Die Ausdrucksstärke von Planungsformalismen: Eine formale Charakterisierung

Bernhard Nebel
Institut für Informatik
Albert-Ludwigs-Universität, Freiburg

16. Juli 1999

Zusammenfassung

Die Ausdrucksstärke von Planungsformalismen wird in vielen Arbeiten im Gebiet der Handlungsplanung thematisiert, ohne daß der Begriff jedoch formal fundiert wird. Insbesondere im Kontext des von Blum und Furst entwickeltem GRAPHPLAN-Algorithmus gewinnt dieses Thema Relevanz, da viele Forschungsarbeiten sich mit dem Problem auseinandersetzen, ob und wie der GRAPHPLAN-Algorithmus erweitert werden kann, um ausdrucksstarke Formalismen zu behandeln. In diesem Papier wird eine Methode zur Messung der relativen Ausdrucksstärke von Planungsformalismen vorgestellt, das auf Ideen aus dem Gebiet der Wissenskompilation beruht. Die Intuition ist dabei, daß ein Formalismus so mächtig wie ein zweiter Formalismus ist, falls sich alle Domänenbeschreibungen des zweiten Formalismus „einfach" innerhalb des ersten Formalismus ausdrücken lassen und die resultierenden Pläne nicht zu stark aufgebläht werden. Diese intuitive Charakterisierung der relativen Ausdrucksstärke wird mit Hilfe von sogenannten „Kompilationsschemata" formalisiert, und darauf aufbauend werden propositionale Planungsformalismen entsprechend ihrer Ausdrucksstärke klassifiziert.

1 Einleitung

In vielen Arbeiten im Gebiet der Handlungsplanung wird immer wieder das Thema der Ausdruckskraft von Planungsformalismen angesprochen. Beispielsweise schreibt Pednault [18], daß der von ihm vorgeschlagene Formalismus ADL in seiner Ausdrucksstärke zwischen STRIPS und dem Situationskalkül liegt. Desweiteren wird zur Zeit von vielen Gruppen an dem Problem gearbeitet, den GRAPHPLAN-Algorithmus [3] so zu erweitern, daß auch für den ausdrucksstarken Planungsformalismus ADL Pläne generiert werden können [1, 8, 9, 12]. Allerdings ist nicht a priori klar, ob es wirklich erforderlich und sinnvoll ist, den Algorithmus zu erweitern, oder ob es vielleicht ausreicht, in ADL formulierte Planungsaufgaben in die im GRAPHPLAN-System benutzte Variante des STRIPS-Formalismus zu *transformieren* und dann den GRAPHPLAN-Algorithmus anzuwenden. Tatsächlich geben Gazen und Knoblock [8] eine solche Transformation an und berichten, daß das Gesamtsystem eine akzeptable Performanz zeigt.

Kann man daraus schließen, daß die Sprachkonstrukte in ADL, die über STRIPS hinausgehen, tatsächlich nur *syntaktischer Zucker* sind? Wenn man die Transformation von Gazen und Knoblock genauer analysiert, stellt man fest, daß bei der Übersetzung *konditionaler Effekte* exponentiell viele STRIPS-Aktionen erzeugt werden, was zu einer nicht tolerierbaren Aufblähung der Planungsaufgabe führt. Dies ist ein Indiz dafür, daß konditionale Effekte kein syntaktischer Zucker sind. Allerdings ist dies natürlich keine zwingende Folgerung, da es ja unter Umständen Transformationen geben könnte, die effizienter sind.

Genau diese Fragestellung hat zur Entwicklung des in diesem Papier beschriebenen formalen Rahmen der *Kompilationsschemata* geführt, mit dem man präzise charakterisieren kann, ob ein bestimmtes Sprachkonstrukt syntaktischer Zucker oder essentiell ist. Die wesentliche Idee dabei ist, daß ein Planungsformalismus $\mathcal{X}$ mindestens so ausdrucksstark wie Formalismus $\mathcal{Y}$ ist, falls man alle Planungsdomänen und die dazugehörigen Pläne des Formalismus $\mathcal{Y}$ innerhalb von $\mathcal{X}$ ausdrücken kann ohne daß dazu erheblich mehr Platz erforderlich ist. Die Übersetzung von $\mathcal{Y}$ nach $\mathcal{X}$ wird dabei von *Kompilationsschemata* geleistet.

Basierend auf dem formalen Rahmen der Kompilationsschemata werden in diesem Papier dann eine große Klasse von Planungsformalismen untersucht,

die von einfachem STRIPS bis zu Formalismen mit *konditionalen Effekten*, *booleschen Formeln in den Vorbedingungen* und *unvollständigen Zustandsbeschreibungen* reichen. Eines der Resultate dieser Analyse ist die Identifikation zweier Klassen von Planungsformalismen, die jeweils die gleiche Ausdruckskraft besitzen. Weiter ergibt sich, daß *konditionale Effekte* und *boolesche Formeln in Vorbedingungen* von ihrer Ausdruckskraft her nicht vergleichbar sind. Insbesondere folgt daraus auch, daß die oben erwähnte Transformation von Gazen und Knoblock tatsächlich in dem Sinne optimal ist, daß es keine andere Transformation gibt, die nur zu einer polynomialer Aufblähung der Planungsaufgabe führen würde falls man fordert, daß die resultierenden Pläne höchstens linear länger sein dürfen.

Der Rest dieses Papiers ist wie folgt aufgebaut. Im nächsten Abschnitt wird Planen mit dem einfachen STRIPS-Formalismus eingeführt und darauf aufbauend in Abschnitt 3 die Klasse der in diesem Papier betrachteten Planungsformalismus definiert. Im Abschnitt 4 wird diskutiert, wie Kompilationsschemata formalisiert werden können. Im darauf folgenden Abschnitt werden die erzielten Ergebnisse vorgestellt, wobei auf Beweise verzichtet wird und nur die wesentlichen Ideen skizziert werden.[1] Schließlich wird dann im zusammenfassenden Abschnitt 6 diskutiert, welche Konsequenzen sich z.B. für den Entwurf von Planungsalgorithmen ergeben.

2 Planen im einfachen STRIPS-Formalismus

Der von Fikes und Nilsson [6] eingeführte Planungsformalismus STRIPS wird auch heute noch in vielen Planungsystemen verwendet, wobei jedoch gegenüber dem Originalformalismus viele Vereinfachungen vorgenommen wurden, nicht zuletzt wegen semantischer Probleme des Originalformalismus [13]. Insbesondere wird in modernen Planern wie GRAPHPLAN [3], IPP [12] und BLACKBOX[10] nur noch (eingeschränkte) Aussagenlogik benutzt.[2] Auch wir werden uns im weiteren auf solch einfache Planungsformalismen beschränken.

[1] Die Beweise findet man in Skizzenform in dem in den Proceedings der *Deutschen Jahrestagung für Künstliche Intelligenz* veröffentlichten Papier [15] und ausführlich in einem technischen Bericht [14].

[2] Während in der Spezifikation von Planungsaufgaben Variablen vorkommen können, werden vor der Verarbeitung durch den entsprechenden Planungsalgorithmus alle Variablen durch Instantiierung beseitigt.

