induktives Lernen der Klassifikationshierarchie

Da es nun also nicht möglich ist, eine Klassifikationshierarchie zu erstellen, der bezüglich der $\sqsubseteq$-Halbordnung korrekt ist, versuchen wir nun diese Ordnung – und damit die Hierarchie – induktiv aus den Planungsfällen zu lernen. Hierzu wird zunächst aus einem neuen konkreten Planungsfall (si_j, sg_j) die Menge aller gültigen abstrakten Pläne $B_j = \{GP_1, ..., GP_k\}$ generiert. Jeder der abstrakten Pläne in B_j ist nun anwendbar für diesen Planungsfall und umgekehrt, ist jeder abstrakte Plan, der für diesen konkreten Planungsfall anwendbar ist, auch in B_j enthalten. 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 die Abstraktionskomponente jedoch immer alle gültigen abstrakten Pläne für einen gegebenen konkreten Planungsfall liefert, ist sichergestellt, daß falls $GP \in B_j$ gilt, daß dann auch für alle GP' mit $GP' \sqsupseteq GP$ gilt, daß $GP' \in B_j$. 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 Planungsproblemen $P = ((si_1, sg_1), ..., (si_n, sg_n))$ und zugehörigen abstrakten Planmengen $B = (B_1, ..., B_n)$ – ein Klassifikations-

baum entstehen, der alle abstrakten Pläne aus B einordnet und für alle bislang gesehenen Planungsprobleme P 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 $\sqsubseteq_P$ - Relation sein.

Definition 7 *$GP_1 \sqsubseteq_P GP_2$ (GP_1 ist für P spezieller als GP_2) gdw für alle konkreten Zustandspaare $(si, sg) \in P$ gilt: falls GP_1 für (si, sg) anwendbar ist, auch GP_2 für (si, sg) anwendbar ist. Beachte: $\sqsubseteq = \sqsubseteq_{S_k \times S_k}$.*

Aufgrund der Eigenschaft der Trainingsbeispiele ist es nun möglich zu überprüfen, ob der aktuelle Baum einem neuen Trainingsbeispiel widerspricht. Ist also z.B. in einem aktuellen Baum $GP_1 \sqsubseteq GP_2$ angeordnet und kommt in einem neuen Trainingsbeispiel B_j der Plan GP_1 vor, nicht jedoch GP_2, so liegt ein Widerspruch vor und der Baum muß umgestaltet werden. Genau dies ist die Aufgabe des Lernalgorithmus.

3.3 Ein Beispiel

Um die bislang vorgestellten Überlegungen zu verdeutlichen, wird nun ein einfaches Beispiel vorgeführt. Hierbei werden konkrete Planungsfälle $F_1, ..., F_5$ angenommen. Wir nehmen weiter an, daß die abstrakte Welt gerade so beschrieben ist, daß es für jeden Planungsfall mehrere abstrakte Pläne gibt.

Die genaue Zuordnung der abstrakten Pläne $GP_1, ..., GP_5$ zu den konkreten Fällen ist in Tabelle 1 dargestellt. Man sieht hier schon an der Tabelle, daß der abstrakte Plan GP_1 für alle konkreten Planungsfälle

	GP_1	GP_2	GP_3	GP_4	GP_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 1: Anwendbarkeit der abstrakten Pläne für die konkreten Fälle

gültig ist.

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, so daß drei Trainingsbeispiele für den Lernalgorithmus zur Verfügung stehen, nämlich: $B_1 = \{GP_1, GP_2\}$, $B_2 = \{GP_1, GP_2, GP_3\}$, $B_3 = \{GP_1, GP_4, GP_5\}$.

Betrachtet man nun, welche Hierarchien 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 $\sqsubseteq$-Relationen gibt, wenn man annimmt, daß es einen abstraktesten Plan gibt, der für jedes Problem anwendbar ist. Von diesen sechs Hierarchien sind vier in Abbildung 2 dargestellt.

