Albert-Ludwigs-Universität Freiburg
Fakultät für Angewandte Wissenschaften
Institut für Informatik

Wintersemester 2005/2006

Diplomarbeit

Erfüllbarkeitsbasierte Handlungsplanung mit temporal erweiterten Zielen

vorgelegt von
Robert Mattmüller

am 6. März 2006

Betreut von
PD Dr. Jussi Rintanen

Gutachter:
Prof. Dr. Bernhard Nebel,
PD Dr. Jussi Rintanen

Zusammenfassung

Handlungsplanung mit temporal erweiterten Zielen ist eine echte Verallgemeinerung klassischer Handlungsplanung, die es nicht nur erlaubt, die Eigenschaften eines Zielzustandes, sondern der gesamten Planausführung zu spezifizieren.

In der vorliegenden Arbeit stellen wir ein neues Verfahren zur Handlungsplanung mit temporal erweiterten Zielen durch aussagenlogische Erfüllbarkeitstests vor, das zur Erhöhung der Planungseffizienz parallele Operatoren erlaubt.

Das Verfahren lässt bei der Suche nach einem Plan, dessen Ausführung eine beliebige LTL-Formel ohne *Nexttime*-Operator erfüllt, parallele Operatoren zu, indem es in die aussagenlogische Kodierung des Planungsproblems eine Bedingung aufnimmt, die gewährleistet, dass es zu der parallelen Ausführung jedes Planes, der einer erfüllenden Belegung der Kodierung entspricht, mindestens eine zulässige Linearisierung gibt, die die gegebene LTL-Spezifikation genau dann erfüllt, wenn die parallele Ausführung dies tut.

In unseren Experimenten schnitt das Verfahren sowohl im Hinblick auf die Anzahlen von Zeitpunkten in der parallelen Planausführung als auch auf die Laufzeiten des Planers in fast allen Fällen mindestens so gut ab wie ein entsprechendes Verfahren, das nur sequentielle Pläne erzeugt. Im besten Fall war es dem sequentiellen Planer bezüglich beider Kriterien jeweils um den Faktor 2 überlegen.

Unsere Ergebnisse stellen einerseits eine Verallgemeinerung der von Rintanen, Heljanko und Niemelä [2005] angegebenen effizienten Kodierungen klassischer Handlungsplanung als aussagenlogische Erfüllbarkeitstests auf Planung mit temporal erweiterten Zielen dar. Andererseits kann das Verfahren zur Steigerung der Effizienz von symbolischen LTL-Modellprüfern dienen, die mit einer Übersetzung in das aussagenlogische Erfüllbarkeitsproblem arbeiten.

Abstract

Planning with temporally extended goals is a proper generalization of classical planning, allowing not only the specification of the properties of a goal state but also of the whole plan execution.

In this work we introduce a new method for planning with temporally extended goals as propositional satisfiability, using parallel operators to increase planning efficiency.

The method admits parallel operators in the search for a plan the execution of which satisfies an arbitrary LTL formula without *next time* operator by incorporating into the propositional encoding a condition ensuring that for the execution of each parallel plan there always exists an admissible linearization satisfying the given LTL specification if an only if the parallel execution does so.

In almost all of our experiments the new method performed at least as good as the corresponding method computing only sequential plans with respect to the number of time points in the parallel plan as well as with respect to the runtimes of the planner. In the best case the new method outperformed the sequential planner by a factor of 2 with respect to both criteria.

On the one hand our results constitute a generalization of the efficient encoding of classical planning as propositional satisfiability given by Rintanen, Heljanko and Niemelä [2005] to planning with temporally extended goals. On the other hand our encoding can serve for improving the efficiency of symbolic bounded model checking tools for LTL formulae using a reduction to propositional satisfiability.

Danksagungen

An dieser Stelle möchte ich all jenen danken, die durch ihre Unterstützung zum Gelingen dieser Arbeit beigetragen haben. Mein besonderer Dank gilt Herrn PD Dr. Jussi Rintanen für die hervorragende Betreuung dieser Arbeit, das Beisteuern vieler guter Ideen und die Bereitstellung des Quellcodes von SMLSATplanner. Bei Herrn Prof. Dr. Bernhard Nebel bedanke ich mich für die Möglichkeit, diese Arbeit an seinem Lehrstuhl anzufertigen, insbesondere für seine Bereitschaft, für diese Arbeit als Gutachter zur Verfügung zu stehen.

Schließlich möchte ich besonders meinen Eltern danken, die mir das Studium der Informatik und damit auch diese Diplomarbeit erst ermöglicht haben.

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe. Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, wurden als solche kenntlich gemacht. Die vorliegende Arbeit wurde nicht, auch nicht auszugsweise, bereits für eine andere Prüfung angefertigt.

Freiburg, den 6. März 2006. (Robert Mattmüller)

Inhaltsverzeichnis

Kapitel 1.

Einleitung

1.1. Motivation

1.1.1. Handlungsplanung

Die Erforschung automatisierter Handlungsplanung [Nau, Ghallab und Traverso, 2004] im Bereich der Künstlichen Intelligenz [Russell und Norvig, 2003] hat den Anspruch, Maschinen dazu zu befähigen, aus ihrem Wissen über ihr eigenes Können und über den relevanten Teil ihrer Umwelt sowie aus einer Zielvorgabe einen Plan zu entwickeln, dessen Ausführung das vorgegebene Ziel erreicht.

Planungsdomänen können sich unter anderem darin unterscheiden, wieviele Agenten betrachtet werden müssen, wieviel Wissen ein Agent zu jedem Zeitpunkt über seine Umwelt besitzt, wieviele verschiedene Zustände die Umwelt annehmen kann, wie zuverlässig die Folgen einer Aktion vorhergesagt werden können und welcher Art die Zielvorgabe für den Agenten ist. Die Vorgabe kann etwa sein, einen Plan zu finden, der die Welt in einen wünschenswerten Zielzustand versetzt (Erreichbarkeitsziel), oder einen Plan zu finden, dessen gesamte Ausführung, d. h. die Abfolge der Weltzustände, die sich während der Ausführung des Plans ergibt, bestimmte Eigenschaften besitzt.

Macht man an die Domäne, in der die Planung stattfinden soll, die Einschränkungen, dass die Umgebung, in der der Agent handelt, dem Agenten zu jedem Zeitpunkt vollständig bekannt, endlich, diskret und statisch, d. h. nur vom betrachteten Agenten selbst veränderbar, ist und die Aktionen deterministisch sind, so spricht man von klassischem Planen.

Man spricht von domänenspezifischem Planen, wenn die Domäne, der die Probleminstanzen angehören, im Voraus bekannt ist, und somit Planungsalgorithmen entwickelt werden können, die besondere Eigenschaften der Domäne bei der Suche nach Plänen berücksichtigen können. Im domänenunabhängigen Planen sind nur abstrakte Eigenschaften von Domänen wie etwa die Endlichkeit des Zustandsraums oder der Determinismus der Zustandsübergänge vorgegeben. Die Eingabe für einen Planungsalgorithmus besteht dann nicht nur aus einer Probleminstanz, sondern auch aus einer Beschreibung der Domäne, der die Instanz angehört. Eigenschaften der Domäne, die das Planen vereinfachen, wie etwa Symmetrien, müssen nun, um bei der Plansuche von Nutzen zu sein, von dem Planungsalgorithmus selbst erkannt werden und können nicht mehr vom Programmierer vorgegeben werden.