Mit Σ wird im folgenden eine **endliche Menge von aussagenlogischen Atomen bezeichnet**. Die **Menge aller Literale**, d.h. aller Atome und negierter Atome, über Σ wird mit $\hat{\Sigma}$ bezeichnet. Eine **Planungsdomäne** im einfachen, aussagenlogischen STRIPS-Formalismus wird durch die Menge der aussagenlogischen Atome Σ und die Menge der verfügbaren **Aktionen O** beschrieben. Aktionen sind dabei Paare $o = \langle pre, post \rangle$ bestehend aus der **Vorbedingung** $pre \subseteq \Sigma$ und der **Nachbedingung** $post \subseteq \hat{\Sigma}$.

Aktionen können Zustände verändern, wobei ein **Zustand** eine maximale, konsistente Menge von Literalen über Σ ist (äquivalent eine Wahrheitsbelegung der Atome in Σ). Eine Aktion ist **ausführbar** in einem Zustand, falls die Vorbedingung erfüllt ist, d.h., falls alle Atome der Vorbedingung im Zustand positiv vorkommen. Wird die Aktion **ausgeführt**, wird der Zustand dahingehend verändert, daß die mit der Nachbedingung der Aktion inkonsistenten Literale aus dem Zustand entfernt werden und die Nachbedingung zum Zustand hinzugenommen wird. Wir wollen außerdem vereinbaren, daß die Ausführung von Aktionen, deren Vorbedingung *nicht* erfüllt wird, zum **inkonsistenten Zustand** $\perp$ führt.

Um die obigen Definitionen zu verdeutlichen, wollen wir die folgende Planungsdomäne annehmen, bei der `bg` für einen „Batzen Geld“, `di` für den Diplomgrad und `dr` für den Doktortitel steht:

$$\begin{aligned} \Sigma &= \{\texttt{bg}, \texttt{di}, \texttt{dr}\}, \\ \mathbf{O} &= \Big\{ \langle \{\texttt{bg}\}, \{\neg\texttt{bg}, \texttt{dr}\} \rangle, \\ &\quad\ \langle \{\emptyset, \{\texttt{di}\} \rangle, \\ &\quad\ \langle \{\texttt{di}\}, \{\texttt{dr}\} \rangle \Big\}. \end{aligned}$$

Die angegebenen Aktionen können als *illegaler Titelkauf*, *Studium* und *Promotion* interpretiert werden. Die Modellierung des Erwerbs akademischer Grade und Titel ist in diesem Beispiel gegenüber der Realität naturgemäß stark vereinfacht – und angesichts all der neuen Abschlüsse auch nicht mehr ganz auf der Höhe der Zeit.

In dem Zustand $\{\texttt{bg}, \neg\texttt{di}, \neg\texttt{dr}\}$ könnten wir jetzt beispielsweise die Aktion $\langle \{\texttt{bg}\}, \{\neg\texttt{bg}, \texttt{dr}\} \rangle$, die den illegalen Titelkauf modelliert, ausführen. Der resultierende Zustand wäre dann $\{\neg\texttt{bg}, \neg\texttt{di}, \texttt{dr}\}$, d.h. das Geld ist weg und man hat einen (zweifelhaften) Titel.

Eine **Planungsaufgabe** Π ist allgmein durch eine Planungsdomäne $\Xi = \langle \Sigma, \mathbf{O} \rangle$, einen **Initialzustand I** und eine **Zielbedingung G** $\subseteq \hat{\Sigma}$ spezifiziert. Gesucht ist eine endliche Folge von Aktionen Δ, die den Initialzustand in einen konsistenten Zustand überführt, der die Zielbedingung erfüllt. Sei beispielsweise

$$\begin{aligned} \mathbf{I} &= \{\mathtt{bg}, \neg\mathtt{di}, \neg\mathtt{dr}\}, \\ \mathbf{G} &= \{\mathtt{dr}\}, \end{aligned}$$

dann gibt es zwei mögliche Aktionsfolgen, um von **I** zu einem Zustand, der **G** erfüllt, zu gelangen – den ehrlichen und den illegalen Weg. Solche erfolgreichen Aktionsfolgen werden auch **Lösungen der Planungsaufgabe** oder **Pläne für die Planungsaufgabe** genannt.

Während es bei obigen Beispiel recht einfach erscheint, Pläne zu finden, ist dies im allgemeinen Fall eine algorithmisch sehr anspruchsvolle Aufgabe. Wie Bylander [4] gezeigt hat, ist das Problem, zu entscheiden, ob für eine im hier vorgestellten Formalismus spezifizierte Planungsaufgabe überhaupt ein Plan existiert, PSPACE-vollständig.[3]

Heutige Planungssysteme die genau den oben beschriebenen Planungsformalismus einsetzen, wie z.B. das GRAPHPLAN-System, lösen Planungsaufgaben, die zwischen 50–100 Aktionen innerhalb eines Plans benötigen. Dies scheint zwar nicht übermäßig viel zu sein, ist aber gegenüber früheren Ansätzen, die schon nach 10–20 Aktionen die Segel strichen, ein signifikanter Fortschritt.

3 Die $\mathcal{S}$-Klasse

Wie schon in der Einleitung erwähnt, scheint es erstrebenswert, den durch den GRAPHPLAN-Algorithmus erzielten Fortschritt auch auf ausdrucksstärkere Sprachen zu übertragen. In diesem Abschnitt werden wir den oben eingeführten Formalismus, den wir im weiteren mit $\mathcal{S}$ bezeichnen wollen, auf verschiedene Arten erweitern.

[3] D.h. um das Problem zu entscheiden ist „nur“ polynomialer Speicherplatz erforderlich (vgl. [7, 17]). Allerdings scheinen die PSPACE-vollständigen Probleme in der Praxis schwerer zu lösen zu sein als die Probleme, die „nur“ NP-vollständig sind.

Als erstes fällt beim $\mathcal{S}$-Formalismus auf, daß es unnatürlich ist, nur Atome in den Vorbedingungen zuzulassen. Genauso könnte man allgemein *Literale* zulassen. Wir wollen eine solche Erweiterung des Formalismus um Literale in den Vorbedingungen mit $\mathcal{S}_{\mathcal{L}}$ bezeichnen.

Eine weitere natürliche Erweiterung besteht darin, daß wir allgemeine *boolesche Formeln* in den Vorbedingungen zulassen. Die entsprechende Erweiterung des Basisformalismus, die $\mathcal{S}_{\mathcal{L}}$ subsumiert, wollen wir mit $\mathcal{S}_{\mathcal{B}}$ bezeichnen.

Eine der wesentlichen Erweiterungen des von Pednault vorgeschlagenen ADL-Formalismus [18] gegenüber dem STRIPS-Formalismus ist die Möglichkeit *kontextsensitive* Aktionen zu modellieren. Wenn man z.B. an die Aktion „Gas geben“ denkt, so ist klar, daß die Effekte dieser Aktion davon abhängen, in welchem Gang sich das Auto befindet. Um diese Kontextsensitivität zu modellieren, kann man *konditionale Effekte* einsetzen. D.h. die Nachbedingung einer Aktion besteht nicht aus einer Menge von Literalen sondern aus einer Menge **konditionaler Effekte**, die ihrerseits aus einer Effektbedingung Γ (entsprechend einer Vorbedingung) und einem Effekt (einem Literal) l bestehen, symbolisch $\Gamma \Rightarrow l$. Wenn eine Aktion mit konditionalen Effekten in einem Zustand ausgeführt wird, so ergibt sich der Gesamteffekt als Vereinigung aller Effekte, deren Efeektbedingungen in dem Zustand erfüllt sind. Eine solche Erweiterung des Basisformalismus bezeichnen wir mit $\mathcal{S}_{\mathcal{C}}$.