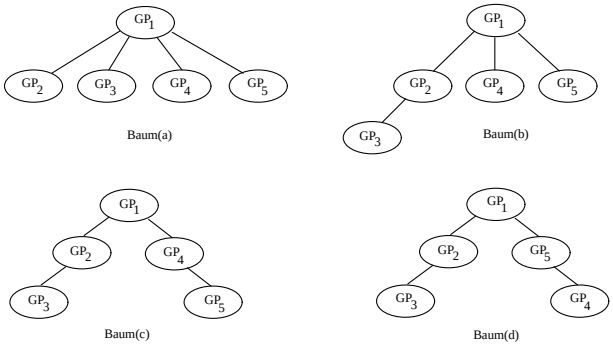

Abbildung 2: *Mögliche Klassifikationsbäume für F_1 F_2 F_5*

3.4 Der Lernalgorithmus

In diesem Abschnitt wird nun der Lernalgorithmus 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 an, so daß die neue Hierarchie sowohl korrekt bzgl. des neuen Beispiels ist, als auch korrekt bzgl. aller vorherigen Beispiele bleibt. Die bereits abgearbeiteten Trainingsbeispiele werden weder im Baum noch außerhalb gespeichert.

Wie bereits im vorherigen Abschnitt 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 mitgeführt werden, da ansonsten die Hypothesenmenge zu groß wäre. Deshalb führt der Algorithmus eine Art Hill-Climbing [Langley *et al.*, 1987] durch und hält nur die "vielversprechensten" Ordnungen fest. Ordnungen, die "tief" sind, wie z.B. Baum (c) in Abbildung 2, 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. 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. Im folgenden soll nun der durch den Algorithmus verwaltete Klassifikationsbaum und dessen Korrektheit definiert werden.

Definition 8 *Eine Klassifikationshierarchie für abstrakte Pläne ist ein Baum mit der Wurzel K_0 und den Knoten K_i ($i = 1...m$). Jedem Knoten ist eine Menge von abstrakten Plänen AP_i zugeordnet. Eine Klassifikationshierarchie ist bzgl. einer Beispielmenge $B = \{B_1, ..., B_n\}$ korrekt, falls:*

- $\forall_{j=1..n}\forall_{i=0..m}(AP_i \cap B_j = \emptyset$ *oder* $AP_i \subseteq B_j)$, *d.h. für alle Trainingsbeispiele und jeden Knoten im Baum sind entweder immer alle abstrakten Pläne für ein Beispiel gültig 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 $AP_l \subseteq B_j$ dann auch $AP_i \subseteq B_j$, 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. $\sqsubseteq_P$.*

Der inkrementelle Algorithmus bearbeitet ein neues Beispiel in zwei Schritten. Zunächst wird getestet, ob der aktuelle Baum auch für das neue Beispiel konsistent ist. Falls dies nicht der Fall ist, wird der Baum korrigiert (Prozedur `Update`). In einem zweiten Schritt werden dann alle neuen abstrakten Pläne, die bislang noch nicht im Baum vorgekommen sind, auf konsistente Weise hinzugenommen. Dieser Toplevel-Algorithmus `LearnHierarchy`(K_0,B_{neu}), der als Eingabe den aktuellen Baum sowie das neue Beispiel erhält, ist im folgenden dargestellt.

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

3.4.1 Überprüfen und Erhalten der Korrektheit der Hierarchie

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 gelten oder aber alle nicht gelten. Liegt jedoch der Fall vor, daß ein Knoten sowohl gültige als auch ungültig Pläne enthält, so muß der Baum an dieser Stelle "repariert" werden. Hierzu wird der "gemischte" Knoten in zwei neue Knoten geteilt, wie in Abbildung 3 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

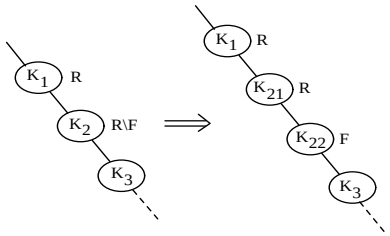

Abbildung 3: *Aufteilen eines Mehrfachknotens: R bezeichnet einen Knoten mit gültigen Plänen, F einen Knoten mit ungültigen Plänen; R/F ist ein gemischter Knoten*

