Planen als Erfüllbarkeitsproblem

Seminar Problemlösen und Planen
WS 06/07

Martin Hofmann

1378246

martin.hofmann@stud.uni-bamberg.de

15. Januar 2007

Zusammenfassung

Dieser Seminarbericht gibt einen Einblick in den Planungsansatz, basierend auf aussagenlogischer Erfüllbarkeit. Er zeigt historische Ursprünge zu Deduktivem Planen auf, beschreibt resultierende Weiterentwicklungen zum aktuellen Ansatz und diskutiert verschiedene Kodierungsvarianten und Methoden, um aussagenlogische Formeln zu erfüllen.

1 Einführung

Planen lässt sich im allgemeinen als die Suche einer bestimmten Aktionsfolge zur Erreichung eines konkreten Zieles beschreiben. Deshalb hat sich die Metapher von Zuständen und Zustandsübergängen als sehr anschaulich erwiesen. Es wird hierbei angenommen, dass sich die zu betrachtenden Diskurswelt zu jedem gegebenen Zeitpunkt in einem bestimmten Zustand befindet. Eine Aktion a wird als Übergang von einem Zustand s_1 — in dem a ausführbar ist (d.h. alle Vorbedingung von a gelten) — in einen Zustand s_2, — in dem die Aktion a durchgeführt wurde (d.h. alle Nachbedingungen von a gelten) — angesehen. Ein Plan ist dann eine gültige Folge von Zustandsübergängen von einem Startzustand zu einem Ziel. Als Beschreibungssprache dient dabei eine Logik, in der sich Zustände und Restriktionen der Planungsdomäne als Fakten sowie Aktionen als Implikation zwischen ihren Vorbedingungen und Wirkungen beschreiben lassen.

Anfänglich verstand man Planen als rein deduktiv zu lösendes Problem. Gültige Pläne wurden mit Hilfe logischer Schlusstechniken erstellt, indem ausgehend von einer axiomatisierten Planungsdomäne die Existenz einer Aktionsfolge, die den Anfangszustand in den Endzustand überführt, angenommen und bewiesen wurde.

Obwohl automatisches Theorembeweisen als sehr schwierig gilt, wurde in der KI angenommen, dass das Lösen der Erfüllbarkeit einer Formel weitaus schwerer sei. Zumindest Formeln der Prädikatenlogik 1. Stufe sind bestenfalls semi-entscheidbar, weshalb es keinen vollständigen Algorithmus gibt, der ein Modell einer beliebigen prädikatenlogischen Formel findet. Eine Suche im Raum aller Refutationsbäume einer Klauselmenge zog man daher einer Suche nach einer Variablenbelegung im exponentiell großen Suchraum aller Variablenbelegungen vor.

Sowohl die beschriebenen Schwierigkeiten wie auch das Scheitern dieser Methoden in Realweltszenarien führten allerdings dazu, dass das Generieren von Plänen mit Hilfe logischer Schlusstechniken und Methoden des Theorembeweisens eine Zeit lang kaum mehr beachtet wurde. Dies wurde ebenso dadurch begünstigt, dass die anfänglichen Erfolge des STRIPS Systems gewissermaßen zu einem Paradigmenwechsel geführt hat und Planen hauptsächlich als Suche im Raum der Zustände oder der Teilpläne gesehen wurde.

Erst Selman und Kautz begannen 1991 wieder damit, ermutigt durch ihre erstaunlichen Erfolge beim Lösen des Erfüllbarkeitsproblems für aussagenlogische Formeln[1], Pläne mit Hilfe logischer Schlußtechniken zu erstellen, wobei sie diesmal jedoch Planungsprobleme aussagenlogisch formulierten.

Der Rest des Artikels ist wie folgt aufgebaut: Abschnitt 2 veranschaulicht Greens deduktiv konstruktiven Ansatz durch Resolution in Prädikatenlogik 1. Stufe [Green, 1969] und zeigt anschließend im Abschnitt 3 aufzuzeigen, wie Planen als aussagenlogisches Erfüllbarkeitsproblem verstanden werden kann. In Abschnitt 4 werden dafür die gängigsten Methoden betrachtet, wie dieses Problem gelöst werden kann, und in Abschnitt 5 werden verschiedene Kodierungsvarianten für Planungsdomänen vorgestellt. Der Bericht endet mit einer Zusam-

[1] im Folgenden auch SAT-Problem genannt

menfassung in Abschnitt 6.

2 Planen durch Deduktion

Die ersten Ansätze des Planens durch Deduktion, basierend auf dem von McCarthy [McCarthy, 1983] beeinflussten *situation calculus*, gehen zurück auf [Green, 1969] von Cordell Green. Dabei bilden Prädikatenlogik 1. Stufe und deren Resolutionskalkül die Grundlage für das Schließen in einer axiomatisierten Planungsdomäne.