Schließlich kann man noch fordern, daß man so etwas wie *unvollständige Zustandsbeschreibungen* modellieren möchte, d.h. Zustandsbeschreibungen, in denen der Wahrheitswert einiger Atome offen ist. Formal kann man dies erreichen, indem man *Zustandsbeschreibungen* als beliebige, konsistente Literalmengen modelliert. Die Intention dabei soll sein, daß solche unvollständigen Zustandsbeschreibungen alle Zustände (max., konsistente Literalmengen bzw. Wahrheitsbelegungen) charakterisieren, die die Zustandsbeschreibung erfüllen. Die entsprechende Erweiterung des Basisformalismus wollen wir mit $\mathcal{S}_{\mathcal{I}}$ bezeichnen.

Natürlich kann man die oben skizzierten Erweiterungen auch miteinander kombinieren. Zum Beispiel würden wir so einen Planungsformalismus $\mathcal{S}_{\mathcal{BIC}}$ erhalten, in dem unvollständige Zustandsbeschreibungen, konditionale Effekte und boolesche Formeln in Vorbedingungen und Effektbedingungen erlaubt sind. Die meisten Kombinationen von Erweiterungen sind dabei völlig unproblematisch in dem Sinne, daß die im Abschnitt 2 skizzierte Seman-

tik von Aktionen sich ohne Probleme erweitern läßt. Allerdings muß man sich einige Gedanken über die Kombination von konditionalen Effekten mit unvollständigen Zustandsbeschreibungen machen. Da es in der Planungsgemeinde für diese Kombination bisher keine Standardlösung gibt, wollen wir die folgende Aktionssemantik annehmen. Eine Aktion ist nur dann ausführbar, wenn alle konditionalen Effekte in allen durch eine unvollständige Zustandsbeschreibung charakterisierten Zuständen zu dem gleichen Gesamteffekt führen. Anderenfalls ist das Ergebnis der Aktion $\perp$.

Die hier eingeführten Planungsformalismen bilden eine partielle Ordnung bezüglich der syntaktischen Merkmale. Diese partielle Ordnung ist in Abbildung 1 mit Hilfe eines Hasse-Diagramms dargestellt. Im weiteren wird der

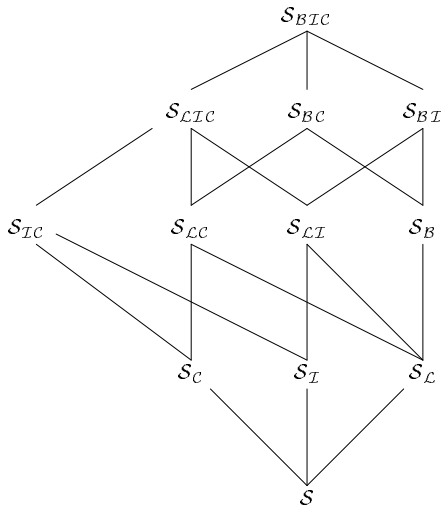

Abbildung 1: Die partielle Ordnung der Planungsformalismen in der $\mathcal{S}$-Klasse entsprechend den syntaktischen Merkmalen

Formalismus $\mathcal{X}$ eine **Spezialisierung** von $\mathcal{Y}$ genannt, symbolisch $\mathcal{X} \sqsubseteq \mathcal{Y}$, wenn $\mathcal{X}$ identisch mit $\mathcal{Y}$ ist oder im Hasse-Diagramm unterhalb von $\mathcal{Y}$ liegt.

Während man erwarten würde, daß Planen im $\mathcal{S}$-Formalismus sehr viel einfacher ist als im $\mathcal{S}_{\mathcal{BIC}}$-Formalismus, stellt sich heraus, daß die Komplexität des *Planexistenzproblem* in allen Formalismen identisch ist. Es ist in allen hier betrachteten Formalismen PSPACE-vollständig [14, 15]. Dieses Resultat

könnte man dahingehend interpretieren, daß man $\mathcal{S}_{\mathcal{BIC}}$-Planungsaufgaben einfach, d.h. in polynomialer Zeit, in $\mathcal{S}$-Aufgaben transformieren kann um sie dann, z.B. durch den GRAPHPLAN-Algorithmus, lösen zu lassen. Insbesondere könnte man daraus schließen, daß die Ausdrucksstärke von $\mathcal{S}_{\mathcal{BIC}}$ und $\mathcal{S}$ identisch sind. Allerdings sagt das Ergebnis ja nichts darüber aus, wie lang die Pläne im Zielformalismus werden, ob es möglich ist Zustandsbeschreibungen und Pläne im Zielformalismus in den Ausgangsformalismus zurückzuübersetzen usw. Tatsächlich ist auch völlig unklar, wie man den GRAPHPLAN-Algorithmus modifizieren müßte, um beispielsweise in dem Formalismus $\mathcal{S}_{\mathcal{BIC}}$ planen zu können.

4 Ausdrucksstärke und Kompilationsschemata

Ein Planungsformalismus $\mathcal{X}$ ist mindestens so ausdrucksstark wie Formalismus $\mathcal{Y}$, falls alle Planungsaufgaben des Formalismus $\mathcal{Y}$ sich in $\mathcal{X}$ ausdrücken lassen. Das heißt, es muß eine *Funktion* F geben, die $\mathcal{Y}$-Planungsaufgaben auf $\mathcal{X}$-Planungsaufgaben abbildet und dabei bestimmte Bedingungen invariant läßt. Man könnte insbesondere fordern, daß die Funktion F eine der folgenden *Invarianzbedingungen* erfüllt:

I1. die Lösungsexistenz ist für beide Planungsaufgaben identisch,

I2. die Länge eines minimalen Plans im Zielformalismus ist linear (oder polynomial) in der Länge eines minimalen Plans im Ausgangsformalismus,

I3. die Länge eines minimalen Plans im Zielformalismus ist identisch mit der Länge eines minimalen Plans im Ausgangsformalismus,

I4. Pläne im Zielformalismus, die durch Transformation einer Planungsaufgabe entstanden sind, lassen sich einfach in den Ausgangsformalismus zurücktransformieren.

Während I1 sicherlich eine Minimalforderung ist, wenn man davon spricht, daß Aufgaben in einem anderen Formalismus *ausgedrückt* werden, sind die

weiteren möglichen Einschränkungen durchaus alle sinnvolle Ergänzungen. Für praktische Anwendungen ist insbesondere I4 sehr sinnvoll.

Man kann weiter fordern, daß die Funktion F bestimmte *Ressourcenbedingungen* erfüllt:

R1. der für die Darstellung des Ergebnisses der Funktion F notwendige Platz sollte beschränkt sein, z.B. polynomial in der Größe der Planungsaufgabe im Ausgangsformalismus;

R2. der für die Berechnung des Ergebnisses erforderliche Aufwand sollte beschränkt sein (z.B. nur polynomiale Zeit).