Im Weiteren befassen wir uns mit Ausnahme einer Verallgemeinerung des Zielbegriffs nur noch mit domänenunabhängigem klassischem Planen. Für dieses Problem gibt es eine Reihe von Verfahren, die sich grundlegend voneinander unterscheiden. Dazu gehören deduktives

Planen, d. h. Planen durch Theorembeweisen [Nau u. a., 2004, Kapitel 12], Planen mit Hilfe sogenannter Planungsgraphen [Blum und Furst, 1995], Planen als heuristische Suche im Zustandsraum [Bonet und Geffner, 2001], Planen im Planraum mit im Allgemeinen noch unfertigen, aus partiell geordneten Mengen von Aktionen bestehenden Plänen [Nau u. a., 2004, Kapitel 5] und Planen als Erfüllbarkeitstest [Kautz und Selman, 1992, 1996].

In dieser Arbeit soll der Ansatz von Planen mit aussagenlogischen Erfüllbarkeitstests weiterverfolgt werden. Insbesondere soll eine neue Kodierung von Planungsinstanzen als aussagenlogische Formeln gefunden werden, die einerseits erweiterte Zielvorgaben erlaubt und andererseits Pläne mit simultanen Aktionen zulässt.

1.1.2. Modellprüfung

Unter Modellprüfung *(Model Checking)* [Clarke, Grumberg und Peled, 2002] versteht man die automatisierte Überprüfung, ob eine gegebene endliche Struktur Modell einer gegebenen Formel ist. In der LTL-Modellprüfung will man entscheiden, ob für eine LTL-Formel φ die Formel $\mathbf{A}\varphi$ (oder $\mathbf{E}\varphi$) in einer Menge Q^{i} von Startzuständen eines gegebenen endlichen Transitionssystems $\mathfrak{M}$ gilt.

Häufig stellt das Transitionssystem eine vereinfachte Beschreibung eines Hardware- oder Softwaresystems dar. Kodiert man in der Formel φ, der Spezifikation, gewünschte Eigenschaften des Systems, etwa Sicherheits- oder Lebendigkeitseigenschaften *(safety/liveness properties)* [Manna und Pnueli, 1992], so kann man automatisch die Korrektheit des Systems überprüfen und ist nicht auf unvollständige Tests angewiesen.

Erfüllt das System die Spezifikation, so gibt ein Modellprüfungs-Algorithmus dies aus, erfüllt es die Spezifikation nicht, erzeugt der Algorithmus ein Gegenbeispiel, das zeigt, dass die Spezifikation verletzt ist. Im Fall der LTL-Modellprüfung ist ein Gegenbeispiel ein Pfad im Transitionssystem, der der Spezifikation widerspricht.

Man unterscheidet zwischen expliziten und symbolischen Modellprüfungs-Verfahren. Im expliziten Fall wird das Transitionssystem als Graph konstruiert und nach Gegenbeispielen zur Spezifikation durchsucht, während im symbolischen Fall die Transitionsrelation des Systems und die Spezifikation beispielsweise durch Binäre Entscheidungsdiagramme *(Binary Decision Diagrams, BDDs)* [Clarke u. a., 2002, Kapitel 5] oder aussagenlogische Formeln repräsentiert werden können. Dabei wird eine aussagenlogische Formel erzeugt, die genau dann erfüllbar ist, wenn es ein Gegenbeispiel vorgegebener Länge gibt. Da der Zustandsraum des Transitionssystems endlich ist, ist nicht nur die Korrektheit dieses Verfahrens garantiert – d. h. wird ein Gegenbeispiel gefunden, so ist die Spezifikation tatsächlich nicht erfüllt –, sondern auch dessen Vollständigkeit –, d. h. wird bis zu einer bestimmten Länge kein Gegenbeispiel gefunden, so ist die Spezifikation erfüllt.

Im Weiteren werden wir im Zusammenhang mit Handlungsplanung nur LTL-Modellprüfung mit Hilfe von aussagenlogischen Erfüllbarkeitstests betrachten.

1.1.3. Zusammenhang zwischen Handlungsplanung und Modellprüfung

Sowohl in der klassischen Handlungsplanung als auch bei der Modellprüfung sucht man nach Folgen von Zuständen mit vorgegebenen Eigenschaften. In der Handlungsplanung sind das solche Zustandsfolgen, die in einem bestimmten Zielzustand enden, in der Modellprüfung

solche Folgen, die ein Gegenbeispiel zu der vorgegebenen Spezifikation darstellen. Aufgrund dieser Gemeinsamkeit können einige Algorithmen aus der Modellprüfung auch in der Handlungsplanung eingesetzt werden und umgekehrt.

1.1.3.1. Handlungsplanung als Modellprüfung

Temporallogische Formeln können in der Handlungsplanung zu unterschiedlichen Zwecken eingesetzt werden. Erstens können mit ihnen sogenannte Steuerungsregeln *(control rules)* formuliert werden (vgl. z. B. [Nau u. a., 2004, Kapitel 10.6], die Arbeit von Bacchus und Kabanza [2000] über das TLplan-System oder die Arbeit von Doherty und Kvarnström [2001] über das TALplanner-System), die helfen, die Suche nach Plänen zu steuern und unter Verwendung von domänenspezifischen Informationen Teile des Zustandsraums von der Suche auszunehmen.

Zweitens kann man bei nichtdeterministischen Operatoren, partieller Beobachtbarkeit der Umgebung oder mehreren möglichen Startzuständen mit Hilfe von temporallogischen Formeln Bedingungen angeben, die besagen, wann ein Plan (eine *policy*) den gestellten Anforderungen entspricht. So kann etwa gefordert werden, dass ein Plan immer in endlich vielen Schritten einen Zielzustand erreicht, dass jeder Endzustand ein Zielzustand ist, oder dass der Plan nie in Zustände führt, von denen aus kein Zielzustand mehr erreichbar ist [Nau u. a., 2004, Kapitel 17].

Drittens kann man mit temporallogischen Formeln erweiterte Ziele *(extended goals)* [Nau u. a., 2004, Kapitel 10.6] definieren, d. h. Zielvorgaben an den Plan, die nicht nur spezifizieren, welche Eigenschaften der vom Plan erreichte Zielzustand zu besitzen hat, sondern auch, welche Eigenschaften der zum Ziel führende Pfad im Zustandsraum besitzt.

Solche erweiterten Ziele sind eine echte Verallgemeinerung der Erreichbarkeitsziele der klassischen Handlungsplanung. Jedes Erreichbarkeitsziel kann auch als erweitertes Ziel formuliert werden, nicht jedoch jedes erweiterte als Erreichbarkeitsziel. Das klassische Handlungsplanungsproblem kann unmittelbar auf das LTL-Modellprüfungs-Problem reduziert werden. Ist $\mathcal{P} = \langle A, I, O, g\rangle$ eine Planungsinstanz mit Zustandsvariablen A, Anfangszustand I, Operatorenmenge $O = \{ o_1, \ldots, o_n \}$ und Zielformel g, so kann man $\mathcal{P}$ in eine LTL-Modellprüfungs-Instanz $\mathfrak{M}, s \stackrel{?}{\models} \varphi$ übersetzen, wobei $\mathfrak{M} = \langle Q, R_1, \ldots, R_n, L\rangle$ ein Kripke-Modell ist. Seine Zustandsmenge ist $Q = 2^A$, die Transitionsrelationen R_i sind definiert durch sR_is' genau dann, wenn o_i in s anwendbar ist und seine Anwendung in den Zustand s' führt, und es ist $L(s) = s$ für alle s sowie $\varphi = \mathbf{G}\neg g$. Ein Plan für $\mathcal{P}$ entspricht dann genau einem Gegenbeispiel gegen die Spezifikation φ. Im Folgenden geben wir LTL-Spezifikationen immer so an, dass ein erfüllender Pfad statt eines Gegenbeispiels gesucht wird, hier etwa $\varphi = \mathbf{F}g$ statt $\varphi = \mathbf{G}\neg g$.