In einem konkreten Planungsproblem werden Aussagen über Zustände wie zum Beispiel den Startzustand durch Prädikate als atomare Fakten (sog. *fluents*)[2] formuliert. Beispielsweise ist in einer Blockwelt

$$auf(b_1, b_2, s_0)$$

wahr, wenn im Zustand s_0 der Block b_1 auf dem Block b_2 steht. Des weiteren stellen Aktionen Funktionen dar, die einen Zustand auf einen anderen Zustand abbilden. Die Tatsache, dass Aktionen nur ausgeführt werden können, wenn bestimmte Bedingungen gelten, wird durch eine Implikation zwischen den Vorbedingungen der Aktion und deren Effekten dargestellt. Folgendes Axiom beschreibt, dass in einem Zustand S ein Block A nur dann von einem Block C auf einen Block B gehoben werden kann, wenn beide frei sind, d.h. keine anderen Blöcke auf ihnen liegen.

$$\begin{aligned} \forall A, B, S. \quad & frei(A, S) \wedge frei(B, S) \wedge auf(A, C, S) \rightarrow \\ & auf(A, B, hebe(A, B, S)) \wedge frei(C, hebe(A, B, S)) \end{aligned} \tag{1}$$

Die Funktion $hebe(A, B, S)$ entspricht hierbei der Aktion, A auf B zu heben, wobei C frei wird. Frame Axiome, wie beispielsweise

$$\begin{aligned} \forall A, B, S. \quad & frei(A, S) \wedge frei(B, S) \wedge auf(A, C, S) \rightarrow \\ & frei(A, hebe(A, B, S)) \end{aligned} \tag{2}$$

$$\begin{aligned} \forall A, B, S. \quad & frei(A, S) \wedge frei(B, S) \wedge auf(A, C, S) \wedge frei(C, S) \rightarrow \\ & frei(C, hebe(A, B, S)) \end{aligned} \tag{3}$$

beschreiben, welche Auswirkungen Aktionen *nicht* haben. Zum Beispiel bleibt ein Block A ebenso wie ein an der Aktion unbeteiligter Block C frei, nachdem A auf B gehoben wurde. Ähnlich ließe sich mit der Konstanten $tisch$ auch ein spezieller Block beschreiben, auf dem beliebig viele Blöcke liegen können, er also stets "frei" ist.

Über Resolution sind nun Schlüsse in diesem Planungsproblem möglich. Vermutungen wie $\exists X.auf(b_1, X, s_0)$ ("Auf welchem anderen Block befindet sich der Block b_1 im Zustand s_0") oder $\exists S.auf(b_2, b_1 S)$ ("Gibt es einen Zustand in dem b_2 auf b_1 liegt?") werden negiert und zusammen mit einem Prädikat $Plan(S)$ als Klausel der Datenbasis hinzugefügt und durch Resolution bewiesen oder widerlegt. Die Variable S soll dabei während der Resolution mit

[2] *fluents* und Aussagen werden synonym verwendet, sofern es aus dem Kontext ersichtlich ist

dem konstruierten Plan unifizieren. Dies kann als die Aussage interpretiert werden, dass entweder die Vermutung falsch oder der Plan S ist. Beispielsweise wäre aus der Vermutung

$$\neg auf(b_2, b_1, S) \vee Plan(S)$$

der Plan

$$Plan(hebe(b_2, b_1, hebe(b_1, tisch, s_0)))$$

ableitbar.

Problematisch ist hier allerdings, da es sich um Prädikatenlogik handelt, dass grundsätzlich Zustandsterme unendlicher Länge bei wiederholter Ausführung von Aktionen möglich sind. Dadurch wird der Suchraum unnötig vergrößert, jedoch sind Pläne in der Regel endlich.

3 Planen als aussagenlogisches Erfüllbarkeitsproblem

Im traditionellen deduktiven Ansatz wird ein Planungsproblem als Theorem beschrieben, das aussagt, dass die Anfangsbedingungen zusammen mit einer Folge von Aktionen den Endzustand implizieren. Im Ansatz des Planens als Erfüllbarkeitsproblem nach Kautz und Selman [Selman, 1992, Kautz and Selman, 1996] ist es hingegen eine Menge aussagenlogischer Axiome mit der Eigenschaft, dass *jedes* Modell dieser Formeln einem gültigen Plan entspricht. Durch Axiomatisierung in aussagenlogischen Formeln wird einerseits das Problem der Unentscheidbarkeit der Prädikatenlogik 1. Stufe umgangen und andererseits werden lediglich endliche Pläne, wenn auch mit unbestimmter Länge, betrachtet. Hierzu erhält jedes Prädikat anstelle der Zustandsvariable eine Variable I, $I \in \mathbb{N}$, $I \leq n$ vom Typ *Zeit*, die aussagt, zu welchem Zeitpunkt eine bestimmte Aussage wahr ist. Hierbei sei n die maximale Länge des Planes. Das Festlegen auf eine konkrete Planlänge (boundend length planning) stellt dabei keine wesentliche Einschränkung dar, da während der Suche die Länge iterativ vergrößert werden kann (iterative deepening).