Während die Bedingung R1 sinnvoll erscheint, scheint R2 nicht notwendig etwas mit Ausdrucksfähigkeit zu tun zu haben. Wenn wir davon reden, daß wir $\mathcal{Y}$-Aufgaben in $\mathcal{X}$ ausdrücken, sagen wir nichts über die Komplexität der entsprechenden Transformationsfunktion F aus. Tatsächlich könnte man immer noch davon reden, daß $\mathcal{X}$ mindestens so ausdrucksfähig wie $\mathcal{Y}$ ist, selbst wenn die entsprechende Funktion F nicht berechenbar ist!

Schließlich kann man noch *Modularitätsbedingungen* für F fordern:

M1. die Funktion F soll ganze Planungsaufgaben transformieren;

M2. die Funktion F soll die Planungsdomäne unabhängig von Initialzustand $\mathbf{I}$ und Zielbedingungen $\mathbf{G}$ transformieren und $\mathbf{I}$ und $\mathbf{G}$ unverändert lassen.

Während man die Bedingung M2 vielleicht überraschend finden mag, so ist es doch nicht völlig abseitig, zu fordern, daß die Planungsdomänen und Zustände unabhängig voneinander transformiert werden sollen. Erstens scheint es sinnvoll zu sein, die Planungsdomänen unabhängig von Initialzustand und Zielspezifikation zu transformieren, wenn wir von der Ausdrucksstärke von Planungsformalismen sprechen. Zweitens benutzen alle praktischen Ansätze [8] und alle theoretischen Ansätze [2], die auf einem transformationellen Rahmen beruhen, solche strukturierten Transformationen. Drittens werden wir feststellen, daß ohne diese Beschränkung die Untersuchung der Ausdrucksstärke keinen Sinn macht.

Eine Möglichkeit, die Ausdrucksstärke von Planungsformalismen zu messen, könnte darin bestehen, daß man von den oben erwähnten Bedingungen die Kombination (I1, R2, M1) wählt, d.h. F muß nur die Lösungsexistenz erhalten, F muß in polynomialer Zeit berechenbar sein und die gesamte Planungsaufgabe wird transformiert. Solche Funktionen sind dann aber nichts anderes als *polynomiale Transformationen*, wie sie in der Komplexitätstheorie auch eingesetzt werden. Insbesondere würde für die Planungsformalismen in der $\mathcal{S}$-Klasse folgen, daß sie alle die gleiche Ausdrucksstärke besitzen, da das Planexistenzproblem ja in all diesen Formalismen PSPACE-vollständig ist.

Da dieses Ergebnis offensichtlich nicht der Intuition entspricht, sollte man die Bedingungen verfeinern. Bäckström [2] fordert in seiner Arbeit, die sich ebenfalls mit dem Problem der Formalisierung des Begriffs der Ausdrucksstärke beschäftigt, daß man diese mit Hilfe von Funktionen bestimmen sollte, die die Bedingungen (I3, R2, M1) erfüllen, d.h. die im Zielformalismus resultierenden Pläne sollten genauso lang sein wie die Pläne im Ausgangsformalismus. Bäckström nennt solche Funktionen *ESP-Transformationen*. Während ESP-Transformationen die Intuition besser zu treffen scheinen, ist nicht klar, warum die Pläne *exakt* die gleiche Länge haben müssen. Außerdem scheint, wie oben schon bemerkt, die Laufzeitbeschränkung der Funktion F auf polynomiale Laufzeit nicht zu rechtfertigen zu sein, wenn man die Ausdrucksfähigkeit messen will.

Das führt uns zu Abschwächungen bei der Invarianzbedingung und bei der Ressourcenbedingung. Wir fordern, daß die Pläne im Zielformalismus nur linear oder polynomial länger sein dürfen und daß das Ergebnis der Übersetzung nur polynomialen Platz benötigt, d.h. wir wählen die Bedingungen (I2, R1, M1). Da jetzt die Funktion F beliebig viel Laufzeit in Anspruch nehmen kann, kann die Funktion F allerdings die ursprüngliche Planungsaufgabe *lösen* und irgendeine Planungsaufgabe im Zielformalismus erzeugen, die genauso viele Schritte benötigt, d.h., wir erhalten sofort wieder die Folgerungen, daß alle Formalismen in der $\mathcal{S}$-Klasse die gleiche Ausdrucksstärke besitzen. Dies rührt aber offensichtlich daher, daß Initialzustand und Zielbedingungen mit transformiert werden dürfen. Wenn man von Ausdrucksstärke von Planungsformalismen spricht, meint man aber meist die Ausdruckstärke der Aktionen unabhängig von konkreten Zuständen. Aus diesem Grund scheint es sinnvoll, nur Transformationen zu betrachten, die unabhängig von Initi-

alzustand und Zielbedingungen sind, d.h., wir wählen die Bedingungen (I2, R1, M2).

Solche Transformationen sind ähnlich den in der sogenannten *Wissenskompilation* [5] eingesetzten Transformationen, bei denen ein Problem in einen *fixen* und einen *variablen* Teil aufgeteilt wird und man versucht, den fixen Teil zu „kompilieren“, um so das Gesamtproblem effizienter zu lösen. Aus diesem Grund werden wir Transformationen, die (I2, R1, M2) erfüllen, auch *Kompilationsschemata* nennen.

Allerdings ist die Bedingung M2 in vielen Fällen zu strikt. Oft ist es hilfreich, Hilfsatome einzuführen, die z.B. für Synchronisation von Aktionen sorgen. Aus diesem Grund wollen wir zulassen, daß Kompilationsschemata Initialzustand und Zielbedingung erweitern. Weiterhin kann es notwendig sein, am Initialzustand und an der Zielbedingung kleine syntaktische Änderungen vorzunehmen, um z.B. Formalismen mit unvollständigen Zustandsbeschreibungen mit solchen Formalismen zu vergleichen, die nur vollständige Zustandsbeschreibungen zulassen. Aus diesem Grund wollen wir sehr eingeschränkte *Zustandstransformationsfunktionen* erlauben.

Formal ist ein **Kompilationsschema von $\mathcal{X}$ nach $\mathcal{Y}$** ein Tupel von Funktionen $\mathbf{f} = \langle f_\xi, f_i, f_g, t_i, t_g \rangle$, das eine Funktion F von $\mathcal{X}$-Planungsaufgaben $\Pi = \langle \Xi, \mathbf{I}, \mathbf{G} \rangle$ nach $\mathcal{Y}$-Planungsaufgaben $F(\Pi)$ auf die folgende Art induziert:

$$F(\Pi) = \langle f_\xi(\Xi), f_i(\Xi) \cup t_i(\Sigma, \mathbf{I}), f_g(\Xi) \cup t_g(\Sigma, \mathbf{G}) \rangle.$$

Dabei sollen $\mathbf{f}$ und die induzierte Funktion F folgende Bedingungen erfüllen:

1. Für Π existiert ein Plan genau dann wenn ein Plan für $F(\Pi)$ existiert.
2. Die **Zustandstransformationsfunktionen** t_i und t_g sind **modular**, d.h., für $\Sigma = \Sigma_1 \cup \Sigma_2$, $S \subseteq \widehat{\Sigma}$ und $S \not\models \bot$ erfüllen die Funktionen t_x (für $x = i, g$) die folgende Bedingung:
$$t_x(\Sigma, S) = t_x(\Sigma_1, S \cap \widehat{\Sigma_1}) \cup t_x(\Sigma_2, S \cap \widehat{\Sigma_2}).$$
3. Die Größe der Resultate der Funktionen f_ξ, f_i, und f_g ist polynomial in der Größe ihrer Argumente.

