Seminar: **Handlungsplanung**

Thema: **Grundlagen und Überblick**

Lehrstuhl : Lehr- und Forschungsgebiet
Informatik V
Prof. Dr. Lakemeyer
RWTH-Aachen

Vorgetragen : 20.10.1997

Verfasser: Dominik Striegler

1. Einleitung

Der folgende Artikel beschreibt die Arbeitsweise von Planungssystemen, möglichen Problemen und Lösungsstrategien. Als erstes müssen grundlegende Definitionen gemacht werden, um die Arbeitsweise eines Planers zu verstehen. Es gibt verschiedene Ansätze Handlungsplanung zu definieren. Ein früher Ansatz lautet:

> Ein Handlungsplaner generiert eine Sequenz von Aktionen.

Diese Definition hat gewisse Nachteile:

1. Pläne sind nur Sequenzen von Aktionen
2. Ergebnisse von Aktionen müssen sicher sein
3. Der Planer verfolgt nur <u>ein</u> Ziel und benötigt einen klar definierten Anfangszustand

Planungssysteme, die diesen drei Richtlinien (Nachteilen) folgen, nennt man klassische Planer. Entwickler gehen von diesen vereinfachten Umständen aus, um mit solchen Planungssystemen Grundwissen über Planungsstrategien zu sammeln. Außerdem können solche vereinfachten Annahmen für gewisse angepaßte Planer ausreichend sein. Wir wollen uns jetzt diese 3 Nachteile(Einschränkungen) genauer ansehen.
Das Problem einer Sequenz von Aktionen ist, daß man einen linearen Plan als Ergebnis bekommt, der auf Veränderungen der Umwelt nicht reagieren kann.
Das Problem von sicheren Aktionen ist, daß die Auswirkung einer Aktion nicht immer vorhersagbar ist.

> z.B.: Ein Roboterfahrzeug soll 10 m nach vorne fahren. Es befindet sich aber auf rutschigem Boden und kommt nur 5 m voran. Der Plan geht aber davon aus, daß 10 m zurückgelegt wurden. Das Fahrzeug wird bei der nächsten Kurve von der Straße abkommen.

Da es nur ein Ziel verfolgen kann, muß der Agent für jedes neue Ziel neu programmiert werden.

Deshalb gibt es neuere Definitionen von Handlungsplanung:

> Handlungsplanung ist automatische Generierung, Fehlerbehebung und Optimierung von Plänen.

Dieser Ansatz ist schon sehr differenziert, es fehlt aber eine Definition von Plänen. Wir wollen nun vereinfacht annehmen, Pläne seien auf irgendeine Weise gespeicherte Informationen, die ein Handlungssystem mit einer Lösung, oder einer Menge von Lösungen versorgt.

2. Notwendige Informationen für ein Planungssystem

- Eine Beschreibung der Welt.
- Einen Zielzustand.
- Eine Auflistung aller möglichen Aktionen.

Optional gehören hierzu noch Zusatzinformationen, sogenannte Einschränkungen (Constraints).

Eine Beschreibung der Welt umfaßt alle Informationen, die der Planer benötigt, z.B. die Zustände seiner Werkzeuge, die Position von Objekten etc.
Der Zielzustand ist ähnlich wie der Anfangszustand eine Beschreibung der Welt, nur diesmal wie sie sein soll, nachdem der Plan ausgeführt wurde. Durch den Unterschied zwischen Anfangs- und Zielzustand erhält der Planer seine Aufgabe.
Die Auflistung aller möglichen Aktionen gibt dem Planer die Werkzeuge (Schritte), mit denen er vom Anfangs- zum Zielzustand kommt. Sie beschreibt alle Möglichgkeiten des Agenten (Roboters) mit seiner Umwelt zu agieren.
Unter den Einschränkungen kann man sich physikalische oder logische Beschränkungen für den Planer vorstellen. Dies kann eine Geschwindigkeitsbeschränkung, Ressourcenbeschränkung oder Ähnliches sein.