Analog der üblichen Schreibweise lassen sich die vorher erwähnten Axiome nun auch in Aussagenlogik ausdrücken:

$$\begin{aligned} \forall A, B, I. \quad & frei(A, I) \wedge frei(B, I) \wedge auf(A, C, I) \wedge hebe(A, B, I) \rightarrow \\ & auf(A, B, I + 1) \wedge frei(C, I + 1). \end{aligned} \tag{4}$$

Allerdings ist zu beachten, dass Aktionen und nun nicht mehr als Terme oder Funktionen repräsentiert werden, sondern als Aussagen, denen die Wahrheitswerte $wahr$ und $falsch$ zugewiesen werden können. Ein Planungsproblem könnte nun wie folgt formuliert werden:

$$\begin{aligned} & \exists X_1, Y_1, X_2, Y_2. \\ & \quad auf(b_1, b_2, 1) \wedge auf(b_2, tisch, 1) \wedge \\ & \quad frei(b_1, 1) \wedge hebe(X_1, Y_1, 1) \wedge hebe(X_2, Y_2, 2) \rightarrow \\ & \qquad\qquad auf(b_2, b_1, 3) \end{aligned}$$

Diese Formel wäre durch eine Instantiierung der Variablen zu beweisen, die sie erfüllt. Ein Plan entspräche dann den beiden korrekt instantiierten Prädikaten $hebe$.

Problematisch ist jedoch, dass alle bisher erwähnten Axiome zwar in der idealisierten Blockwelt $wahr$ sind, jedoch trifft es nicht zu, dass alle Welten in denen diese Axiome gültig sind (d.h. alle ihre Modelle) auch wirklich einer Blockwelt entsprechen. Zwar ist alles, was aus ihnen geschlossen werden kann, wahr, aber nicht alle Pläne, die konsistent mit diesen Axiomen sind, sind auch sinnvoll.

Ein Modell einer Formel ist eine Belegung der atomaren Aussagen, die diese zu $wahr$ auswerten lässt. Beispielsweise wäre

$$\left\{ \begin{array}{l} auf(b_1, b_2, 1), auf(b_2, tisch, 1), frei(b_1, 1), \\ auf(b_2, b_1, 2), auf(b_2, b_1, 3), frei(b_2, 2), \\ frei(b_2, 3) \end{array} \right\}$$

ein korrektes Modell unter den oben beschriebenen Axiomen. Allerdings beschreibt dieses Modell eine Welt, die sich ändert, ohne dass eine Aktion ausgeführt wird. Ebenso ist

$$\left\{ \begin{array}{l} auf(b_1, b_2, 1), auf(b_2, tisch, 1), frei(b_1, 1), \\ hebe(b_2, b_1, 1), auf(b_2, b_1, 2), auf(b_2, b_1, 3) \end{array} \right\}$$

ein gültiges Modell. Nach dem klassischen *ex falso sequitur quodlibet* kann aus einer verletzten Prämisse beliebiges gefolgert werden. Daher ist es konsistent mit den Axiomen, dass der Effekt $auf(b_2, b_1, 3)$ auftritt, obwohl die Vorbedingungen der Aktion $hebe(b_2, b_1, 1)$ verletzt sind.

Diese unerwünschten Modelle lassen sich durch Axiome ausschließen, in denen eine Aktion ihre Vorbedingungen und Wirkungen impliziert [Selman, 1992].

$$\begin{aligned} \forall A, B, I.\ hebe(A, B, I) \quad \rightarrow \quad & frei(A, I) \wedge frei(B, I) \wedge auf(A, C, I) \wedge \\ & auf(A, B, I + 1) \wedge frei(C, I + 1) \wedge frei(A, I + 1) \end{aligned} \quad (5)$$

Sogenannte *complete exclusion* Axiome (vgl. Abschnitt 5) stellen sicher, dass nie zwei Aktionen gleichzeitig ausgeführt werden.

$$\forall A, A', B, B', I.\ (A \neq A' \vee B \neq B') \quad \rightarrow \quad \neg hebe(A, B, I) \vee \neg hebe(A', B', I) \quad (6)$$

Das Axiom *at-least-one* (vgl. Abschnitt 5) sorgt dafür, dass zu jedem Zeitpunkt mindestens eine Aktion ausgeführt wird. Es sei n wieder die maximale Planlänge.

$$\forall I < n. \exists A, B. hebe(A, B, I) \quad (7)$$

Wird jedoch zum Zeitpunkt i keine Aktion durchgeführt, so kann keine Aussage über den Wahrheitswert der *fluents* zum Zeitpunkt $i + 1$ gemacht werden und diese können daher beliebige Werte annehmen. Schließlich wird der Startzustand noch vollständig durch seine *fluents* beschrieben, d.h. alle Aussagen, die zu Beginn gelten, sowie alle, die *nicht* gelten[3] [Kautz et al., 1996], werden konjunktiv verknüpft.