Obwohl keine Forderungen an den Rechenzeitbedarf von f_ξ, f_i und f_g gestellt werden, sind wir natürlich auch an *effizienten Kompilationsschemata* interessiert. Kompilationsschemata, bei denen f_ξ, f_i und f_g in polynomialer Zeit berechenbar sind, werden im folgenden als **Polynomialzeit-Kompilationsschemata** bezeichnet.

Desweiteren soll ja auch die Länge der resultierenden Pläne beschränkt werden. Falls ein Kompilationsschema $\mathbf{f}$ die Eigenschaft hat, daß es zu jedem Plan Δ für Π einen Plan Δ' für $F(\Pi)$ gibt, der höchstens k Aktionen länger als Δ ist, für eine Konstante $k > 0$, so nennen wir $\mathbf{f}$ **exakt planlängenerhaltend** (modulo einem konstanten Summanden). Ist der Plan Δ' höchstens k-mal länger, ist das Kompilationsschema **linear planlängenerhaltend**. Ist die Größe des Plans Δ' polynomial durch die Größe von Δ und Π beschränkt, ist das Kompilationsschema **polynomial planlängenerhaltend**. Generell ist $\mathcal{X}$ nach $\mathcal{Y}$ **kompilierbar** (in poly. Zeit, exakt, linear, oder polynomial planlängenerhaltend), falls es ein entsprechendes Kompilationschema gibt. Die Notation $\mathcal{X} \preceq^x \mathcal{Y}$ soll bedeuten, daß $\mathcal{X}$ nach $\mathcal{Y}$ **kompilierbar** ist, wobei $x \in \{1, c, p\}$ je nachdem ob das entsprechende Kompilationsschema exakt, linear oder polynomial planlängenerhaltend ist. Die Notation $\mathcal{X} \preceq_p^x \mathcal{Y}$ bedeutet, daß es ein Polynomialzeit-Kompilationsschema gibt.

Eine offensichtliche Folgerung aus diesen Definitionen ist die folgende Aussage.

Aussage 1 *Die Relationen $\preceq^x$ und $\preceq_p^x$ sind für alle $x \in \{1, c, p\}$ transitiv und reflexiv.*

Wenn wir mit π_i die Projektion auf das i-te Argument bezeichnen und mit $\emptyset$ die Funktion, die immer die leere Menge als Resultat hat, induziert das Kompilationsschema $\mathbf{f} = \langle \pi_1, \emptyset, \emptyset, \pi_2, \pi_2 \rangle$ offensichtlich die identische Transformation. Daraus folgt, daß es immer ein Polynomialzeit-Kompilationsschema gibt, das exakt planlängenerhaltend ist, wenn wir uns in der partiellen Ordnung von unten nach oben bewegen.

Aussage 2 $\mathcal{X} \sqsubseteq \mathcal{Y}$ *impliziert* $\mathcal{X} \preceq_p^1 \mathcal{Y}$.

Die Frage, ob es außer den durch die Spezialisierungsrelation und die Transitivität implizierten Kompilierbarkeitsbeziehungen weitere Kompilierbarkeitsbeziehungen gibt, ist Gegenstand des folgenden Abschnitts.

5 Kompilierbarkeit

Als erstes, einfaches Resultat ergibt sich, daß die Formalismen $\mathcal{S}_{\mathcal{LI}}$, $\mathcal{S}_{\mathcal{L}}$, $\mathcal{S}_{\mathcal{I}}$und $\mathcal{S}$ in polynomialer Zeit, exakt planlängenerhaltend ineinander kompilierbar sind. Dies folgt aus den Resultaten von Bäckström [2], dessen ESP-Reduktionen als Kompilationsschemata interpretierbar sind. Um die Idee solcher Kompilationsschemata zu illustrieren, soll exemplarisch das Kompilationsschema für $\mathcal{S}_{\mathcal{L}} \preceq_p^1 \mathcal{S}$ skizziert werden.

Um Aktionen, die negative Literale in der Vorbedingung enthalten können, in solche zu übersetzen, die nur Atome in der Vorbedingungen zulassen, führen wir zu jedem Atom p ein neues Atom $\overline{p}$ ein, das für die Negation von p steht. In den Nachbedingungen der Aktionen müssen dann p und $\overline{p}$ entsprechend gesetzt werden. Hat also eine Aktion in $\mathcal{S}_{\mathcal{L}}$ die folgende Gestalt:

$$\langle \{p, \neg q\}, \{\neg p, r\} \rangle,$$

so muß die transformierte Aktion im $\mathcal{S}$-Formalismus folgende Form haben:

$$\langle \{p, \overline{q}\}, \{\neg p, \overline{p}, r, \neg \overline{r}\} \rangle.$$

Entsprechend müssen die Initialzustände und Zielbedingungen modifiziert werden, was aber durch *modulare Zustandstransformationsfunktionen* geleistet werden kann. Es ist dann klar, daß es für jeden Plan in $\mathcal{S}_{\mathcal{L}}$ einen Plan genau gleicher Länge in $\mathcal{S}$ geben muß (und umgekehrt).[4] Das heißt, die skizzierte Transformation ist tatsächlich ein exakt planlängenerhaltendes Kompilationsschema, das in polynomialer Zeit ausgeführt werden kann.

Eine Sicht auf dieses Resultat ist, daß unvollständige Zustände und negierte Atome syntaktischer Zucker sind, wenn man von $\mathcal{S}$ als Formalismus ausgeht. Interessanterweise kann man dieses Resultat generalisieren. Falls konditionale Effekte zu $\mathcal{S}$ hinzugenommen werden, bleiben unvollständige Zustände und negierte Atome syntaktischer Zucker. Dieses Resultat ist schwieriger zu beweisen, da die Semantik konditionaler Effekte im Zusammenhang mit unvollständigen Zustandsbeschreibungen einige Komplikationen aufwirft. Aber es ist möglich, diese Komplikationen für die hier angenommene Aktionssemantik zu lösen.

[4] Diese Transformation findet man z.B. auch in dem Papier von Blum und Furst [3].

Eine Zusammenfassung dieser Ergebnisse ist in Abbildung 2 dargestellt. Es gibt zwei Äquivalenzklassen bezüglich $\preceq_p^1$, die im folgenden mit $[\mathcal{S}_{\mathcal{LI}}]$ und $[\mathcal{S}_{\mathcal{LIC}}]$ entsprechend ihrer größten Elemente bezeichnet werden.

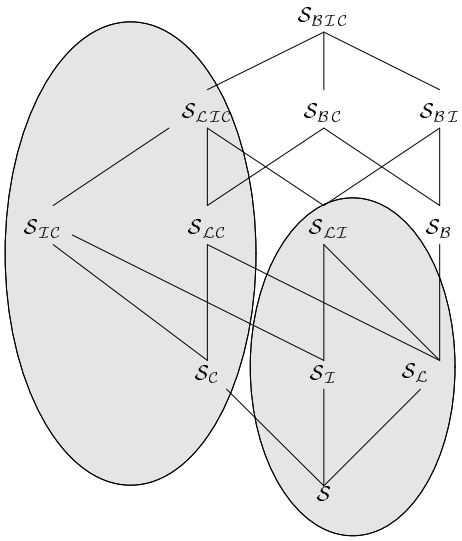

Abbildung 2: Äquivalenzklassen bezüglich $\preceq_p^1$

Außer den in Abbildung 2 dargestellten exakt planlängenerhaltenden Kompilierbarkeitsbeziehungen gibt es keine weiteren, auch wenn wir beliebige Berechnungsressourcen für den Kompilationsprozeß erlauben. Dies gilt tatsächlich auch noch, wenn die Kompilationsschemata nur linear planlängenerhaltend sein sollen.