Ein kleines Beispiel soll das oben gesagte verdeutlichen:

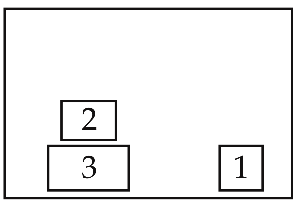


Bild 1

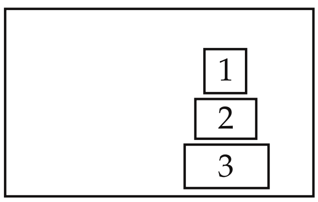


Bild 2

Bild 1 zeigt den Anfangszustand. Um ihn zu beschreiben würde man sagen:

Ontable(1), Ontable(2), Ontable(3)	-	dies bedeutet alle 3 Bausteine sind auf dem Tisch
Clear(1), Clear(2)	-	dies bedeutet Stein 1 und 2 sind frei und können bewegt werden
On(2,3)	-	Stein 2 liegt auf Stein 3
Handempty	-	die Roboterhand ist leer

Bild 2 zeigt den Zielzustand: einen kleinen Turm von Hanoi:

> Ontable(1), Ontable(2), Ontable(3)
> Clear(1)
> On(1,2), On(2,3)

Die Einschränkung hierbei ist, daß kein Stein mit einer größeren Nummer auf einen Stein mit kleinerer Nummer gelegt werden darf. (Siehe Spielregeln vom Türme von Hanoi Problem)

Nun wollen wir uns mit Aktionen beschäftigen. Um ihnen den Aufbau einer Aktion zu beschreiben, benutzen wir ein Beispiel aus dem Planungssystem STRIPS:

Definition der Aktion Pickup(x) unter STRIPS:

Pickup(x)

Precondition: ONTABLE(x) ^
HANDEMPTY ^
CLEAR(x)

Delete List: ONTABLE(x)
HANDEMPTY
CLEAR(x)

Add List: HOLDING(x)

STRIPS arbeitet mit einer Liste von Weltbeschreibungen. Veränderungen werden durch Löschen und Hinzufügen von einzelnen Beschreibungen erreicht.

Es gibt verschiedene Arten einen Plan darzustellen.
Die einfachste Variante ist die von STRIPS. Der Plan besteht hier aus den Änderungen der einzelnen World-States:

Clear(1) On(2,3) Clear(2) Ontable(1) Ontable(2) Ontable(3) Handempty	Pickup(1)	On(2,3) Clear(2) Holding(1) Ontable(2) Ontable(3)	Stack(1,2)	Clear(1) On(1,2) On(2,3) Ontable(3) Handempty

Dies entspricht der Lösung von unserem 1. Beispiel.
Eine andere Möglichkeit besteht aus der partiellen Lösung eines Problems.
Hierbei wird zuerst unser Ziel betrachtet:

On(1,2) ; On(2,3)

Dann wird von diesen Teilschritten eine partielle Lösung erarbeitet:

Pickup(1) - Stack(1,2) ; Pickup(2) - Stack(2,3)

Und als letztes werden die Teillösungen sortiert:

Pickup(2)
Stack(2,3)
Pickup(1)
Stack(1,2)

3. Planung als Suche im Raum der möglichen Pläne

Wenn wir uns die Menge der möglichen Pläne betrachten, stellen wir fest, daß nur wenige zum Ziel führen. Um diese Pläne zu finden setzen wir Suchverfahren ein, die den Suchraum durchsuchen, bis sie eine Lösung finden. Hieraus ergibt sich direkt eine kleine Ableitung der Komplexität. Eine Suche ist um so schwieriger, um so größer der Suchraum ist.
Es eignen sich fast alle Suchverfahren zur Plansuche, wie z.B. Backtracking, Tiefensuche etc. Es ist zu beachten das manche Suchverfahren zu ineffektiv sind oder gar wegen des Platzaufwandes und Zeitaufwandes den Plan nicht finden können. So ist meistens die Breitensuche ungeeignet um Pläne zu suchen, da der Suchraum schon bei wenigen Aktionen sehr groß wird.
Flexiblere Eingaben und weniger Einschränkungen sorgen dafür, daß sich der Suchraum vergrößert. Ist der Suchraum zu groß, kann er wegen der zu geringen Rechnerkapazität unlösbar sein. Deshalb unterscheidet man 2 Arten von Planern:

1. die Domain-Independent Planer. Hierbei wird der Planer mit allgemeinen Lösungstechniken ausgerüstet, er soll für ein möglichst breites Spektrum von Problemen hilfreich sein.

2. die Domain-Dependent oder Domain-Specific Planer. Sie sind speziell auf ein Anwendungsgebiet zugeschnitten. Das bedeutet Suchverfahren, Syntax und Einschränkungen sind speziell auf das Problem zugeschnitten. Dies kann sich im Suchverfahren wiederspiegeln, wo bestimmte Bereiche aus dem Suchraum von

Anfang an ausgeschlossen werden können. z.B. ein Roboter bekommt den Auftrag Waren in ein Regal einzuordnen, alle Pläne bei denen er ohne Ware zum Regal fährt können weggelassen werden.