$$\begin{aligned} & frei(b_1, 1) \wedge \neg frei(b_2, 1) \wedge frei(tisch, 1) \wedge auf(b_1, b_2, 1) \wedge \\ & \neg auf(b_2, b_1, 1) \wedge \neg auf(b_1, tisch, 1) \wedge auf(b_2, tisch, 1) \end{aligned} \quad (8)$$

[3] d.h. im Startzustand gilt explizit die Closed World Assumption

Der Zielzustand dagegen wird lediglich partiell durch diejenigen Aussagen beschrieben, die in ihm gelten:

$$auf(b_2, b_1, 3) \tag{9}$$

Die Axiome[4] 4 - 9 spezifizieren nun das Planungsproblem vollständig, das von einem beliebigen Algorithmus für aussagenlogische Erfüllbarkeit gelöst werden kann. Dabei expandieren allquantifizierte Formeln zur Konjunktion ihrer Instanzen (mögliche Variablenbelegungen) und existenzquantifizierte Formeln zur Disjunktion ihrer Instanzen. Der daraus resultierende Plan ist eine vollständig geordnete Sequenz von Aktionen.

4 Techniken zur aussagenlogischen Erfüllbarkeit

Der folgende Abschnitt diskutiert Algorithmen, die zur Erfüllbarkeit aussagenlogischer Formeln verwendet werden. Man unterscheidet hierbei zwischen vollständigen und stochastischen Methoden.

4.1 Systematische Erfüllbarkeitstechniken

Systematische Erfüllbarkeitstechniken sind vollständig und korrekt und basieren in der Regel auf der Davis-Putnam Methode [Cook and Mitchell, 1997]. Dabei wird die Vollständigkeit der Resolution bezüglich Widerspruchs ausgenutzt, um alle Resolventen einer Klauselmenge zu generieren und schließlich zu testen, ob die “leere” Klausel generiert wurde.

In jedem Schritt wählt eine Heuristik eine Variable aus. Sind in der Klauselmenge Einheitsklauseln oder Variablen enthalten, die entweder nur als positives oder nur als negatives Literal vorkommen, so werden diese zuerst ausgewählt. Die ausgewählte Variable wird aus der Klauselmenge eliminiert, indem alle Resolventen mit diesem Literal gebildet und anschließend alle Klauseln gelöscht werden, in denen dieses Literal noch vorkommt.

Gibt es weder Einheitskauseln noch Einheitsliterale, so wird eine Variable gemäß der Heuristik ausgewählt. Dabei wird das Problem auf zwei kleinere Teilprobleme reduziert – eines für jeden Wahrheitswert der Variablen – und in ihnen das jeweilige Literal eliminiert und Davis-Putnam rekursiv aufgerufen, bis entweder die “leere” Klausel generiert wurde oder ein Widerspruch gefunden wurde. Der Davis-Putnam Algorithmus (Abb. 1) vollführt dabei eine Tiefensuche mit backtracking durch den Raum aller möglichen (partiellen) Wahrheitswertbelegungen.

Für die Auswahl der zu eliminierenden Variablen ($\star$ in Abb. 1) lassen sich in der Literatur die verschiedensten Heuristiken finden. Ursprünglich wurde die erste Variable in der ersten Klausel mit minimaler Länge gewählt, aber auch komplexere Heuristiken sind denkbar [Cook and Mitchell, 1997].

[4] Das Beispiel entspricht einer regulären Kodierung mit klassischen Frame Axiomen und vollständigem Ausschluss (vgl. Abschnitt 5), in [Kautz et al., 1996, Kautz and Selman, 1996] auch “linear” genannt

```
Davis-Putnam(Φ, μ)
    if ∅ ∈ Φ then return μ ← ∅
    else if Φ = ∅ then return μ
    Elim(Φ, μ)
(⋆) wähle Variable v ∈ Φ
    Davis-Putnam(Φ ∪ v, μ)
    Davis-Putnam(Φ ∪ ¬v, μ)
end

Elim(Φ, μ)
  while Einheitsklausel oder Einheitsliteral {l} ∈ Φ do
    μ ← μ ∪ {l}
    foreach Klausel C ∈ Φ do
      if l ∈ C then Φ ← Φ − {C}
      else if ¬l ∈ C then Φ ← Φ − {C} ∪ {C − ¬{l}}
    end
  end
end
```

Abbildung 1: Davis-Putnam Algorithmus

Der Algorithmus belegt alle Variablen sukzessive mit Wahrheitswerten. Die Gesamtformel wird schrittweise vereinfacht, indem Literalen in Einheitsklauseln der Wahrheitswert $wahr$ zugewiesen wird und jene Literale aus der Formelmenge löscht werden, die in der aktuellen Belegung falsch sind. Gegebenenfalls wird backtracking angewendet.