Wie zeigt man aber, daß es kein linear planlängenerhaltendes Kompilationsschema zwischen zwei Planungsformalismen geben kann? Die Idee ist, daß man eine Folge von Planungsdomänen im Ausgangsformalismus spezifiziert, die für alle Paare von Initialzuständen und Zielbedingungen entweder einen einschrittigen Plan besitzen oder unlösbar sind. Dann zeigt man, daß die Annahme eines linear planlängenerhaltenden Kompilationsschema vom Ausgangs- zum Zielformalismus zu einem elementaren Widerspruch oder aber zu einer Folgerung führt, die als sehr unwahrscheinlich angesehen wird.

Zur Illustration dieser Vorgehensweise soll skizziert werden wie man zeigt,

daß $\mathcal{S}$ erweitert um konditionale Effekte und negative Atome in den Vorbedingungen (also $\mathcal{S}_{\mathcal{LC}}$) nicht linear planlängenerhaltend nach $\mathcal{S}$ erweitert um boolesche Vorbedingungen (also $\mathcal{S}_{\mathcal{B}}$) kompiliert werden kann. Wir wollen dazu vereinfachend annehmen, daß die Zustandstransformationsfunktionen t_i und t_g die Identitätsfunktionen sind.

Wir betrachten eine Folge von $\mathcal{S}_{\mathcal{LC}}$-Planungdomänen $\Xi_n = \langle \Sigma_n, \mathbf{O}_n \rangle$, wobei

$$\begin{aligned} \Sigma_n &= \{p_1, \ldots, p_n\}, \\ \mathbf{O}_n &= \Big\{\langle \emptyset, \{p_1 \Rightarrow \neg p_1, \neg p_1 \Rightarrow p_1, \ldots, p_n \Rightarrow \neg p_n, \neg p_n \Rightarrow p_n\}\rangle\Big\}. \end{aligned}$$

Mit anderen Worten, es gibt n verschiedene Atome in Σ_n und es gibt genau eine Aktion in $\mathbf{O}_n$, die alle Literale im Zustand invertiert. Aus der Konstruktion folgt, daß es für alle 2^n verschiedenen Paare $(\mathbf{I}, \neg\mathbf{I})$ (wobei $\neg S$ die elementweise Negation bezeichnen soll) einen einschrittigen Plan gibt, und daß es für alle Paare $(\mathbf{I}, \mathbf{G})$ mit $\mathbf{I} \not\supseteq \neg\mathbf{G}$ keinen Plan gibt.

Wenn wir jetzt annehmen, daß es ein linear planlängenerhaltendes Kompilationsschema von $\mathcal{S}_{\mathcal{LC}}$ nach $\mathcal{S}_{\mathcal{B}}$ gibt, muß es für alle $(\mathbf{I}, \neg\mathbf{I})$ Paare einen k-schrittigen Plan in der kompilierten Domäne geben. Da bei der Kompilation höchstens polynomial viele Aktionen in der $\mathcal{S}_{\mathcal{B}}$-Planungsdomäne erzeugt worden sind, gibt es auch nur polynomial viele verschiedene k-schrittige Aktionsfolgen. Wenn man also n groß genug wählt, muß ein $\mathcal{S}_{\mathcal{B}}$-Plan Δ' für zwei verschiedene Paare $(\mathbf{I}, \neg\mathbf{I})$ und $(\mathbf{I}', \neg\mathbf{I}')$ benutzt werden. Wir wollen o.B.d.A annehmen, daß es ein Atom p gibt mit $p \in \mathbf{I}$ und $\neg p \in \mathbf{I}'$. Da die Pläne im $\mathcal{S}_{\mathcal{B}}$-Formalismus keine konditionalen Effekte besitzen und alle Literale der Zielbedingung durch den Plan generiert werden müssen, muß der Plan Δ' tatsächlich die gleichen Zielliterale erzeugen. Das heißt aber, daß der Plan sowohl p als auch $\neg p$ erzeugen muß, was unmöglich ist. Das impliziert aber, daß es das angenommene Kompilationsschema nicht geben kann.

Wenn man allgemeine, modulare Zustandstransformationsfunktionen zuläßt, wird das Argument etwas komplizierter, aber der Kern der Argumentation ist derselbe. Mit konditionalen Effekten kann man mehrere Dinge parallel machen und dies kann man nicht durch einen nur linear längeren Plan, in dem keine konditionalen Effekte zugelassen sind, simulieren. Weiter ist das Argument völlig unabhängig davon, ob wir vollständige oder unvollständige Zustandsbeschreibungen haben [14, Theorem 11]:

Satz 1 $\mathcal{S}_{\mathcal{LIC}} \not\preceq^c \mathcal{S}_{\mathcal{BI}}$.

Dieses Resultat beantwortet unter anderem auch die in der Einleitung aufgeworfene Frage, ob es effizientere Transformation von $\mathcal{S}_{\mathcal{LC}}$ nach $\mathcal{S}$ als die von Gazen und Knoblock [8] gibt. Da $\mathcal{S}_{\mathcal{LIC}}$ und $\mathcal{S}_{\mathcal{LC}}$ äquivalent bezüglich $\preceq_p^1$ sind, folgt aus Satz 1, daß $\mathcal{S}_{\mathcal{LC}} \not\preceq^c \mathcal{S}_{\mathcal{BI}}$. Da außerdem $\mathcal{S} \sqsubseteq \mathcal{S}_{\mathcal{BI}}$, folgt mit Aussage 2 $\mathcal{S} \preceq_p^1 \mathcal{S}_{\mathcal{BI}}$. Das heißt aber, daß $\mathcal{S}_{\mathcal{LC}} \preceq^c \mathcal{S}$ unmöglich ist, da ansonsten mit Aussage 1 und $\mathcal{S} \preceq_p^1 \mathcal{S}_{\mathcal{BI}}$ folgen würde, daß $\mathcal{S}_{\mathcal{LC}} \preceq^c \mathcal{S}_{\mathcal{BI}}$, was ja dem obigen Satz widersprechen würde. Das heißt aber, daß falls nur eine lineare Aufblähung der resultierenden Pläne im Zielformalismus zugelassen ist, es kein Kompilationsschema gibt – d.h. wir haben notwendig eine superpolynomiale Aufblähung der Planungsdomäne, wenn wir von $\mathcal{S}_{\mathcal{LC}}$ nach $\mathcal{S}$ transformieren wollen.

Auch wenn wir Kompilierbarkeit in die andere Richtung betrachten, d.h. von booleschen Formeln zu konditionalen Effekten, erhalten wir kein positives Ergebnis. Für den Beweis der Nichtkompilierbarkeit von $\mathcal{S}_{\mathcal{B}}$ nach $\mathcal{S}_{\mathcal{LIC}}$ sind allerdings Resultate aus der Schaltkreiskomplexität notwendig, auf die hier aus Platzgründen nicht weiter eingegangen werden soll. Stattdessen wird nur das folgende Resultat präsentiert [14, Theorem 18]:

Satz 2 $\mathcal{S}_{\mathcal{B}} \not\preceq^c \mathcal{S}_{\mathcal{LIC}}$.