Deshalb sind Domain-Dependent Planer auf ihr spezielles Problem optimiert, und dort schneller als Domain-Independent Planer. Die Grenze dieser Unterscheidungsmöglichkeit ist aber fließend, ein Domain-Specifik Planer kann auch allgemein anwendbar sein, nur ist er in einem fremden Sachgebiet nicht der geeignetste Planer.

4. Erweiterungen der Suchverfahren

Die jetzt folgenden Erweiterungen sind jeweils nur Teilaspekte. Ein Planungssystem kann viele Erweiterungen in Kombination besitzen.

1) Erweiterung durch partielle Pläne:
   Bei partiellen Plänen haben wir immer einen Skelettplan, der aus Einzelschritten besteht. Es wird eine Liste von Bedingungen erarbeitet, die für jeden Schritt notwendig sind. Es werden Verknüpfungen erstellt, die beschreiben, wie diese Teilschritte zusammengehören. Diese Schritte werden geordnet und anschließend werden die hierdurch entstandenen Teilpläne gelöst. Das Ergebnis eines solchen Verfahrens ist eine geordnete Liste von Aktionen. z.B.:

   > Ein Montageroboter soll einen Rahmen zusammensetzen und anschließend verschrauben.

   Für ihn ist das Zusammensetzen und das Verschrauben jeweils ein Schritt. Er ordnet diese 2 Schritte, erst den Rahmen, dann die Schrauben. Danach erstellt er Teilpläne, wie setzt er den Rahmen zusammen und anschließend, wie schraubt er ihn zusammen.
   Durch das Aufspalten des großen Plans in zwei Teilpläne ist der jeweilige Searchspace kleiner und kann deshalb schneller durchlaufen werden.

2) Verfolgung mehrere Ziele:
   Wir nehmen einmal an, unser Planer bekommt nicht nur ein Ziel genannt, sondern mehrere. Wie soll der Planer mit diesen Zielen umgehen, im Normalfall würde er sie hintereinander ausführen. Dies führt zu 2 Nachteilen:

   – Es kann leicht passieren, daß überflüssige Aktionen ausgeführt werden. z.B. ein Transportroboter, er soll 3 Ladungen von A nach B transportieren. Theoretisch kann er alle Kisten gleichzeitig transportieren, aber er führt die Ziele hintereinander aus und pendelt immer zwischen A und B.

- Es kann zu unlösbaren Problemen führen.
  Wir nehmen das gleiche Beispiel wie oben, nur diesmal ist die Fahrt zu B ein eigenes Ziel. Außerdem reicht der Treibstoff nur noch für eine Fahrt.
  Der Planer bekommt die Teilziele in folgender Reihenfolge:

  Lade Kiste 1, fahre von A nach B, Lade Kiste 2, Lade Kiste 3

  Der Roboter lädt Kiste 1 und bringt sie zu Punkt B. Nun hängt er fest und kann die weiteren Ziele nicht mehr erreichen.

Im folgenden werde ich mich mit 3 bekannten Lösungsansätzen auseinandersetzen.

- Man führt verschiedene Prioritäten ein. Im vorigen Beispiel hieße das, die Aktion Kisten zu laden hätte höhere Priorität als die Aktion Fahre zu X. Also würde derRoboter erst losfahren, wenn er alles geladen hätte.

- Es wird ein Skelettplan eingeführt, in dem die einzelnen Ziele sortiert eingetragen und danach partiell gelöst werden. Im vorigen Beispiel hieße das wiederum, die Kisten zu laden würde im Skelettplan zuerst eingetragen, da sonstnicht mehr alle Ziele erreichbar wären.
- Es gibt eine bessere Variante des Skelettplanes. Hierbei kann der Skelettplan bei auftretenden Problemen wieder geändert werden.