Beispielsweise wurde, ausgehend von dem Axiom (5) im vorhergegangenen Abschnitt 3 während des Instantiierungsprozesses u. a. folgende Klausel erzeugt:

$$\begin{aligned} hebe(b_2, b_1, 1) \quad \rightarrow \quad & frei(b_2, 1) \wedge frei(b_1, 1) \wedge auf(b_2, tisch, 1) \wedge \\ & auf(b_2, b_1, 2) \wedge frei(tisch, 2) \end{aligned} \tag{10}$$

Aus dem Startzustand (8) folgt die Einheitsklausel $\neg frei(b_2, 1)$, weshalb in der ersten Iteration der $Elim$-Subprozedur die Klausel (10) zu $\neg hebe(b_2, b_1, 1)$ vereinfacht wird und in der zweiten $hebe(b_2, b_1, 1)$ der Wert $falsch$ zugewiesen wird. Dies entspricht eben der Tatsache, dass $hebe(b_2, b_1, 1)$ im ersten Schritt nicht durchführbar ist. Hierbei wird deutlich, dass während des initialen Instantiierungsprozesses zwar sehr viele "unmögliche" Klauseln erzeugt werden, gerade diese aber sukzessive genutzt werden, um die korrekte Wahrheitswertbelegung der Gesamtformel zu finden.

Die Unerfüllbarkeit aller Axiome durch ein Modell kann aber auch der Unmöglichkeit eines Zustandes zugeschrieben werden, zum Beispiel bei Zuständen, in denen ein Block auf sich selber liegt. Logisch ist es nicht notwendig, solche Zustände durch zusätzliche Axiome auszuschließen. Allerdings hat es sich gezeigt, dass dadurch die Laufzeit des Algorithmus verringert werden kann [Selman, 1992, Selman and Kautz, 1993].

4.2 Stochastische Erfüllbarkeitstechniken

Im, Gegensatz zu den systematischen Methoden sind die stochastischen unvollständig. Zwar können sie die Unerfüllbarkeit einer Formel nicht beweisen, sind aber in der Regel wesentlich schneller im Auffinden einer erfüllenden Belegung, sofern sie existiert.

Unvollständige Methoden sind in der Regel Varianten einer "greedy local search". Über Wahrheitswertbelegungen ist eine Kostenfunktion definiert (z.B. Anzahl der nicht erfüllten

Klauseln), so dass ein globales Minimum einer erfüllenden Belegung entspricht. Ausgehend von einer initialen, meist geratenen Belegung versucht man schrittweise diese bezüglich der Kostenfunktion zu verbessern, indem man sie mit Belegungen in der Nachbarschaft (z.B. bezüglich der Hamming-Distanz) vergleicht. Selmans GSAT [Selman et al., 1992] basiert gerade auf diesen Prinzipien. In jedem Schritt wird die Variable "umgedreht", die zur besten Belegung in der Nachbarschaft führt. Entgegen dem klassischen Verständnis von "local search" geschieht dies allerdings auch, wenn dies die aktuelle Belegung verschlechtert. Da unvollständige Methoden *a priori* keine inhärente Abbruchbedingung haben, beginnt GSAT die Suche nach einer maximalen Anzahl von "flips"neu und terminiert nach einer maximalen Anzahl von Versuchen.

Ein weiterer Ansatz geht ähnlich beim Finden eines Modells vor. "Iterative repair" versucht in jedem Schritt gezielt eine bisher unerfüllte Teilklausel zu "reparieren" und verändert dabei den Wahrheitswert einer der darin enthaltenen Variablen. Die einfachste Variante RANDOM WALK wählt dabei zufällig eine Variable aus einer bisher noch unerfüllten, ebenfalls zufällig gewählten Teilklausel und dreht den Wahrheitswert um. Selman, Kautz und Cohen [Selman et al., 1994] entwickelten eine probabilistische "greedy" Variante von RANDOM WALK, die sich als effektiver als GSAT und andere "local search" Methoden erwiesen hat [Kautz and Selman, 1996]. Dieser Algorithmus, WALKSAT genannt, wählt zufällig eine unerfüllte Klausel aus und dreht mit einer Wahrscheinlichkeit p eine zufällig gewählte Variable um, oder mit der Wahrscheinlichkeit $1 - p$ diejenige, die gemäß einer Kostenfunktion die Kosten minimiert.

4.3 Bewertung

Prinzipiell kann keinem der beiden Ansätze, systematisch oder stochastisch, der absolute Vorzug gegeben werden. Vielmehr zeigten Kautz und Selman [Kautz and Selman, 1996], dass sie sich sehr gut ergänzen können. Systematische Verfahren können genutzt werden, um generell die unterste Grenze der Planlänge zu finden und stochastische, um dann den eigentlichen Plan zu erstellen. Interessanterweise waren nämlich systematische Algorithmen in einigen Fällen beim Beweis der Nichtexistenz eines Planens mit gegebener Länge erfolgreicher, als beim Finden einer Lösung für ein vergleichbares Problem. Andererseits sind stochastische Methoden nur dann vorzuziehen, wenn die Unvollständigkeit ein akzeptabler Preis für die Geschwindigkeit des Verfahrens ist.