Bei den bisherigen Resultaten hat die Unvollständigkeit von Zustandsbeschreibungen keine Rolle gespielt. Allerdings wird die Unvollständigkeit relevant, wenn allgemeine boolesche Formeln in Vorbedingungen zugelassen werden. Man sieht dann leicht ein, daß die Allgemeingültigkeit von booleschen Formeln auf die Existenz von einschrittigen Plänen reduziert werden kann. Daraus folgt aber sofort, daß das *Planexistenzproblem für einschrittige Pläne* in $\mathcal{S}_{\mathcal{BI}}$ coNP-schwierig ist.[5]

Betrachten wir dagegen den ausdrucksstärksten Formalismus mit vollständigen Zustandsbeschreibungen, den Formalismus $\mathcal{S}_{\mathcal{B}}$, dann wird schnell klar, daß das *Planexistenzproblem für Pläne mit polynomial vielen Schritten* innerhalb von NP liegt, da es ausreicht, polynomial viele Aktionen zu erraten und dann die Ausführbarkeit in polynomialer Zeit zu verifizieren.

Es wird nun allgemein angenommen, daß NP $\neq$ coNP. Unter dieser plausiblen Annahme folgt dann, daß es keine polynomial planlängenerhaltende

[5] Ein Problem ist coNP-schwierig falls das komplementäre Problem NP-schwierig ist.

Polynomialzeit-Kompilationsschemata von $\mathcal{S}_{\mathcal{B}}$ nach $\mathcal{S}_{\mathcal{LIC}}$ geben kann. Dieses Ergebnis ist aber insofern unbefriedigend, da nur Polynomialzeit-Kompilationsschemata ausgeschlossen werden. Unter Benutzung von Resultaten zur Beziehung zwischen sogenannten *nicht-uniformen Komplexitätsklassen* und der *polynomialen Hierarchie*[6] [19] sowie einer Beweistechnik, die erstmals von Kautz und Selman [11] eingesetzt wurde, kann man jedoch das Resultat generalisieren. Allerdings gilt dies nur unter der Annahme, daß *die polynomiale Hierarchie nicht kollabiert.* Obwohl diese Annahme stärker als $\mathsf{NP} \neq \mathsf{coNP}$ ist, wird sie jedoch allgemein für wahr gehalten.

Satz 3 *$\mathcal{S}_{\mathcal{BI}} \not\preceq^p \mathcal{S}_{\mathcal{BC}}$ falls die polynomiale Hierarchie nicht kollabiert.*

Eine Zusammenfassung der in diesem Abschnitt vorgestellten Resultate findet sich in Tabelle 1. Dabei bedeutet $=$, daß die Kompilierbarkeit sich aus

$\preceq^x$	$\mathcal{S}_{\mathcal{BIC}}$	$[\mathcal{S}_{\mathcal{LIC}}]$	$\mathcal{S}_{\mathcal{BC}}$	$\mathcal{S}_{\mathcal{BI}}$	$\mathcal{S}_{\mathcal{B}}$	$[\mathcal{S}_{\mathcal{LI}}]$
$\mathcal{S}_{\mathcal{BIC}}$	$=$	$\not\preceq^p$ (3)	$\not\preceq^p$ (3)	$\not\preceq^c$ (1)	$\not\preceq^p$ (3)	$\not\preceq^p$ (3)
$[\mathcal{S}_{\mathcal{LIC}}]$	$\sqsubseteq$	$=$	$\sqsubseteq$	$\not\preceq^c$ (**1**)	$\not\preceq^c$ (1)	$\not\preceq^c$ (1)
$\mathcal{S}_{\mathcal{BC}}$	$\sqsubseteq$	$\not\preceq^c$ (2)	$=$	$\not\preceq^c$ (1)	$\not\preceq^c$ (1)	$\not\preceq^c$ (2)
$\mathcal{S}_{\mathcal{BI}}$	$\sqsubseteq$	$\not\preceq^p$ (3)	$\not\preceq^p$ (**3**)	$=$	$\not\preceq^p$ (3)	$\not\preceq^p$ (3)
$\mathcal{S}_{\mathcal{B}}$	$\sqsubseteq$	$\not\preceq^c$ (**2**)	$\sqsubseteq$	$\sqsubseteq$	$=$	$\not\preceq^c$ (2)
$[\mathcal{S}_{\mathcal{LI}}]$	$\sqsubseteq$	$\sqsubseteq$	$\sqsubseteq$	$\sqsubseteq$	$\sqsubseteq$	$=$

Tabelle 1: Resultate zur Nichtkompilierbarkeit

der Reflexivität ergibt. Der Eintrag $\sqsubseteq$ bedeutet, daß es ein Kompilationsschema auf Grund der Aussage 2 gibt. In allen anderen Fällen wird das stärkste Nichtkompilierbarkeitsresultat und der Satz angegeben, auf dem das Resultat beruht (möglicherweise unter Anwendungen der Aussagen 1 und 2).

Natürlich wird man sich die Frage stellen, wie es sich mit der Kompilierbarkeit verhält, wenn man eine polynomiale Aufblähung der Pläne im Zielformalismus zuläßt. Wie man zeigen kann [14], existieren Polynomialzeit-Kompilationsschemata, die polynomial planlängenerhaltend sind, für alle Paa-

[6] Die polynomiale Hierarchie ist eine Hierarchie von Komplexitätsklassen zwischen NP und PSPACE. Man nimmt allgemein an, daß diese Hierarchie unendlich ist.

re von Formalismen, für die ein solches Kompilationsschema nicht durch die obigen Resultate ausgeschlossen wird.

6 Zusammenfassung und Diskussion

Motiviert durch die Thematisierung des Begriffs *Ausdrucksstärke* in vielen Arbeiten zur Handlungsplanung, wurde die Frage gestellt, wie dieser Begriff formal gefaßt werden könnte. Offensichtlich bezieht sich der Begriff der Ausdrucksstärke darauf, wie konzise Planungsdomänen und Pläne dargestellt werden können. Basierend auf dieser Charakterisierung und Ideen, die im Gebiet der Wissenskompilation [5] entwickelt wurden, wurde der Begriff des *Kompilationsschema* eingeführt, mit Hilfe dessen die relative Ausdrucksstärke von Planungsformalismen beurteilt werden kann.

Dieser formale Rahmen wurde dann benutzt, um eine große Klasse von Planungsformalismen zu analysieren. Das wohl überraschendste Resultat dieser Analyse ist, daß es möglich war, für alle Paare von Formalismen zu einem definitiven Ergebnis zu kommen. Entweder konnte gezeigt werden, daß ein Polynomialzeit-Kompilationsschema mit den geforderten Eigenschaften für die Planlänge existiert, oder es konnte bewiesen werden, daß ein Kompilationsschema unmöglich ist, auch wenn man beliebige Berechnungsressourcen für den Kompilationsprozeß zuläßt. Damit gehen die Ergebnisse, die hier erzielt wurden, signifikant über eine ähnliche Analyse von Bäckström [2] hinaus. Bäckström hatte sogenannte ESP-Reduktionen benutzt, um die Ausdrucksfähigkeit von Planungsformalismen zu beurteilen. Seine Untersuchung beschränkte sich dabei allerdings auf die Formalismen $\mathcal{S}$, $\mathcal{S}_{\mathcal{L}}$, $\mathcal{S}_{\mathcal{I}}$ und $\mathcal{S}_{\mathcal{LI}}$. Zudem ist unklar, wie man mit Hilfe von ESP-Reduktionen zu negativen Ergebnissen gelangen könnte.