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 Korrektheitsbedingung 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 Korrektheitsbedingung verletzt 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 4 beispielhaft dargestellt. Hier ist der Knoten K_4 gültig, obwohl

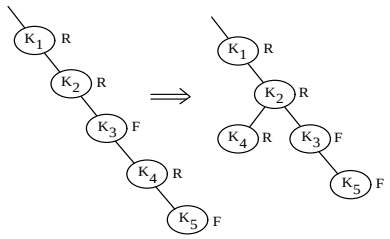

Abbildung 4: *Hochziehen eines (Teil)-Knoten*

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 Vorgänger Knoten 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 nicht flach) ist, wählen wir hier immer die tiefste mögliche Stelle. Beim Betrachten neuer Trainingsbeispiele ist es jedoch möglich, daß der Knoten trotzdem weiter nach oben wandert.

Dieses Hochziehen des Knotens korrigiert den Baum für das aktuelle Beispiel und erhält ebenfalls die Korrektheit für alle früheren Beispiele. Der formale Beweis für diese Eigenschaft muß hier leider aus Platzgründen unterbleiben.

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 $AP_i \subseteq B_{neu}$ **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 $AP_i \cap B_{neu} = \emptyset$ **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 enthählt 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) $AP_u := AP_i \cap B_{neu}$
e) $AP_n := AP_i - AP_u$
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 $AP_i \cap B_{neu} = \emptyset$ **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:
$AP_{neu} := AP_i \cap B_{neu}$
2. Setze $AP_i := AP_i - B_{neu}$
IF $AP_i = \emptyset$ **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

3.4.2 Einfügen des neuen Beispiels

Nachdem nun also die Klassifikationshierarchie bzgl. des neuen Beispieles 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. Wir verwenden nun zum Bestimmen eines guten Einfügepunktes eine Heuristik, die uns einen möglichst tiefen Baum erzeugen soll. Hierzu verfolgen wir einen Pfad, ausgehend von der Wurzel des Baumes, solange wie alle die abstrakten Pläne für das aktuelle Beispiel noch gültig sind oder ein Blattknoten erreicht wurde. Ist eine solche Stelle im Baum gefunden, wird ein neuer Knoten angehangt 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. Wir wählen den Nachfolgerknoten, 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.

3.5 Ein Beispiellauf des Lernalgorithmus

Um diesen Lernalgorithmus an einem Beispiel zu verdeutlichen, wird er für die Beispiele in Tabelle 1 angewandt. Hierbei wollen wir annehmen, daß die Planungsfälle in der Reihenfolge F_2, F_5, F_1, F_4, F_3 vorgegeben werden. Abbildung 5 zeigt die resultierende Folge von Klassifikationsbäumen. Nachdem das erste Beispiel F_2 bearbeitet wurde, wird genau ein Knoten erzeugt, der alle abstrakten Pläne (GP_1, GP_2, GP_3) dieses Beispiels enthält. Bei dem zweiten Beispiel (F_5) stellt nun die `update` Prozedur fest, daß der bisherige Knoten, sowohl anwendbare Pläne (nämlich GP_1) enthält, als auch nicht anwendbare Pläne (GP_1 und GP_2). Deshalb wird der Knoten zunächst geteilt (siehe Baum b1). Danach werden beim `Einsortieren` die fehlenden Pläne GP_4 und GP_5 zusammen an einem Knoten angehängt (siehe Baum b2). Das Bearbeiten des Beispiels F_1 führt nun weiter dazu, daß der Knoten mit GP_2 und GP_3 gesplittet wird (Baum c). Weitere Pläne kommen