Neben LTL-Formeln der Gestalt $\mathbf{F}\Phi$ für aussagenlogische Formeln Φ sind einige weitere einfache LTL-Formeln von Interesse. Sucht man etwa nach einem Plan, der bzw. dessen *unendliche* Ausführung ein Aufrechterhaltungsziel *(maintenance goal)* [Rintanen, 2004] erfüllt, d. h. der garantiert, dass die Umgebung beliebig lange in einer Menge von Zielzuständen bleibt, so kann man einen solchen Plan dadurch finden, dass man die Spezifikation $\mathbf{G}g$ vorgibt, wobei g die Eigenschaft kodiert, die aufrechterhalten werden soll bzw. durch die die Menge der Zielzustände charakterisiert ist. Fordert man nicht, dass das System schon zu Beginn in einem Zielzustand ist, sondern erlaubt man, dass die Zielzustandsmenge erst in endlich

vielen Schritten erreicht wird, so kann man statt $\mathbf{G}g$ auch $\mathbf{FG}g$ spezifizieren. Vergleichbar mit Aufrechterhaltungszielen sind Sicherheitsziele, die etwa in Verbindung mit einem Erreichbarkeitsziel angegeben werden können. So kann man durch eine Formel der Gestalt $\mathbf{F}g \wedge \bigwedge_{i=1}^{k}(\Phi_i \rightarrow \mathbf{G}\Phi_i)$ mit aussagenlogischen Formeln g, Φ_i, $i = 1, \ldots, k$, fordern, dass der Agent einen Zustand herbeiführt, in dem g gilt, ohne dabei gefährliche Folgen von Aktionen auszuführen, d. h. solche Aktionenfolgen, die in einen Zustand führen, in dem eine der Formeln Φ_i, die im Anfangszustand noch gilt, nicht mehr wahr ist.[1] Ein weiterer interessanter Typ von LTL-Formeln sind solche der Gestalt $\mathbf{F}(\Phi_1 \wedge \mathbf{F}(\Phi_2 \wedge \ldots \mathbf{F}\Phi_n \ldots))$ mit aussagenlogischen Formeln $\Phi_1, \ldots, \Phi_n$. Damit kann spezifiziert werden, dass zuerst das Ziel Φ_1 erreicht werden soll, danach Φ_2 usw. bis schließlich das Ziel Φ_n erreicht wird.

1.1.3.2. Modellprüfung als Handlungsplanung

Es ist nicht nur möglich, Planungsaufgaben als Modellprüfungs-Instanzen zu betrachten, sondern auch umgekehrt, wenn man den Bereich des klassischen Planens verlässt und erweiterte Zielspezifikationen zulässt. In der Arbeit von Edelkamp [2003] wird beschrieben, wie die Sprache PDDL (Planning Domain Definition Language) erweitert werden kann, um dies zu ermöglichen, wie die Verifikation von Kommunikationsprotokollen in PDDL kodiert werden kann und welche Möglichkeiten und Grenzen die PDDL-Modellierung von Modellprüfungs-Instanzen mit sich bringt. Auch von Gerevini und Long [2005] wird eine entsprechende Erweiterung von PDDL beschrieben.

1.2. Verwandte Arbeiten

Handlungsplanung durch Übersetzung in das aussagenlogische Erfüllbarkeitsproblem wurde erstmals von Kautz und Selman [1992, 1996] beschrieben. Eine ausführliche Würdigung dieses Ansatzes einschließlich einer effizienten Verallgemeinerung auf parallele Pläne findet sich bei Rintanen, Heljanko und Niemelä [2004, 2005].

Eine Kodierung des LTL-Modellprüfungs-Problems in Aussagenlogik ohne Berücksichtigung möglicher Parallelitäten wird von Biere, Cimatti, Clarke und Zhu [1999] vorgestellt. Die Kodierung, an der wir uns in dieser Arbeit orientieren, stammt von Latvala, Biere, Heljanko und Junttila [2004].

Der für diese Arbeit zentrale Begriff des Stotterns *(stuttering)* bzw. der Äquivalenz von Pfaden *(stuttering equivalence)* wird erstmals von Lamport [1983] gebraucht. Der Zusammenhang zwischen temporallogischen Sprachen und der *Invarianz unter Stottern* in diesen Sprachen ausdrückbarer Eigenschaften wird in der Arbeit von Peled und Wilke [1997] genauer untersucht. Die Frage, wann Transitionen voneinander unabhängig sind und somit parallel durchgeführt werden dürfen, wird in der Arbeit von Godefroid [1996] diskutiert.

Einen Überblick über Handlungsplanung gibt das Lehrbuch von Nau, Ghallab und Traverso [2004], einen über Modellprüfung das von Clarke, Grumberg und Peled [2002].

[1] In der Arbeit von Weld und Etzioni [1994] wird als Beispiel ein Software-Agent genannt, der das Ziel hat, den benutzten Anteil einer Festplatte auf weniger als 90% zu verringern, dabei jedoch bestimmte unersetzliche Dateien nicht löschen darf.

1.3. Überblick

Diese Arbeit ist wie folgt gegliedert: In Kapitel 2 werden die wichtigsten Begriffe aus Handlungsplanung und Modellprüfung wiederholt. In Kapitel 3 wird beschrieben, wie symbolische Modellprüfung auf einem Ausführungspfad eines Planes betrieben werden kann, selbst wenn wegen paralleler Aktionen nicht alle Zustände auf dem Pfad explizit repräsentiert und damit der Modellprüfung zugänglich sind. In Kapitel 4 wird eine Implementierung der zuvor beschriebenen Idee vorgestellt. Kapitel 5 enthält eine Zusammenfassung.

Der Beitrag dieser Arbeit besteht darin, eine einfache Möglichkeit aufzuzeigen, wie symbolische Modellprüfung durch aussagenlogische Erfüllbarkeitstests mit einer effizienten aussagenlogischen Kodierung paralleler Operatoranwendungen kombiniert werden kann. Dies ermöglicht einerseits eine effiziente Durchführung von beschränkter Modellprüfung *(Bounded Model Checking)* durch das Zulassen paralleler Transitionen, andererseits die Anwendung des von Rintanen, Heljanko und Niemelä [2005] beschriebenen effizienten Planungsverfahrens mit parallelen Aktionen auf verallgemeinerte Planungsprobleme mit erweiterten Zielspezifikationen.

Kapitel 2.

Grundlagen und Definitionen

In diesem Kapitel sollen zum einen die verwendete Notation geklärt (Abschnitt 2.1), zum anderen grundlegende Definitionen und Ergebnisse aus der Handlungsplanung (Abschnitt 2.3) und der Modellprüfung (Abschnitt 2.4) wiederholt werden. Die sowohl für Handlungsplanung als auch Modellprüfung relevanten Kripke-Rahmen werden in Abschnitt 2.2 beschrieben.

2.1. Notation

Die in dieser Arbeit verwendete Notation orientiert sich weitgehend an den üblichen Schreibweisen. Dennoch sollen kurz die wichtigsten Symbole erklärt werden.

Unter $\mathbb{N}$ verstehen wir die Menge der **natürlichen Zahlen** *einschließlich* der Null.