5. Interaktionsprobleme

Diese Lösungsstrategien funktionieren, solange es eine sequenzielle Lösung gibt, die durch Umordnung von Teilzielen (Plänen) gefunden werden kann.
Es ist aber möglich, daß es keine solche Lösung gibt. z.B.:

Ein Roboter soll einen Rahmen verschrauben. Als Ziele bekommt er jede einzelne Verschraubung genannt. Der Roboter hat nur einen Arm und als partielle Lösung für eine Verschraubung hat er die Befehlsreihenfolge:

nimm Schraube, setze Schraube ein, nimm Schraubendreher, drehe Schraube

Nun besteht das Problem darin, daß jede Teillösung so wie eben aufgeführt aussieht. Wenn sie hintereinander gehangen werden, liegt das Problem im Schraubeaufnehmen, da der Roboter den Schraubendreher noch in der Hand hält. Eine Umsortierung der Ziele würde zu keinem Erfolg führen. Diese nennt man Interakitionsprobleme.
Sie können auf verschiedene Art und Weise gehandhabt werden.

- Der Algorithmus erkennt Interaktionen und geht daraufhin in seiner Planung zurück (Backtracking) bis er zu einem Punkt kommt, wo er die Ziele anders

ordnen kann. Dort wählt er nun den anderen Wert. z.B.:
Der Montageroboter plant erst die Schrauben einzusetzen, dann den Rahmen zu verschrauben. Er stellt fest, daß dies zu Interaktionen führt und vertauscht die Ziele.
Nun treten keine Interaktionen mehr auf und er kann den Plan ausführen.

– Der Algorithmus erkennt die problematische Aktion und verschiebt diese an eine andere Stelle. z.B.:

  Unser Montageroboter hat nur eine Hand. Für das Verschrauben sieht seine partielle Lösung wie folgt aus. Schraube nehmen - einsetzen - Schraubendreher nehmen - Schraube festdrehen.

  Leider muß er 10 Schrauben verschrauben. Dies führ zu dem Konflikt, daß der Roboter nach dem ersten Verschrauben den Schraubendreher in der Hand hält. Beim nächsten Verschrauben will er die neue Schraube nehmen, kann dies aber nicht, da er den Schraubendreher hält. Um das Problem zu lösen verschiebt er die Aktionen Schraube nehmen und einsetzen, so daß erst alle Schrauben eingesetzt werden. Anschließend nimmt er den Schraubendreher, um alle vorher eingesetzten Schrauben fest zu drehen.

- Der Algorithmus erkennt problematische Ziele und verschiebt diese.

Es ist zu beachten, daß es nicht immer notwendig ist, alle Interaktionen durch Algorithmen zu vermeiden. Man muß grundsätzlich nur die Interaktionen vermeiden, die bei dem zu lösenden Problem auftreten können.

6. Verschiedene Möglichkeiten wie der entgültige Plan aussehen kann.

Total Order Planer:
Beim Total Order Planer wird der Plan als Sequenz von Aktionen gespeichert. Der Searchspace besteht aus einer Reihe von Sequenzen.

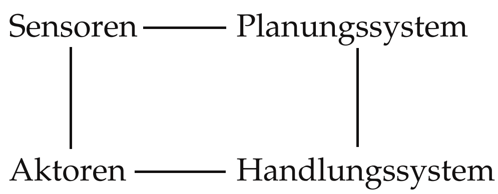
Partial Order Planer:
Hierbei besteht der Plan aus zwei sich abwechselnden Teilen, erstens aus Aktionssequenzen und zweitens aus Aktionsmengen. Sobald eine Aktionsmenge an der Reihe ist, werden alle Aktionen aus der Aktionsmenge in beliebiger Reihenfolge ausgeführt.

7. Planen in einer dynamischen Welt

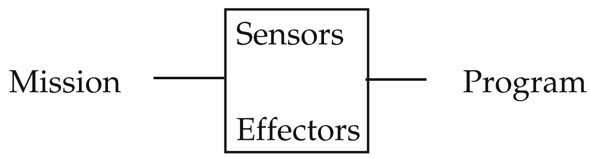
Wir haben den Roboter bisher nur in einer Scheinwelt arbeiten lassen. Agiert er aber in der wirklichen Welt, bekommt er weitere Probleme. Er muß sich auf Veränderungen einstellen können. Das bedeutet, trifft ein Transportroboter auf andere Fahrzeuge, muß er reagieren. Er sollte notfalls einen neuen Weg planen, um nicht den Verkehr zu behindern.