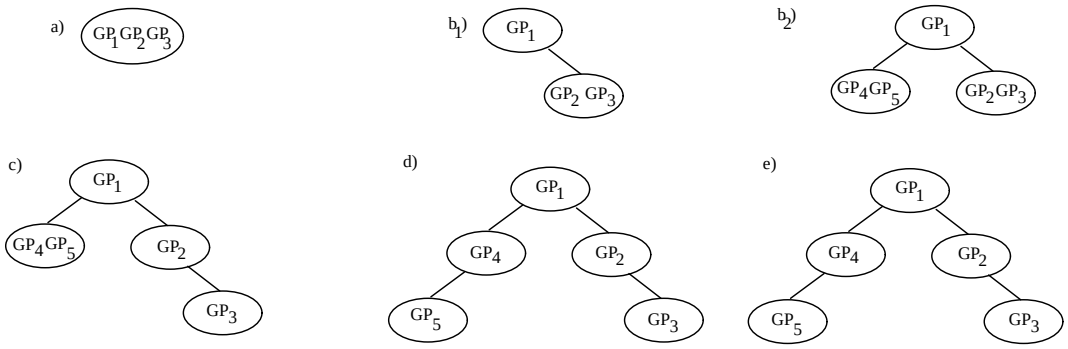

Abbildung 5: *Beispielfolge von Klassifikationshierarchien*

nun nicht mehr hinzu, da bereits alle Pläne aus dem (und allen folgenden) Beispiel im Baum vorhanden sind. Das Beispiel F_4 bewirkt nun lediglich noch ein Splitten eines Knotens (siehe Baum d), während das letzte Beispiel F_3 den Baum nun überhaupt nicht mehr verändert.

4 Experimentelle Untersuchung

Zur experimentellen Bewertung des vorgestellten Ansatzes, wurde dieser auf einen Teilbereich der Arbeitsplanung, nämlich auf die Planung der Fertigung von rotationssymmetrischen Drehteilen – die CabPlan-Domäne [Paulokat und Wess, 1993] – angewendet. Hierzu sind entsprechende Beschreibung von konkreter- und abstrakter Planungswelt, sowie die generische Theorie entwickelt worden. Eine Fallbasis von 116 verschiedenen Planungsfällen wurde sodann automatisch generiert.

In der Trainingsphase wurde das PARIS-System in zwei verschiedenen Testläufen mit

- allen Planungsfällen
- einer zufälligen Auswahl von 10% der Fälle

trainiert.

Danach wurde versucht, alle 116 Planungsprobleme mit dem so trainierten System zu lösen. Hierbei wurden jeweils die Problemlösezeiten gemessen. Als Vergleichsbasis wurden zusätzlich alle Planungsprobleme durch reine Suche, also ohne heuristische Suchraumeinschränkung, gelöst. Für die Problemlösung wurde dabei jeweils eine Zeitlimit von 200 CPU-Sekunden angesetzt. Ist dieses überschritten worden, so galt das Problem als nicht gelöst. Die Ergebnisse dieses Experiments sind in Abbildung 6 dargestellt. Die durchschnittliche Problemlösezeit ist in Abhängigkeit der Problemkomplexität (Länge der Lösung) aufgetragen. Es zeigt sich, daß für Planung durch Suche die Problemlösezeit exponentiell mit der Problemkomplexität wächst. Probleme, die eine Lösung verlangen die länger als 8 Schritte ist, konnten nicht mehr innerhalb der Zeitschranke gelöst werden. Durch Verfeinerung von abstrakten Plänen hingegen konnten alle Problem, die eine Lösung von weniger als 17 Operatoren verlangen, gelöst werden. Komplexere Problem blieben jedoch auch hier ungelöst. Eine Analyse dieses Sachverhaltes hat gezeigt, daß bei den komplexen Problemen, ein oder mehrere abstrakte Operatoren zu einer Sequenz von mehr als 6 konkreten Operatoren verfeinert werden mußten. Deshalb wurden die Teilprobleme in diesen Fällen so groß, daß die exponentielle Natur des Suchraumes wieder zum Tragen kam. Dies kann als Indiz dafür angesehen werden, daß die Modellierung der abstrakte Planungswelt für die Arbeitsplanung einer weiteren Differenzierung (zusätzliche abstrakte Operatoren) bedarf.

Eine weitere Beobachtung aus dieser Untersuchung ist, daß bereits 10% der Planungsfälle zum Training des Systems ausreichen. Hierin zeigt sich die hohe Allgemeinheit der abstrakten generalisierten Pläne.