Ist X eine Menge, so ist 2^X die Potenzmenge von X und $|X|$ ihre Mächtigkeit. Der Definitions- bzw. Wertebereich einer Relation f wird als $\mathrm{dom}(f) := \{\, a \mid \exists b : (a,b) \in f \,\}$ bzw. $\mathrm{ran}(f) := \{\, b \mid \exists a : (a,b) \in f \,\}$ notiert. Die Schreibweise $f \oplus g$ steht für die Relation bzw. Funktion, die auf dem Definitionsbereich von g gleich g ist und sich sonst wie f verhält, d. h. $(f \setminus (\mathrm{dom}(g) \times \mathrm{ran}(f))) \cup g$, die Notation $f \restriction_A$ für f eingeschränkt auf $A \subseteq \mathrm{dom}(f)$, d. h. $\{\, (a,b) \in f \mid a \in A \,\}$.

Unter einer **partiellen** bzw. **totalen Ordnungsrelation** verstehen wir hier immer eine *strenge* Ordnungsrelation, d. h. eine *irreflexive, transitive* und ggf. *trichotomische (totale)* Relation im Sinne von $<$.

Die Menge aller **aussagenlogischen Formeln** bezeichnen wir mit PL. PL-Formeln werden von einer Menge A von aussagenlogischen Variablen und den Konstanten $\top$ (wahr) und $\bot$ (falsch) ausgehend mit den Symbolen $\neg$, $\wedge$, $\vee$, $\rightarrow$ und $\leftrightarrow$ gebildet, d. h.

$$\mathsf{PL} ::= A \mid \top \mid \bot \mid \neg\mathsf{PL} \mid \mathsf{PL} \wedge \mathsf{PL} \mid \mathsf{PL} \vee \mathsf{PL} \mid \mathsf{PL} \rightarrow \mathsf{PL} \mid \mathsf{PL} \leftrightarrow \mathsf{PL}.$$

Die Länge einer Formel Φ bezeichnen wir mit $\|\Phi\|$.

Eine **Variablenbelegung** für eine Menge A von aussagenlogischen Variablen ist eine Abbildung $v : A \rightarrow \{\, 0,1 \,\}$. Um die Darstellung übersichtlicher zu halten, identifizieren wir eine Abbildung $f : X \rightarrow \{\, 0,1 \,\}$ mit $X_f = \{\, x \in X \mid f(x) = 1 \,\} \in 2^X$. Dies bedeutet insbesondere, dass wir Mengen $A' \subseteq A$ von Variablen mit ihren charakteristischen Funktionen $\chi_{A'} : A \rightarrow \{\, 0,1 \,\}$, $\chi_{A'}(a) = 1$, wenn $a \in A'$, und $\chi_{A'}(a) = 0$, sonst, die als Belegungen gedeutet werden können, identifizieren. Darüber hinaus identifizieren wir für endliches A an einigen Stellen eine Belegung $v : A \rightarrow \{\, 0,1 \,\}$ mit der Formel $\bigwedge_{a \in A : v(a) = 1} a \wedge \bigwedge_{a \in A : v(a) = 0} \neg a$.

Ein **Literal** ist eine aussagenlogische Formel der Gestalt a oder $\neg a$ für $a \in A$. Das zu ℓ **komplementäre** Literal $\overline{\ell}$ ist definiert als $\overline{a} = \neg a$ und $\overline{\neg a} = a$ für $a \in A$. Ist A eine Menge von Aussagenvariablen, so sei $Lit(A) := A \cup \{ \neg a \mid a \in A \}$ die Menge aller Literale über A. Eine **Klausel** ist eine Disjunktion $\ell_1 \vee \cdots \vee \ell_n$ von Literalen.

Wenn eine Variablenbelegung v eine aussagenlogische Formel Φ **erfüllt**, so notieren wir dies kurz als $v \models \Phi$. Eine Formel Φ heißt **erfüllbar**, wenn es eine Variablenbelegung v mit $v \models \Phi$ gibt. Φ heißt **allgemeingültig**, kurz $\models \Phi$, wenn $\neg\Phi$ nicht erfüllbar ist. Wir schreiben $\Phi_1 \equiv \Phi_2$, wenn Φ_1 und Φ_2 logisch äquivalent sind, d. h. wenn für jede Variablenbelegung v gilt, dass $v \models \Phi_1$ genau dann, wenn $v \models \Phi_2$. Die von einer Variablenbelegung v erfüllten Literale bezeichnen wir mit $l(v) := \{ \ell \in Lit(A) \mid v \models \ell \}$.

Die **positiv und negativ in Φ vorkommenden Variablen** sind induktiv wie folgt definiert, wobei a für eine atomare Formel steht:

$$
\begin{aligned}
pos(a) &:= \{ a \} & neg(a) &:= \varnothing \\
pos(\Phi_1 \wedge \Phi_2) &:= pos(\Phi_1) \cup pos(\Phi_2) & neg(\Phi_1 \wedge \Phi_2) &:= neg(\Phi_1) \cup neg(\Phi_2) \\
pos(\Phi_1 \vee \Phi_2) &:= pos(\Phi_1) \cup pos(\Phi_2) & neg(\Phi_1 \vee \Phi_2) &:= neg(\Phi_1) \cup neg(\Phi_2) \\
pos(\neg\Phi) &:= neg(\Phi) & neg(\neg\Phi) &:= pos(\Phi)
\end{aligned}
$$

Die Fälle der Konnektive $\rightarrow$ und $\leftrightarrow$ können auf die obigen Fälle zurückgeführt werden. Für die Menge aller in Φ vorkommenden Variablen schreiben wir $var(\Phi) := pos(\Phi) \cup neg(\Phi)$.

Um die Definition positiv und negativ vorkommender Variablen auf Literale zu verallgemeinern, schreiben wir für $a \in A$:

$$
\begin{aligned}
&\mathrm{pos}(a, \Phi) \text{ gdw. } a \in pos(\Phi), && && \mathrm{neg}(a, \Phi) \text{ gdw. } a \in neg(\Phi), \\
&\mathrm{pos}(\neg a, \Phi) \text{ gdw. } a \in neg(\Phi) && \text{und} && \mathrm{neg}(\neg a, \Phi) \text{ gdw. } a \in pos(\Phi).
\end{aligned}
$$

2.2. Kripke-Rahmen

Sowohl in der Handlungsplanung als auch bei der Modellprüfung hat man es mit Systemen von Zuständen und Übergängen zwischen den Zuständen zu tun. Eine für unsere Zwecke hinreichend allgemeine Formalisierung dieser Systeme bietet der Begriff eines Kripke-Rahmens bzw. eines Kripke-Modells [Blackburn, de Rijke und Venema, 2001; Clarke u. a., 2002].

Definition 1 (Kripke-Rahmen). Ein **Kripke-Rahmen** ist ein Tupel $\mathfrak{F} = \langle Q, R_1, \ldots, R_n \rangle$ für $n \in \mathbb{N}$ mit einer Menge Q von Zuständen und n Transitionsrelationen $R_i \subseteq Q \times Q$ für $i \in \{ 1, \ldots, n \}$.

Für uns ist ein **Zustand** immer eine Belegung $s : A \rightarrow \{ 0, 1 \}$ der Variablen aus einer Menge A, die in diesem Zusammenhang auch als **Zustandsvariable** bezeichnet werden.

Definition 2 (Pfad). Sei $\mathfrak{F} = \langle Q, R_1, \ldots, R_n \rangle$ ein Kripke-Rahmen. Eine Folge $\pi : \mathbb{N} \rightarrow Q$ heißt **unendlicher Pfad** in $\mathfrak{F}$, falls für alle $k \in \mathbb{N}$ ein $i \in \{ 1, \ldots, n \}$ existiert, so dass $\pi(k) R_i \pi(k+1)$. Ist π nur auf einem Anfangsabschnitt $\{ 0, \ldots, B-1 \} \subset \mathbb{N}$ der natürlichen Zahlen definiert, so muss die Forderung $\pi(k) R_i \pi(k+1)$ nur für $k \in \{ 0, \ldots, B-2 \}$ erfüllt sein und man spricht von einem **endlichen Pfad** der Länge B. Der **leere Pfad** der Länge 0 wird mit ε bezeichnet.