Weitere Probleme ergeben sich durch die Umwandlung von Sensorinformationen in die Beschreibung einer für den Roboter verständlichen Welt. Es muß beachtet werden, welche Informationen für den Roboter wichtig sind und welche unwichtig. Das Planungssystem interessiert es nicht, wie es diese Informationen erhält, aber es ist wichtig mit der Unsicherheit dieser Informationen zu planen.
Eine Möglichkeit mit dieser Unsicherheit umzugehen, besteht darin, den Plan neu zu überdenken, sobald der Planer erfährt, daß etwas anderes passiert als vorhergesehen ist. Dies bedeutet, der Planer macht sich keine Gedanken über die Unsicherheit, sondern er plant bei Problemen einfach neu.
Eine leichte Abwandlung dieser Methode besteht darin, den Planer in gewissen Zeitabständen neu planen zu lassen, und dann den ursprünglichen Plan zu ersetzen. z.B.:

Der Planer eines Transportroboters soll jede Minute seinen Streckenplan überdenken, um auf Unfälle und gesperrte Straßen reagieren zu können.
Um als Entwickler von Planungssystemen an solche Speziallösungen zu denken, hilft folgende Zeichnung:



Dieses Schema zeigt, daß unser Planungssystem nur eins von vielen Systemen ist und sich in das Ganze eingliedern muß. Ein Entwickler sollte seine Planungssysteme nach folgendem Schema an das Problem anpassen:



Erst hat man eine Aufgabe (Mission). Diese Aufgabe hilft uns dabei Sensoren und Aktoren zu wählen. Schließlich wird speziell für diesen Roboter ein Handlungsplaner (Program) entwickelt. Hierdurch entsteht ein Domain-Specific Planer, der auf das Problem zugeschnitten und optimiert worden ist.

Eine weitere Möglichkeit, um mit der dynamischen Umwelt umzugehen, ergibt sich durch die Einführung von Schleifen und Bedingungen. Man verknüpft z.B. Teilpläne durch **If**-Anweisungen. (Wenn der Roboter Objekt x findet kann er es nehmen und den Plan fortführen, andernfalls wird der Alternativplan gewählt.) So erhält man in einfachster Form mehrere Einzelpläne, die miteinander verknüpft sind. Der Roboter kann dann zur Laufzeit zwischen alternativen Plänen wählen.

z.B. Es ist bekannt, daß eine Aktion unsicher ist. Also kann der Planer, immer wenn er diese Aktion einfügen soll, eine Schleife einfügen, die diese Aktion solange wiederholt, bis sie gelingt.

Der Nachteil dieser Lösung ist die große Komplexität, da jede Verzweigung einen neuen kompletten Plan darstellt, der gesucht werden muß. Noch komplexer wird solch ein Plan in der Erstellung, wenn Teilpläne auch wieder zusammenlaufen können.

Ein weiterer spezieller Ansatz, um mit der dynamischen Welt zurechtzukommen, ist der Anytime-Algorithmus.

Der Anytime-Algorithmus soll nach kürzester Zeit den Anfang eines Planes liefern oder eine Vorablösung, deren Qualität unwichtig ist (die aber eine funktionierende Lösung darstellt). Ein Beispiel wo ein Anytime Algorithmus benötigt wird ist z. B. ein Transportroboter im Falle einer gesperrten Straße möglichst schnell seinen Weg fortführen kann und nicht all zu lange die Straße blockiert.
Die Qualität einer so erhaltenen Lösung ist natürlich stark davon abhängig, wie lange der Algorithmus an ihr arbeiten konnte. Hat er mehr Zeit zur Verfügung, kann er den Plan optimieren.
Heutzutage ist die Verwirklichung eines Anytime-Algorithmus nur schwer zu erreichen, da ein Planungssystem unter Zeitdruck falsche Lösungsansätze liefert. Es gibt nur für einige Spezialprobleme Planungssysteme, die in kürzester Zeit den Anfang eines funktionstüchtigen Planes liefern können.

Die Einschränkungen und Anpassungen eines Planers an das jeweilige Anwendungsgebiet spiegeln sich in verschiedenen Bereichen wieder.

- Die Definition der Aktionen (wie z.B. bei STRIPS) ist eine klare Anpassung an das Anwendungsgebiet, da sie speziell auf die Möglichkeiten des Agenten zugeschnitten sind.
- Hierarchische Systeme beinhalten Anpassungen durch die Wahl der Prioritäten, Sie spiegeln die Prioritäten in der Wirklichkeit wieder.
- Manche Systeme sortieren die Teilziele mit einem Algorithmus vor, um diese in den Gesamtplan einzuordnen. (Der Algorithmus basiert auf Erfahrungswerten des

Programmieres und ersetzt dessen Aktive Prioritäten vergabe)
Hierdurch haben wir eine Einschränkung der möglichen Pläne. z.B.:
Unser Transportroboter bekommt die Einschränkung, daß er zuerst alles aufladen soll, und dann erst fahren und abladen soll.