5 Zusammenfassung

In diesem Papier wurde das PARIS-System als Beispiel einer integierten Architektur zum Problemlösen (Planen) und Lernen im Sinne des fallbasierten Schließens [Plaza *et al.*, 1993] vorgestellt. Aus gelösten

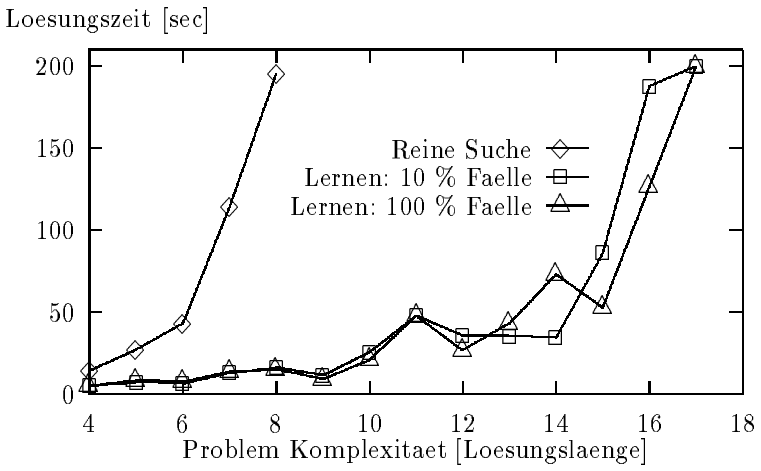

Abbildung 6: Experimentelle Ergebnisse: Planung durch Suche vs. Planung mit abstrakten Plänen

Planungsproblemen werden abstrakte generalisierte Pläne akquiriert und in einer Hierarchie angeordnet. Die Verwendung dieses gelernten Wissens führt zur Steigerung von Kompetenz und Performanz des Planungssystemes. Im Unterschied zu ähnlichen Arbeiten im Rahmen von PRODIGY [Minton *et al.*, 1989; Knoblock, 1993; Veloso und Carbonell, 1993] kann PARIS domänenspezifische Abstraktionen bilden, die den Suchraum gezielter einschränken können als ABSTRIPS-artige Abstraktionen. Hierfür muß man jedoch den Preis einer zusätzlichen Wissensakquisition von domänenspezifischen abstrakten Operatoren bezahlen.

Methodisch bietet der Ansatz eine wesentlich engere Integration von Generalisierung und Abstraktion als dies bisher in PRODIGY erreicht wurde. Darüber hinaus kann PARIS aus Planungsfällen lernen, die nicht durch das System gelöst wurden, sondern z.B. durch einen menschlichen Experten. Sowohl die Analogiekomponente als auch der EBL-Ansatz in PRODIGY verlangen hingegen, daß der Basisplaner die vorliegenden Probleme bereits selbständig lösen kann.

Der Schwerpunkt dieses Papiers liegt jedoch auf dem inkrementellen Lernalgorithmus zum Bilden einer Abstraktionshierarchie, die den Zugriff auf anwendbare abstrakte Pläne optimiert. Hierbei wurde im wesentlichen eine Halbordnung erworben, die a priori nicht bekannt war und nur indirekt aus der Struktur der Trainingsbeispiele abgeleitet werden konnte. Bei dem dargestellten Verfahren bestehen starke Bezüge zu Clusterungsalgorithmen für Begriffsbildungsaufgaben [Gennari *et al.*, 1989]. Durch die gelernte Hierarchie werden indirekt konkrete Planungsfälle in hierarchische Cluster angeordnet. So gibt es zu jedem Knoten in der Hierarchie immer auch ein Cluster von Planungsfällen. Die Menge der Planungsfälle, die dabei einem Knoten zugeordnet ist, ist hierbei immer eine Teilmenge der Planungsfälle des Vorgängerknotens. Die abstrakten Pläne an den Knoten sind nun immer eine gemeinsame Abstraktion der zugehörigen Planungsfälle. Mit anderen Worten, der vorgestellte Algorithmus clustert konkrete Planungsfälle so, daß es gemeinsame Abstraktionen gibt.