5 Kodierung

Dieser Abschnitt versucht die gängigsten Kodierungsverfahren zu systematisieren, beschreibt die unterschiedlichen Aktionskodierungen sowie Varianten der Frame Axiome und schließt mit einer zusammenfassenden Bewertung.

5.1 Basisaxiome

Folgende Axiome sind allen Kodierungsvarianten gemein [Ghallab et al., 2004].

1. Der **Startzustand** s_0 wird vollständig als Konjunktion aller geltenden und der Negation der nicht geltenden Aussagen f beschrieben, d.h. es gilt explizit die *Closed World Asumption* [Kautz et al., 1996]:

$$\bigwedge_{f \in s_0} f_0 \wedge \bigwedge_{f \notin s_0} \neg f_0 \tag{11}$$

2. Der **Endzustand** g wird analog zum Startzustand beschrieben, kann aber wahlweise entweder partiell als Konjunktion der geforderten Aussagen f oder vollständig beschrieben werden [Kautz et al., 1996]:

$$\bigwedge_{f \in g} f_n \wedge \bigwedge_{f \notin g} \neg f_n \tag{12}$$

3. Eine **Aktion** a impliziert ihre Vorbedingungen P_a und Wirkungen E_a. Für alle Aktionen $a \in A$ und für jeden Zeitpunkt i, $0 \leq i \leq n-1$ gilt:

$$a_i \rightarrow \left(\bigwedge_{p \in P_a} p_i \wedge \bigwedge_{e \in E_a} e_{n+1} \right) \tag{13}$$

   Hierbei deutet der Index i an, dass das jeweilige Atom für den Zeitpunkt i instantiiert ist. Des weiteren gilt für $e \in E_a, E_a = E_a^- \cup E_a^+, E_a^- \cap E_a^+ = \emptyset$, dass e analog der STRIPS-Notation entweder ein positives, falls es sich um einen ADD-Effekt handelt ($e \in E_a^+$) oder ein negatives Literal, im Falle eines DEL-Effektes ($e \in E_a^-$) ist.

5.2 Frame Axiome

Frame Axiome beschränken Aussagen über Zustände, die von Aktionen unberührt bleiben. Dabei gibt es die Möglichkeit, diese entweder als Klassische Frame Axiome (a) oder Erklärende Frame Axiome (b) zu formulieren.

4a. **Klassische Frame Axiome** beschreiben, welche Aussagen bei gegebenen Aktionen unverändert bleiben, das heißt, welche Auswirkungen eine Aktion nicht hat[5]. Für jede Aktion a, jede Aussage $f \notin E_a$ und für jeden Zeitpunkt i, $0 \leq i \leq n-1$ gilt:

$$f_i \wedge a_i \rightarrow f_{i+1} \tag{14}$$

[5] Eine vollständige Beschreibung des Zielzustandes kann jedoch zu einem Effizienzgewinn führen, da die Variablenelimination nun sowohl vom Startzustand als auch vom Zielzustand aus durchgeführt werden kann. Dies entspräche quasi einer Vorwärts- und Rückwärtssuche.

5a. Problematisch dabei ist allerdings, falls eine Aktion in einem Schritt nicht ausgeführt wird, dass Formel (14) trivialerweise wahr wird. Um dies zu verhindern stellt das ***at-least-one*** **Axiom** sicher, das zu jedem Zeitpunkt i, $0 \leq i \leq n-1$ mindestens eine Aktion ausgeführt wird:

$$\bigvee_{a \in A} a_i \tag{15}$$

Dies entspricht der Kodierung des Beispiels in Abschnitt 3, aber ohne vollständigen Ausschluss. Dieser ist überflüssig, da die Kombination der Formeln (13, 14) schon sicher stellt, dass zwei zum Zeitpunkt t ausgeführte Aktionen zum selben Zustand im Zeitpunkt $t+1$ führen [6]. Falls also mehr als eine Aktion zu einem Zeitpunkt auftritt, kann eine von beiden ausgeführt werden [Ernst et al., 1997].

4b. Im Gegensatz dazu zählen **Erklärende** (explanatory) **Frame Axiome** diejenigen Aktionen auf, die für eine Zustandsänderung verantwortlich gewesen sein könnten.

$$\begin{aligned} \neg f_i \wedge f_{i+1} &\rightarrow \left(\bigvee_{a \in A | f_i \in E_a^+} a_i \right) \\ f_i \wedge \neg f_{i+1} &\rightarrow \left(\bigvee_{a \in A | f_i \in E_a^-} a_i \right) \end{aligned} \tag{16}$$

Da Erklärende Frame Axiome fordern, dass eine Änderung der *fluents* auch eine Aktion impliziert, bedeutet dies, dass keine Aktion durchgeführt wurde, wenn *fluents* sich nicht änderten. Dies entspricht implizit einer "no-op"-Aktion, die *at-least-one* Axiome überflüssig macht [Kautz et al., 1996].