– Zeitfenster für Aktionen sind eine weitere Anpassung vom Planer an das Problem, sie schränken den Suchbereich ein.

8. Zeit und Ressourcenverwaltung

Manchmal ist es notwendig die Zeit und die Ressourcen die eine Aktion braucht, mit einzuberechnen. Hierbei bekommt die Aktion die vorher gemessenen Werte zugewiesen und versucht im Laufe der Optimierung des Plans, möglichst wenig Zeit und Ressourcen zu verbrauchen. Hierdurch hat man die Möglichkeit, Pläne auf ihre Zeiteffizienz zu vergleichen. Also ein guter Ansatz für optimierende Planer. Als Einschränkung kann man Zeitfenster benutzen, die angeben, wann eine bestimmte Aktion spätestens beendet sein muß.

9. Lernen und Speichern

Früh hat man versucht, die Arbeit eines Planers mit dem Planen von Menschen zu vergleichen.
Beim menschlichen Planen kommt es häufig vor, daß alte Pläne hervorgeholt werden und für neue Pläne benutzt werden. Dies hat man auch bei Planern versucht.
Hier gibt es 2 Ansätze:

– Der Planer verwandelt alte Pläne in neue Funktionen, so daß er bei erneutem Planen mehr Aktionen zur Verfügung hat. Das hat den Vorteil, daß bei wiederkehrenden Problemen schnell die neue Aktion ausgeführt wird und man ist schon am Ziel. Der Nachteil besteht im Speicherbedarf und in der immer steigenden Komplexität der Lösungssuche. Dies geschieht durch die erhöhte Anzahl von Aktionen, die den Searchspace vergrößert.
– Eine andere Möglichkeit besteht im Speichern von alten Plänen. Wird ein neuer Plan gesucht, durchsucht der Planer alle alten Pläne und wählt den, in seiner jetzigen Situation passensten Plan aus.
Anschließend wird dieser Plan modifiziert, um ihn an das neue Problem anzupassen. Auch hier entsteht nach kurzer Zeit ein großer Speicherbedarf.

Ein weiteres Problem ist die Effizienz solcher Verfahren. In der Praxis hat es sich gezeigt, daß das Modifizieren von Plänen (oder das Arbeiten mit neuen Aktionen) vom

Aufwand her genauso zu bewerten ist, wie das Neuplanen.

Es ist noch zu sagen, daß das Suchverfahren im Searchspace kein Allheilmittel darstellt. In bestimmten Speziallösungen ist es effizienter, optimierte Algorithmen zu verwenden. Als Beispiel möchte ich hier einfache Bewegungsalgorithemn erwähnen. Sie sind in wenigen Zeilen programmiert und suchen den kürzesten Weg. Hierfür wäre ein Planer ein zu großer Aufwand. Es ist sinnvoller, solche Programme mit Planern zu kombinieren.

Also, noch einmal Zusammenfassend:
Ein Planungssystem braucht eine Beschreibung des Startzustandes, Zielzustandes und aller möglichen Aktionen. Zusätzlich können Einschränkungen den Suchraum verkleinern. Aus der Menge aller möglichen Pläne sucht sich das Planungssystem einen Plan, der zum Ziel führt. Hierbei können Interaktionsprobleme auftreten wenn mehrere Ziele verfolgt werden. Das Ergebnis, der Plan kann entweder in totaler Ordnung vorliegen oder partiell geordnet. Die dynamische Umwelt kann störend auf ein solches Planungssystem einwirken, um dem entgegen zu wirken kann man bei Veränderungen der Umwelt den Planer einen neuen Plan erstellen lassen. Resourcen wirken für ein Planungssystem wie zusätsliche Einschränkungen. Ein Planungssystem kann so programmiert werden, daß er ähnlich wie ein Mensch lernt und alte Pläne wiederverwendet.

Verwendete Literatur:

"A Review of AI Planning Techniques"
von Austine Tate,
James Hendler und
Mark Drummond
1990

"Robot Planning"
von Drew McDermott,
Yale University 1991

"Planning: What it is,
What it could be,
An introduction to the Special Issue on Planning and Scheduling"
von Drew McDermott und
James Hendler,
Yale University 1995

"AI Planning: A Prospectus on theory and applications"
von Subbarao Kambhampati,
Arizona State University

"Planning and Scheduling"
von Thomas Dean,
Brown University und
Subbarao Kambhampati,
Arizona State University