Danksagung

Die Autoren möchten sich bei Manuela Veloso und Jaime Carbonell für wertvolle Diskussionen und Anregungen bedanken. Darüber hinaus bietet das gute Arbeitsklima in der Arbeitsgruppe von Prof. Richter die beste Voraussetzung für den Fortschritt dieser Arbeit.

Literatur

[Aamodt und Plaza, 1994] A. Aamodt und E. Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications*, 1994. To be published.

[Althoff *et al.*, 1992] K.-D. Althoff, Stefan Wess, B. Bartsch-Spörl, D. Janetzko, F. Maurer, und A. Voss. Fallbasiertes Schliessen in Expertensystemen: Welche Rolle spielen Fälle für wissensbasierte Systeme? *KI – Künstliche Intelligenz*, 92(4), December 1992.

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

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

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

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

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

[Fisher *et al.*, 1992] D. Fisher, D. Subramanian, und P. Tadepalli. An overview of current research on knowledge compilation and speedup learning. In P. Tadepalli, (Herausgeber), *Proceedings of the ML'92 Workshop on Knowledge Compilation and Speedup Learning*, University of Aberdeen, Scotland, 1992.

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

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

[Knoblock, 1993] C. Knoblock. *Generating abstraction hierarchies: An automated approach to reducing search in planning.* Kluwer Academic Publishers, 1993.

[Kolodner, 1993] Janet L. Kolodner. *Case-based reasoning.* Morgan Kaufmann, 1993.

[Korf, 1985] R. E. Korf. Macro-operators: A weak method for learning. *Artifical Intelligence*, 26:35–77, 1985.

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

[Lifschitz, 1987] V. Lifschitz. On the semantics of strips. In *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop*, Seiten 1–9, Timberline, Oregon, 1987.

[Michalski und Tecuci, 1991] R.S. Michalski und G. Tecuci, (Herausgeber). *Proceedings of the First International Workshop on Multistrategy Learning (MSL-91)*, Fairfax, VA, George Mason University, 1991.

[Michalski, 1993] R. Michalski. Inferential theory of learning as a conceptual basis for multistrategy learning. *Machine Learning*, (11):111–151, 1993.

[Minton *et al.*, 1989] S. Minton, J. G. Carbonell, C.A. Knoblock, D. R. Kuokka, O. Etzioni, und Y. Gil. Explanation-based learning:a problem solving perspective. *Artificial Intelligence*, 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, und S. T. Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning*, 1(1):47–80, 1986.

[Paulokat und Wess, 1993] Jürgen Paulokat und Stefan Wess. Fallauswahl und fallbasierte Steuerung bei der nichtlinearen hierarchischen Planung. In A. Horz, (Herausgeber), *Beiträge zum 7. Workshop Planen und Konfigurieren*, number 723 in Arbeitspapiere der GMD, Seiten 109–120, 1993.

[Plaza *et al.*, 1993] E. Plaza, A. Aamodt, A. Ram, W. VandeVelde, und M. vanSommeren. Integrated learning architectures. In B. Brazdil, (Herausgeber), *European Conference on Machine Learning: ECML-93*, volume 667 of *Lecture Notes in Artificial Intelligence*, Seiten 429–441, Berlin, 1993. Springer.

[Sacerdoti, 1974] E.D. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5:115–135, 1974.

[Tadepalli, 1991] P. Tadepalli. A formalization of explanation-based macro-operator learning. In Morgan Kaufmann, (Herausgeber), *Proceedings of the International Join Conference on Artifical Intelligence-91*, Seiten 616–622, 1991.

[Tenenberg, 1986] J. Tenenberg. Planning with abstraction. In McDermott, (Herausgeber), *Proceedings of the 6th National Conference on Artifical Intelligence*, Seiten 76–80, Philadelphia, PA, 1986.

[Veloso und Carbonell, 1993] M. M. Veloso und J. G. Carbonell. Towards scaling up machine learning: A case study with derivational analogy in PRODIGY. In Steven Minton, (Herausgeber), *Machine Learning Methods for Planning*, chapter 8, Seiten 233–272. Morgan Kaufmann, 1993.

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

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