Ist π ein unendlicher Pfad, so ist $\pi^i : \mathbb{N} \to Q$ definiert durch $\pi^i(j) := \pi(i+j)$. Ist π ein endlicher Pfad der Länge B und $i \in \mathbb{N}$, so ist $\pi^i = \varepsilon$, falls $i \geq B$, und $\pi^i : \{ 0, \ldots, B-i-1 \} \to Q$ mit $\pi^i(j) := \pi(i+j)$ für alle $j \in \{ 0, \ldots, B-i-1 \}$, falls $i < B$.

Sei $\pi = \pi(0), \ldots, \pi(B-1)$ ein endlicher Pfad der Länge B und π' ein endlicher oder unendlicher Pfad. Dann ist π^ω der eindeutig bestimmte unendliche Pfad mit $\pi^\omega(i) = \pi(m)$ für alle $i \in \mathbb{N}$, wobei $0 \leq m \leq B-1$ und $m \equiv i \pmod{B}$. Außerdem ist $\pi \circ \pi'$ der endliche oder unendliche Pfad, der durch Konkatenation von π und π' entsteht, d. h.

$$(\pi \circ \pi')(i) = \begin{cases} \pi(i), & \text{falls } 0 \leq i \leq B-1 \\ \pi'(i-B), & \text{falls } i \geq B. \end{cases}$$

Definition 3 (Kripke-Modell). Sei A eine endliche Menge von aussagenlogischen Variablen und $\mathfrak{F} = \langle Q, R_1, \ldots, R_n \rangle$ ein Kripke-Rahmen. Ein **Kripke-Modell** über $\mathfrak{F}$ ist ein Tupel $\mathfrak{M} = \langle \mathfrak{F}, L \rangle$, wobei $L : Q \to 2^A$ eine Funktion ist, die jedem Zustand eine Belegung der Aussagenvariablen in A zuordnet.

2.3. Handlungsplanung

Dieser Abschnitt gibt einen kurzen Überblick über automatische Handlungsplanung im allgemeinen (Teilabschnitte 2.3.1 und 2.3.2) und über Handlungsplanung durch Übersetzung in das aussagenlogische Erfüllbarkeitsproblem im Besonderen (Teilabschnitt 2.3.3). Mit den Definitionen, Sätzen und Beweisen in den Teilabschnitten 2.3.1, 2.3.2 und 2.3.3 folgen wir weitgehend der Arbeit von Rintanen u. a. [2005].

2.3.1. Überblick

Definition 4 (Operator). Ein **Operator** auf einer Menge A von Zustandsvariablen ist ein Tripel $o = \langle p, e, c \rangle$. Dabei ist

1. p eine aussagenlogische Formel über A (die **Vorbedingung**),
2. e eine Menge von Literalen über A (die **unbedingten Effekte**) und
3. c eine Menge von Paaren $f \rhd d$ (die **bedingten Effekte**), wobei f eine aussagenlogische Formel über A und d eine Menge von Literalen über A ist.

Die Menge aller Effekte eines Operators $o = \langle p, e, c \rangle$ ist

$$[o]_\Diamond := e \cup \bigcup \{ d \mid f \rhd d \in c \}.$$

Analog dazu bezeichnen wir die Menge e der unbedingten Effekte von o gelegentlich auch mit $[o]_\Box$. Die Menge der in einem Zustand s **aktiven Effekte** von $o = \langle p, e, c \rangle$ ist

$$[o]_s := e \cup \bigcup \{ d \mid f \rhd d \in c \text{ und } s \models f \}.$$

Der Operator $o = \langle p, e, c \rangle$ ist in s **anwendbar**, falls $s \models p$ und $[o]_s$ konsistent ist, d. h. wenn es kein $a \in A$ mit $\{ a, \neg a \} \subseteq [o]_s$ gibt. Ist o anwendbar, dann ist $app_o(s)$ der eindeutig bestimmte Zustand, den man von s aus erreicht, indem man alle Literale in $[o]_s$ wahr macht und die Wahrheitswerte aller Variablen, die nicht in $[o]_s$ vorkommen, beibehält, d. h.

$app_o(s) := s \oplus (\{\,(a,1) \mid a \in [o]_s\,\} \cup \{\,(a,0) \mid \neg a \in [o]_s\,\})$. Ist $o_1; o_2; \ldots; o_n$ eine Folge von Operatoren, so ist $app_{o_1;o_2;\ldots;o_n}(s)$ definiert als der Zustand s_n, den man erhält, wenn $s_0 := s$ und $s_i := app_{o_i}(s_{i-1})$ für alle $i \in \{\,1,\ldots,n\,\}$. Ist S eine Menge von Operatoren und s ein Zustand, dann ist $app_S(s)$ der Zustand, der sich aus s durch *simultane* Anwendung aller Operatoren $o \in S$ ergibt, d. h. $app_S(s) := s \oplus (\{\,(a,1) \mid a \in [S]_s\,\} \cup \{\,(a,0) \mid \neg a \in [S]_s\,\})$ mit $[S]_s := \bigcup_{o \in S} [o]_s$. Dazu muss $app_o(s)$ für alle $o \in S$ definiert und die Menge $[S]_s$ der in s aktiven Effekte aller Operatoren in S konsistent sein. Ein Operator $o = \langle p, e, c\rangle$ heißt **STRIPS-Operator**, wenn $c = \varnothing$ und p eine Konjunktion von Literalen ist.

Definition 5 (Effektvorbedingung). Ist $o = \langle p, e, c\rangle$ ein Operator und ℓ ein atomarer Effekt der Form a oder $\neg a$ für ein $a \in A$, so ist die **Effektvorbedingung** von ℓ unter dem Operator o definiert als die aussagenlogische Formel

$$EPC_\ell(o) := \begin{cases} \top, & \text{falls } \ell \in e \\ \bigvee \{\, f \mid f \rhd d \in c \text{ und } \ell \in d\,\}, & \text{sonst.} \end{cases}$$

Lemma 1. *Sei ℓ ein Literal, o ein Operator und s ein Zustand. Dann ist $\ell \in [o]_s$ genau dann, wenn $s \models EPC_\ell(o)$.* □

Definition 6 (Planungsinstanz). Eine **Planungsinstanz** ist ein Tupel $\mathcal{P} = \langle A, I, O, g\rangle$, wobei A eine endliche Menge von Zustandsvariablen, I der Startzustand, O eine endliche Menge von Operatoren und g eine aussagenlogische Formel, die Zielformel, ist. Ein Zustand $s : A \to \{\,0,1\,\}$ ist genau dann ein Zielzustand, wenn $s \models g$.

Ist $\mathcal{P} = \langle A, I, O, g\rangle$ eine Planungsinstanz mit $O = \{\,o_1,\ldots,o_n\,\}$, so kann man einen von $\mathcal{P}$ **induzierten Kripke-Rahmen** $\mathfrak{F}(\mathcal{P}) = \langle Q, R_1, \ldots, R_n\rangle$ definieren durch $Q = 2^A$ und dadurch, dass für alle $s, s' \in Q$ und $i \in \{\,1,\ldots,n\,\}$ gilt: sR_is' genau dann, wenn $app_{o_i}(s)$ definiert und gleich s' ist.