5b. Erklärende Frame Axiome erlauben Parallelismus, da sie nicht zwangsläufig fordern, dass *fluents*, die nicht von einer Aktion betroffen sind unverändert bleiben. Prinzipiell können alle Aktionen, deren Vorbedingungen zum Zeitpunkt t gelten und deren Effekte sich nicht gegenseitig aufheben, können parallel ausgeführt werden. Dabei kann es aber sein, dass derartig erzeugte parallele Pläne keine lineare Lösung haben. Beispielsweise können eine Aktion α mit Vorbedingung X und Effekt $\neg Y$ und eine Aktion β mit Vorbedingung Y und Effekt $\neg X$ zwar parallel, aber nie sequentiell ausgeführt werden. Dies kann durch **vollständigen Ausschluss** (total exclusion) für alle $a_i, b_i \in A$ und $0 \leq i \leq n-1$ erreicht werden:

$$\neg a_i \vee \neg b_i \tag{17}$$

[6] sie haben dieselben Vorbedingungen, ADD- und DEL-Effekte

Das Ergebnis ist hierbei wieder ein vollständig geordneter Plan, da zu jedem Zeitpunkt genau eine (oder keine) Aktion durchgeführt wird.

Der vollständige Ausschluss kann dadurch abgeschwächt werden, dass er nur auf solche Aktionen beschränkt wird, deren Vorbedingungen und Effekte sich widersprechen. Das Resultat ist ein partiell geordneter Plan mit der Eigenschaft, dass jeder vollständig geordnete Plan, der konsistent mit dieser partiellen Ordnung ist, einen gültigen Plan beschreibt. Man spricht hierbei von **konfliktärem Ausschluss** oder conflict exclusion [Ernst et al., 1997].

5.3 Repräsentation von Aktionen

Bei der regulären Repräsentation wird jede atomare Aktion durch eine eigene logische Variable dargestellt mit insgesamt $A = n|Ops||Dom|^{A_o}$. Hierbei sei A die Anzahl der atomaren Aktionen, n die Planlänge, $|Ops|$ die Anzahl der Operatoren, $|Dom|$ die Anzahl der Konstanten der Planungsdomäne und A_o die maximale Stelligkeit der Operatoren.

Da die Laufzeit von systematischen Lösungsverfahren (s. Abschnitt 4) exponentiell abhängig von der Anzahl der Variablen ist und eine große Anzahl von Variablen auch stochastische Verfahren (s. Abschnitt 4) verlangsamt, gibt es verschiedene Ansätze, diese möglichst gering zu halten.

In der **regulären** Repräsentation ist die gesamte Anzahl der Variablen exponentiell abhängig von der maximalen Stelligkeit der Operatoren A_o. Um dies zu umgehen ersetzten Kautz und Selman in [Kautz and Selman, 1996] jeden n-stelligen Operator durch n einstellige Operatoren. Aus $hebe(A, B, t)$ wird so $hebeArg1(A, t)$ und $hebeArg2(b, t)$. Durch diese sogenannte einfache **"operator splitting"** kann die Anzahl der Variablen auf $A = n|Ops||Dom|A_o$ reduziert werden.

Lässt man zusätzlich zu, dass Operatoren sich *fluents* teilen, gelangt man zum **"overloaded operator splitting"**. Beispielsweise wird $hebe(A, B, t)$ durch die Konjunktion von $Act(hebe, t)$, $Arg1(A, t)$ und $Arg2(b, t)$ ersetzt und eine andere Aktion $faerbe(A, Rot, t)$ durch $Act(faerbe, t)$, $Arg1(A, t)$ und $Arg2(Rot, t)$ repräsentiert wird. Dadurch kann die Anzahl der verwendeten Variablen noch einmal auf $A = n(|Ops| + |Dom|A_o)$ reduziert werden [Ernst et al., 1997].

Repräsentiert man die Aktionen durch **Bits**, sind lediglich $\lceil \log_1 A \rceil$ aussagenlogische Variablen nötig. Bei beispielsweise vier Aktionen zum Zeitpunkt t wird die erste durch $\neg bit(t) \wedge \neg bit(t)$, die zweite durch $\neg bit(t) \wedge bit(t)$ usw. dargestellt.

5.4 Bewertung

Allerdings ist der Umfang der Kodierung nicht allein von der Anzahl der Variablen, sondern auch von der Anzahl der Klauseln abhängig. Zusätzlich konnten Ernst et al. in [Ernst et al., 1997] zeigen, dass sich ihre theoretischen Komplexitätsberechnungen für die Anzahl der Variablen und Klauseln in der Praxis oft nicht bestätigten. Entgegen diesen theoreti-

schen Überlegungen zeigte sich, dass reguläre und "simple splitting" Kodierungen mit erklärenden Frame Axiomen die kleinsten sind.