Inwieweit sind jetzt diese Resultate relevant für das Design von Planungsalgorithmen? Kurz gesagt bedeutet ein positives Kompilierbarkeitsresultat, daß man das entsprechende Sprachmerkmal durch Vorverarbeitung behandeln kann und der eigentliche Algorithmus nicht notwendig geändert werden muß. So kann man z.B. Planen im Formalismus $\mathcal{S}_{\mathcal{LI}}$ dadurch realisieren, daß man das entsprechende Kompilationsschema benutzt um aus $\mathcal{S}_{\mathcal{LI}}$-Planungsaufgaben $\mathcal{S}$-Planungsaufgaben zu erzeugen, die dann z.B. von GRAPHPLAN gelöst werden können. Um den von GRAPHPLAN erzeugten Plan

dann auch einsetzen zu können, müßte man diesen allerdings zurück in einen $\mathcal{S}_{\mathcal{L}}$-Plan übersetzen. Während diese Rückübersetzbarkeit bisher nicht explizit gefordert wurde, stellt sich doch heraus, daß bei allen Kompilationsschemata, die für die in diesem Papier vorgestellten Resultate benutzt wurden, diese immer zu Plänen im Zielformalismus führen, die in den Ausgangsformalismus übersetzbar sind.

Die Nicht-Kompilierbarkeitsresultate zeigen entsprechend, daß Vorverarbeitungsschritte nicht ausreichend sind, da sie zu einer exponentiellen Aufblähung der Planungsaufgabe führen. In diesem Kontext konnte insbesondere gezeigt werden, daß die von Gazen und Knoblock [8] vorgeschlagene Transformation von $\mathcal{S}_{\mathcal{LC}}$ nach $\mathcal{S}$ optimal in dem Sinne ist, daß alle solche Transformationen, die das Planwachstum linear beschränken, notwendig zu einer superpolynomialen Aufblähung der Planungsaufgabe führen. Das heißt insbesondere auch, daß es sinnvoll ist, den GRAPHPLAN-Algorithmus zu erweitern, wenn man in $\mathcal{S}_{\mathcal{LC}}$ planen will.

Schließlich mag man sich noch fragen, welche Resultate man erhält, wenn man statt allgemeiner boolesche Formeln in Vorbedingungen nur syntaktisch eingeschränkte Formeln zuläßt. Diese Frage wurde im vorliegenden Artikel aus Platzgründen nicht behandelt, wurde aber in einem anderen Papier [16] untersucht.

Danksagung

Ein Teil der in diesem Papier vorgestellten Forschungsarbeit wurde während eines Forschungsaufenthalts am KI-Institut der University of New South Wales geleistet. Dank gebührt Norman Foo, Maurice Pagnucco, Abhaya Nayak und den anderen Mitgliedern des Instituts für die vielen Diskussionen und Cappucinos. Außerdem möchte ich mich bei Birgitt Jenner und Jacobo Toran für die Hinweise bezüglich Schaltkreiskomplexität bedanken.

Literatur

[1] C. R. Anderson, D. E. Smith und D. S. Weld. Conditional effects in graphplan. In R. Simmons, M. Veloso und S. Smith, Hrsg., *Proceedings*

of the 4th International Conference on Artificial Intelligence Planning Systems (AIPS-98), S. 44–53. AAAI Press, Menlo Park, 1998.

[2] C. Bäckström. Expressive equivalence of planning formalisms. *Artificial Intelligence*, 76(1–2):17–34, 1995.

[3] A. L. Blum und M. L. Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90(1-2):279–298, 1997.

[4] T. Bylander. The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69(1–2):165–204, 1994.

[5] M. Cadoli und F. M. Donini. A survey on knowledge compilation. *AI Communications*, 10(3,4):137–150, 1997.

[6] R. E. Fikes und N. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[7] M. R. Garey und D. S. Johnson. *Computers and Intractability—A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, CA, 1979.

[8] B. C. Gazen und C. Knoblock. Combining the expressiveness of UCPOP with the efficiency of Graphplan. In S. Steel and R. Alami, Hrsg., *Recent Advances in AI Planning. 4th European Conference on Planning (ECP'97)*, S. 221–233, Toulouse, France, Sept. 1997. Springer-Verlag.

[9] S. Kambhampati, E. Parker und E. Lambrecht. Understanding and extending Graphplan. In S. Steel and R. Alami, Hrsg., *Recent Advances in AI Planning. 4th European Conference on Planning (ECP'97)*, S. 260–272, Toulouse, France, Sept. 1997. Springer-Verlag.

[10] H. Kautz und B. Selman. The role of domain-specific knowledge in the planning as satisfiability framework. In R. Simmons, M. Veloso und S. Smith, Hrsg., *Proceedings of the 4th International Conference on Artificial Intelligence Planning Systems (AIPS-98)*, S. 181–189. AAAI Press, Menlo Park, 1998.

[11] H. A. Kautz und B. Selman. Forming concepts for fast inference. In *Proceedings of the 10th National Conference of the American Association*

for Artificial Intelligence (AAAI-92), S. 786–793, San Jose, CA, July 1992. MIT Press.

[12] J. Koehler, B. Nebel, J. Hoffmann und Y. Dimopoulos. Extending planning graphs to an ADL subset. In S. Steel and R. Alami, Hrsg., *Recent Advances in AI Planning. 4th European Conference on Planning (ECP'97)*, S. 273–285, Toulouse, France, Sept. 1997. Springer-Verlag.

[13] V. Lifschitz. On the semantics of STRIPS. In M. P. Georgeff and A. Lansky, Hrsg., *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop*, S. 1–9, Timberline, OR, June 1986. Morgan Kaufmann.

[14] B. Nebel. On the compilability and expressive power of propositional planning formalisms. Technical Report 101, Albert-Ludwigs-Universität, Institut für Informatik, Freiburg, Germany, June 1998.

[15] B. Nebel. Compilation schemes: A theoretical tool for assessing the expressive power of planning formalisms. In W. Burgard, A. B. Cremers und T. Christaller, Hrsg., *KI-99: Advances in Artificial Intelligence*, Bonn, Germany, 1999. Springer-Verlag. Erscheint demnächst.

[16] B. Nebel. What is the expressive power of disjunctive preconditions? In S. Biundo and M. Fox, Hrsg., *Recent Advances in AI Planning. 5th European Conference on Planning (ECP'99)*, Durham, UK, Sept. 1999. Springer-Verlag. Erscheint demnächst.

[17] C. H. Papadimitriou. *Computational Complexity.* Addison-Wesley, Reading, MA, 1994.

[18] E. P. Pednault. ADL: Exploring the middle ground between STRIPS and the situation calculus. In R. Brachman, H. J. Levesque und R. Reiter, Hrsg., *Principles of Knowledge Representation and Reasoning: Proceedings of the 1st International Conference (KR-89)*, S. 324–331, Toronto, ON, May 1989. Morgan Kaufmann.

[19] C. K. Yap. Some consequences of non-uniform conditions on uniform classes. *Theoretical Computer Science*, 26:287–300, 1983.