Definition 7 (Sequentieller Plan). Ein **sequentieller Plan** für eine Planungsinstanz $\mathcal{P} = \langle A, I, O, g\rangle$ ist eine Folge $\Pi = o_1; o_2; \ldots; o_n$ von Operatoren in O, so dass $app_\Pi(I) \models g$, d. h. dass die Anwendung der Operatoren in der gegebenen Reihenfolge definiert und der resultierende Zustand ein Zielzustand ist.

Definition 8 (Ausführung eines sequentiellen Planes). Sei $\mathcal{P} = \langle A, I, O, g\rangle$ eine Planungsinstanz, $s \in Q$ und $\Pi = o_1; \ldots; o_n$ eine Folge von Operatoren aus O, so dass $app_{o_1;o_2;\ldots;o_n}(s)$ definiert und gleich s' ist. Die im Zustand s beginnende **Ausführung** von Π ist dann der Pfad $\pi = s_0, s_1, \ldots, s_n$ in $\mathfrak{F}(\mathcal{P})$ mit $s_0 = s$, $s_n = s'$ und $s_i = app_{o_i}(s_{i-1})$ für alle $i \in \{\,1,\ldots,n\,\}$.

Beispiel 9. Sei $\mathcal{P} = \langle A, I, O, g\rangle$, wobei $A = \{\,a_1, a_2, a_3\,\}$, $I = \{\,a_1 \mapsto 0, a_2 \mapsto 0, a_3 \mapsto 0\,\}$, $O = \{\,o_1, o_2, o_3, o_4\,\}$ mit

$$o_1 = \langle \neg a_1, \{\,a_1\,\}, \varnothing\rangle, \qquad o_2 = \langle \neg a_2, \{\,a_2\,\}, \{\,a_1 \rhd \{\,a_3\,\}\,\}\rangle,$$
$$o_3 = \langle a_1 \wedge \neg a_2, \{\,a_2, \neg a_3\,\}, \varnothing\rangle, \qquad o_4 = \langle \neg a_3, \{\,a_3\,\}, \varnothing\rangle$$

und $g = a_1 \wedge a_2 \wedge a_3$. Unter den Operatoren sind o_1, o_3 und o_4 STRIPS-Operatoren. Operator o_2 besitzt die unbedingten Effekte $[o_2]_\square = \{\,a_2\,\}$, die Menge aller Effekte $[o_2]_\lozenge = \{\,a_2, a_3\,\}$ und die Menge der in I aktiven Effekte $[o_2]_I = \{\,a_2\,\}$ (wegen $I \not\models a_1$). Die Operatoren o_1, o_2 und o_4 sind in I simultan anwendbar, denn wegen der erfüllten Vorbedingungen ist jeder einzelne Operator anwendbar und darüber hinaus sind die aktiven Effekte der Operatoren konsistent.

Identifiziert man einen Zustand, d. h. eine Belegung s der Variablen a_1, a_2 und a_3, mit der Zeichenkette $s(a_1)s(a_2)s(a_3)$, so kann man den von $\mathcal{P}$ induzierten Kripke-Rahmen $\mathfrak{F}(\mathcal{P}) = \langle Q, R_1, R_2, R_3, R_4 \rangle$ wie in Abbildung 2.1 visualisieren. Dabei ist $I = 000$ der Anfangszustand und 111 der einzige Zielzustand, da nur $111 \models g$. Formal ist $Q = 2^A$,

$$
\begin{aligned}
R_1 &= \{\, (000, 100), (010, 110), (001, 101), (011, 111) \,\}, \\
R_2 &= \{\, (000, 010), (001, 011), (100, 111), (101, 111) \,\}, \\
R_3 &= \{\, (100, 110), (101, 110) \,\} \text{ und} \\
R_4 &= \{\, (000, 001), (100, 101), (010, 011), (110, 111) \,\}.
\end{aligned}
$$

Sequentielle Pläne für $\mathcal{P}$ sind unter anderem $\Pi_1 = o_2; o_1; o_4$ und $\Pi_2 = o_1; o_2$. Ihre Ausführungen sind $\pi_1 = 000, 010, 110, 111$ bzw. $\pi_2 = 000, 100, 111$.

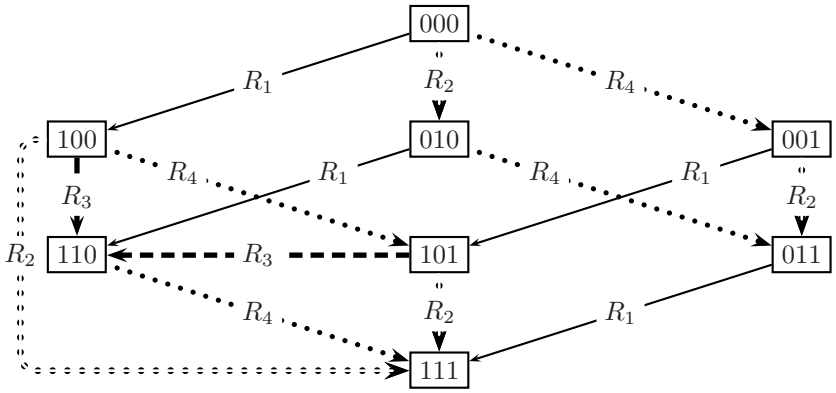


Abbildung 2.1.: Von Planungsinstanz aus Beispiel 9 induzierter Kripke-Rahmen

2.3.2. Parallele Pläne

Häufig kann es vorkommen, dass in einem Zustand s mehrere Aktionen ausgeführt werden müssen, um einen Zustand s' zu erreichen, es dabei aber unerheblich ist, in welcher Reihenfolge dies geschieht. Um den Planungsaufwand zu verringern, kann man deshalb nach Plänen suchen, in denen Operatoren parallel angewandt werden dürfen. Der Begriff einer Planungsinstanz ist der gleiche wie im Fall sequentiellen Planens, Pläne müssen nun jedoch allgemeiner definiert werden. Hier werden lediglich sogenannte 1-Linearisierungs-Pläne betrachtet. Weitere Arten von parallelen Plänen werden von Rintanen u. a. [2005] beschrieben.

Bei der Berechnung eines 1-Linearisierungs-Plans lässt man zu, dass während des Planens die Reihenfolge, in der eine Menge von Operatoren angewandt wird, unberücksichtigt bleibt, wenn es *mindestens eine* Reihenfolge gibt, in der die betreffenden Operatoren tatsächlich angewandt werden können. Beachte, dass es nicht möglich sein muss, die Operatoren in *jeder* Reihenfolge anzuwenden, und dass in dem Fall, dass sie in mehr als einer Reihenfolge angewandt werden können, die Ausführung abhängig von der Reihenfolge in unterschiedlichen Zuständen resultieren kann.

Definition 10 (1-Linearisierungs-Plan). Sei $\mathcal{P} = \langle A, I, O, g\rangle$ eine Planungsinstanz. Ein **1-Linearisierungs-Plan** der Länge b für $\mathcal{P}$ ist ein Tupel $\Pi = \langle S_0, \ldots, S_{b-1}\rangle$ mit $S_t \in 2^O$ für $t \in \{0, \ldots, b-1\}$ zusammen mit einer Folge $\pi = s_0, \ldots, s_b$ von Zuständen, der **Ausführung** von Π, so dass

1. $s_0 = I$ und
2. für jedes $t \in \{0, \ldots, b-1\}$ existiert eine totale Ordnung $o_{t,1} \prec \cdots \prec o_{t,|S_t|}$ von S_t, so dass $s_{t+1} = app_{o_{t,1};\ldots;o_{t,|S_t|}}(s_t)$.