Der Vorteil der erklärenden Frame Axiomen gegenüber den klassischen liegt darin, dass sie nur beschreiben, was sich ändert und nicht, was sich nicht ändert, wenn eine Aktion durchgeführt wird. Eine reguläre erklärende Kodierung erlaubt zusätzlich parallele Aktionen, was selbstverständlich die Planlänge reduziert. Außerdem ist konfliktärer Ausschluss als echte Teilmenge des vollständigen Ausschlusses anzusehen.

Dagegen führten Kodierungen mit einer vergleichsweise geringen Anzahl von Variablen ("splitting" Strategien und bitweise Kodierung) zu einer Explosion der Klauselmenge. Besonders die *at-least-one* Axiome, eine Disjunktion aller voll instantiierten Aktionen, führen bei der Konvertierung in CNF zu einer exponentiellen Explosion der Klauselzahl.

Abschließend sei außerdem anzumerken, dass, obwohl die Kodierungen alle auf Spezifikationen im STRIPS-Stil basieren, grundsätzlich aber auch beliebige Restriktionen in Klauselform möglich sind, die in dieser Form in STRIPS nicht ausdrückbar sind. Beispielsweise werden in den meisten Planern im STRIPS-Stil auf und $frei$ separat behandelt [Kautz and Selman, 1996], im Ansatz des Planens als Erfüllbarkeitsproblem ließe sich das dagegen einfach durch folgende Äquivalenz beschreiben:

$$\forall A, B, I.frei(A, I) \equiv \neg\exists B.auf(B, A, I).$$

Für eine detailliertere Diskussion der unterschiedlichen Kodierung und zusätzlicher Optimierungsmöglichkeiten sei auf [Ernst et al., 1997] und [Ghallab et al., 2004] verwiesen.

6 Fazit

Dieser Bericht gab Einblicke in den Planungsansatz durch Erfüllbarkeit, zeigte die historischen Ursprünge auf und diskutierte verschiedene Lösungs- und Kodierungsverfahren.

Kautz und Selman legten dar, wie aussagenlogische Kodierungen schwieriger STRIPS-Planungsprobleme mit Algorithmen zur aussagenlogischen Erfüllbarkeit gelöst werden können. Dabei wurden die Axiome des Deduktiven Planens erweitert, um fehlerhafte Modelle zu vermeiden. Entgegen der sich lange gehaltenen Auffassung sind zum Lösen von Planungsproblemen nicht zwangsläufig spezialisierte Algorithmen notwendig. Vielmehr stellte sich heraus, dass zur Zeit derartige Planer im STRIPS-Stil die schnellsten ihrer Art sind.

Literatur

[Cook and Mitchell, 1997] Cook, S. A. and Mitchell, D. G. (1997). Finding hard instances of the satisfiability problem: A survey. In Du, Gu, and Pardalos, editors, *Satisfiability Problem: Theory and Applications*, volume 35, pages 1–17. American Mathematical Society.

[Ernst et al., 1997] Ernst, M., Millstein, T. D., and Weld, D. S. (1997). Automatic SAT-compilation of planning problems. In *IJCAI*, pages 1169–1177.

[Ghallab et al., 2004] Ghallab, M., Nau, D., and Traverso, P. (2004). Automated planning : Theory & practice.

[Green, 1969] Green, C. C. (1969). Application of theorem proving to problem solving. In *IJCAI*, pages 219–240.

[Kautz and Selman, 1996] Kautz, H. and Selman, B. (1996). Pushing the envelope: Planning, propositional logic, and stochastic search.

In Shrobe, H. and Senator, T., editors, *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, pages 1194–1201, Menlo Park, California. AAAI Press.

[Kautz et al., 1996] Kautz, H. A., McAllester, D., and Selman, B. (1996). Encoding plans in propositional logic. In *Proceedings of the Fifth International Conference on the Principle of Knowledge Representation and Reasoning (KR'96)*, pages 374–384.

[McCarthy, 1983] McCarthy, J. (1983). Situations, actions, and causal laws. Technical Report Memo 2, Stanford Artificial Intelligence Project, Stanford University.

[Selman and Kautz, 1993] Selman, B. and Kautz, H. A. (1993). Domain-independent extensions to GSAT: solving large structured satisfiability problems. In *Proceedings of theInternational Joint Conference on Artificial Intelligence(IJCAI-93)*, Chambéry, France.

[Selman et al., 1994] Selman, B., Kautz, H. A., and Cohen, B. (1994). Noise strategies for improving local search. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI'94)*, pages 337–343, Seattle.

[Selman et al., 1992] Selman, B., Levesque, H. J., and Mitchell, D. (1992). A new method for solving hard satisfiability problems. In Rosenbloom, P. and Szolovits, P., editors, *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 440–446, Menlo Park, California. AAAI Press.

[Selman, 1992] Selman, H. A. K. . B. (1992). Planning as satisfiability. In *Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI'92)*, pages 359–363.