Eine solche Folge

$$\tilde{\Pi} = o_{0,1}; o_{0,2}; \ldots; o_{0,|S_0|}; o_{1,1}; o_{1,2}; \ldots; o_{1,|S_1|}; \ldots; o_{b-1,1}; o_{b-1,2}; \ldots; o_{b-1,|S_{b-1}|},$$

dass für jedes $t \in \{0, \ldots, b-1\}$ $app_{o_{t,1};\ldots;o_{t,|S_t|}}(s_t)$ definiert und gleich s_{t+1} ist, heißt **zulässige Linearisierung** von Π. Beachte, dass es für einen 1-Linearisierungs-Plan $\Pi = \langle S_0, \ldots, S_{b-1}\rangle$ mit Ausführung $s_0, \ldots, s_b$ im Allgemeinen mehr als eine zulässige Linearisierung gibt.

Definition 11 (Sequentielle Ausführung eines parallelen Planes). Sei $\Pi = \langle S_0, \ldots, S_{b-1}\rangle$ ein 1-Linearisierungs-Plan für $\mathcal{P} = \langle A, I, O, g\rangle$ mit Ausführung $\pi = s_0, \ldots, s_b$. Eine **sequentielle Ausführung** von Π bzgl. π ist ein Pfad $\tilde{\pi}$, der die Ausführung einer zulässigen Linearisierung von Π ist.

Lemma 2. *Sei $\Pi = \langle S_0, \ldots, S_{b-1}\rangle$ ein 1-Linearisierungs-Plan für $\mathcal{P} = \langle A, I, O, g\rangle$ mit Ausführung $\pi = s_0, \ldots, s_b$. Ein Pfad $\tilde{\pi}$ ist eine sequentielle Ausführung von Π bzgl. π genau dann, wenn $\tilde{\pi}$ die Form*

$$s_0, q_{0,1}, \ldots, q_{0,|S_0|-1}, s_1, q_{1,1}, \ldots, q_{1,|S_1|-1}, s_2, \ldots, s_{b-1}, q_{b-1,1}, \ldots, q_{b-1,|S_{b-1}|-1}, s_b$$

hat, wobei

1. *$s_0 = I$ und*
2. *für jedes $t \in \{0, \ldots, b-1\}$ eine totale Ordnung $o_{t,1} \prec \ldots \prec o_{t,|S_t|}$ von S_t existiert, so dass gilt:*
   a) *$q_{t,1} = app_{o_{t,1}}(s_t)$,*
   b) *$q_{t,j+1} = app_{o_{t,j+1}}(q_{t,j})$ für alle $j \in \{1, \ldots, |S_t| - 2\}$ und*
   c) *$s_{t+1} = app_{o_{t,|S_t|}}(q_{t,|S_t|-1})$.*

Beweis. Angenommen, $\tilde{\pi}$ ist eine sequentielle Ausführung von Π bzgl. π. Dann gibt es eine zulässige Linearisierung

$$\tilde{\Pi} = o_{0,1}; o_{0,2}; \ldots; o_{0,|S_0|}; o_{1,1}; o_{1,2}; \ldots; o_{1,|S_1|}; \ldots; o_{b-1,1}; o_{b-1,2}; \ldots; o_{b-1,|S_{b-1}|}$$

von Π, so dass für alle $t \in \{0, \ldots, b-1\}$ gilt: $s_{t+1} = app_{o_{t,1};o_{t,2};\ldots;o_{t,|S_t|}}(s_t)$. Definiert man $q_{t,1} = app_{o_{t,1}}(s_t)$ und $q_{t,j+1} = app_{o_{t,j+1}}(q_{t,j})$ für alle $j \in \{1, \ldots, |S_t| - 2\}$, so ist $q_{t,|S_t|-1} = app_{o_{t,1};\ldots;o_{t,|S_t|-1}}(s_t)$. Also ist auch

$$s_{t+1} = app_{o_{t,1};\ldots;o_{t,|S_t|}}(s_t) = app_{o_{t,|S_t|}}(app_{o_{t,1};\ldots;o_{t,|S_t|-1}}(s_t)) = app_{o_{t,|S_t|}}(q_{t,|S_t|-1}).$$

Die Ausführung $\tilde{\pi}$ von $\tilde{\Pi}$ hat also die gewünschte Form. $s_0 = I$ gilt nach Voraussetzung.

Sei nun ein Pfad $\tilde{\pi}$ wie oben mit den beschriebenen Eigenschaften gegeben. $\tilde{\pi}$ ist eine sequentielle Ausführung von Π bzgl. π, denn nach Voraussetzung ist für alle $t \in \{0, \ldots, b-1\}$ der Zustand $s_{t+1} = app_{o_{t,1};\ldots;o_{t,|S_t|}}(s_t)$ für die von $\tilde{\pi}$ gegebene totale Ordnung $o_{t,1} \prec \cdots \prec o_{t,|S_t|}$ von S_t. Dass $s_0 = I$ ist, gilt nach Voraussetzung. □

Beispiel 12. In der Planungsinstanz aus Beispiel 9 ist $\Pi_4 = \langle\{ o_1, o_2 \}\rangle$ zusammen mit der Ausführung $\pi_4 = 000, 111$ ein 1-Linearisierungs-Plan der Länge 1. Seine einzige zulässige Linearisierung ist $\tilde{\Pi}_4 = o_1; o_2$, die entsprechende sequentielle Ausführung ist $\tilde{\pi}_4 = 000, 100, 111$.

2.3.3. Planen durch Übersetzung in Aussagenlogik

In ihrer Arbeit aus dem Jahr 1992 schlagen Kautz und Selman vor, automatisiertes Planen auf der Grundlage von aussagenlogischen Erfüllbarkeitsprüfungen anstelle von logischer Deduktion durchzuführen. Dieser Ansatz, der im einfachsten Fall auf sequentielle Pläne beschränkt ist, kann auch auf den Fall von parallelen Plänen verallgemeinert werden. Einen Überblick gibt die Arbeit von Rintanen, Heljanko und Niemelä [2005], in der auch neue effiziente Kodierungen vorgestellt werden. In diesem Abschnitt werden die grundlegenden Ideen sowie die von den genannten Autoren vorgestellten Kodierungen in Aussagenlogik beschrieben.

Planen mit Hilfe aussagenlogischer Erfüllbarkeitstests kann erfolgen, indem man eine solche Folge von aussagenlogischen Formeln $\Phi_1, \Phi_2, \Phi_3, \ldots$ erzeugt, dass Φ_b genau dann erfüllbar ist, wenn ein Plan der Länge b für die gegebene Planungsinstanz existiert. Dabei wird für aufsteigendes b jeweils zunächst Φ_b berechnet und anschließend auf Erfüllbarkeit geprüft. Wird eine erfüllende Belegung für Φ_b gefunden, so kann aus dieser Belegung ein Plan extrahiert werden.

Algorithmus 1 Planen durch aussagenlogische Erfüllbarkeitstests

```
 1: procedure BasicSatPlanner(P)
 2:     K ← UpperBoundForShortestPlanLength(P)
 3:     for b = 1, ..., K do
 4:         Φ_b ← Translate(P, b)
 5:         (sat, v) ← Solve(Φ_b)
 6:         if sat then
 7:             Π ← ExtractPlan(v)
 8:             return Π
 9:         end if
10:     end for
11:     return „es ex. kein Plan für P“
12: end procedure
```

Alternativ kann man auch Formeln $\Phi_1, \Phi_2, \Phi_4, \Phi_8, \ldots, \Phi_{2^i}$ erzeugen, bis zum ersten Mal eine erfüllbare Formel Φ_{2^i} auftritt und dann durch Erzeugung entsprechender Formeln eine binäre Suche zwischen den Planlängen $2^{i-1}+1$ und 2^i betreiben, um einen möglichst kurzen Plan zu finden.

Besonders effizient kann diese Form des Planens gestaltet werden, wenn mehrere Prozessoren zur Verfügung stehen und zeitgleich mehrere aussagenlogische Formeln auf Erfüllbarkeit überprüft werden können, etwa die Formel Φ_b auf Prozessor b.

Unabhängig davon, ob man einen sequentiellen oder einen parallelen Plan sucht, und unabhängig von der genauen Definition von Parallelität kann man die folgende Basiskodierung

einer Planungsinstanz in Aussagenlogik definieren [Kautz und Selman, 1992; Rintanen u. a., 2005].

2.3.3.1. Basiskodierung

Definition 13 (Basiskodierung einer Planungsinstanz). Sei $\mathcal{P} = \langle A, I, O, g\rangle$ eine Planungsinstanz. Für jede Zustandsvariable $a \in A$ sei a_t eine aussagenlogische Variable, die den Wert von a zum Zeitpunkt $t \in \{0, \ldots, b\}$ repräsentiert. Entsprechend beschreibt für alle $o \in O$ und alle Zeitpunkte $t \in \{0, \ldots, b-1\}$ die Variable o_t, ob o zum Zeitpunkt t angewandt wird. Im Folgenden werden wir häufig stillschweigend einen Operator und die zugehörige Variable mit dem gleichen Symbol bezeichnen, wenn keine Verwechslung möglich ist. Ist Φ eine Formel, die nur Zustandsvariable enthält, so sei Φ_t die entsprechende Formel, in der alle Variablen den Index t tragen. Entsprechendes gelte auch für Mengen von Variablen, d. h. ist $X = \{x^1, \ldots, x^n\}$ eine Menge von Variablen, so sei $X_t = \{x_t^1, \ldots, x_t^n\}$.

Gegeben eine Instanz $\mathcal{P} = \langle A, I, O, g\rangle$, wird eine Formel $[\![\mathcal{P}]\!]^b_{basic}$ angegeben, die genau dann erfüllbar ist, wenn es einen Plan der Länge b mit nicht notwendigerweise linearisierbaren simultanen Anwendungen von Operatoren für $\mathcal{P}$ gibt.

Es ist

$$[\![\mathcal{P}]\!]^b_{basic} = I_0 \wedge g_b \wedge \bigwedge_{t=0}^{b-1} \mathcal{R}(A_t, A_{t+1}, O_t) \qquad \text{mit}$$

$$\mathcal{R}(A_t, A_{t+1}, O_t) = \mathit{precond}_t \wedge \mathit{effects}_t \wedge \mathit{conditionalEffects}_t \wedge \mathit{frameNeg}_t \wedge \mathit{framePos}_t.$$

Dabei sagt $\mathit{precond}_t$ aus, dass die Vorbedingung jeder Operatoranwendung erfüllt sein muss, d. h.

$$\mathit{precond}_t = \bigwedge_{o=\langle p,e,c\rangle \in O} (o_t \rightarrow p_t).$$

Die Formel $\mathit{effects}_t$ drückt aus, dass aus der Anwendung eines Operators folgt, dass zum nächsten Zeitpunkt die unbedingten Effekte des Operators wahr werden, d. h.

$$\mathit{effects}_t = \bigwedge_{o=\langle p,e,c\rangle \in O} (o_t \rightarrow \bigwedge e_{t+1}).$$

Entsprechend beschreibt $\mathit{conditionalEffects}_t$, dass aus der Anwendung eines Operators und der Erfülltheit des Antezedens eines bedingten Effektes dieses Operators folgt, dass das Sukzedens des bedingten Effekts im nächsten Schritt wahr ist, d. h.

$$\mathit{conditionalEffects}_t = \bigwedge_{o=\langle p,e,c\rangle \in O} \bigwedge_{f \triangleright d \in c} ((o_t \wedge f_t) \rightarrow \bigwedge d_{t+1}).$$

Die Konjunktionsglieder $\mathit{frameNeg}_t$ und $\mathit{framePos}_t$ besagen, dass sich der Wert einer Zustandsvariable von einem Zeitpunkt zum nächsten nur dann ändern kann, wenn ein Operator angewandt wird, der dies bewirkt:

$$\mathit{frameNeg}_t = \bigwedge_{a \in A} \left((a_t \wedge \neg a_{t+1}) \rightarrow \bigvee_{o \in O} (o_t \wedge (EPC_{\neg a}(o))_t) \right) \qquad \text{bzw.}$$

$$\mathit{framePos}_t = \bigwedge_{a \in A} \left((\neg a_t \wedge a_{t+1}) \rightarrow \bigvee_{o \in O} (o_t \wedge (EPC_a(o))_t) \right).$$

Beachte, dass $[\![\mathcal{P}]\!]^b_{basic}$ das Konjunktionsglied g_b enthält, um zu garantieren, dass der letzte Zustand ein Zielzustand ist. Dieses Konjunktionsglied kann später, wenn wir allgemeinere $\mathsf{LTL}_{-\mathbf{X}}$-Zielformeln zulassen, unter Umständen durch $\mathbf{F}g$ ersetzen werden. Da man jedoch mit $\mathsf{LTL}_{-\mathbf{X}}$ nicht direkt über den i-ten Zustand eines Pfades, insbesondere also nicht über den b-ten bzw. letzten Zustand, sprechen kann, behalten wir hier das Konjunktionsglied g_b bei.

Lemma 3. *Ist n die Anzahl der Operatoren, v die Anzahl der Zustandsvariablen, b die Anzahl der Zeitpunkte und $F = \max(\{\, \|f\| \mid f \rhd d \in c \,\} \cup \{\, |d| \mid f \rhd d \in c \,\} \cup \{\, \|p\|, |e|, |c| \,\})$, so hat die Formel $[\![\mathcal{P}]\!]^b_{basic}$ die Größe $\mathcal{O}(b \cdot n \cdot v \cdot F^2)$.* □

Beispiel 14. Betrachte wieder die Planungsinstanz aus Beispiel 9. Sei $t \in \mathbb{N}$. Dann ist

$$\begin{aligned} \mathit{precond}_t &= (o_{1t} \to \neg a_{1t}) \wedge (o_{2t} \to \neg a_{2t}) \wedge (o_{3t} \to a_{1t}) \wedge \neg a_{2t}, \\ \mathit{effects}_t &= (o_{1t} \to a_{1(t+1)}) \wedge (o_{2t} \to a_{2(t+1)}) \wedge \\ &\quad (o_{3t} \to a_{2(t+1)} \wedge \neg a_{3(t+1)}) \wedge (o_{4t} \to a_{3(t+1)}) \quad \text{und} \\ \mathit{conditionalEffects}_t &= (o_{2t} \wedge a_{1t}) \to a_{3(t+1)}. \end{aligned}$$

Das Konjunktionsglied für a_3 in $\mathit{framePos}_t$ hat die Gestalt

$$(\neg a_{3t} \wedge a_{3(t+1)}) \to ((o_{2t} \wedge a_{1t}) \vee o_{4t}).$$

Die weiteren Konjunktionsglieder von $\mathit{framePos}_t$ sowie $\mathit{frameNeg}_t$ werden analog konstruiert.

Die beiden folgenden Definitionen und das sich anschließende Lemma beschreiben den Zusammenhang zwischen Plänen für Planungsinstanzen und erfüllenden Belegungen für deren aussagenlogische Übersetzungen. Wir geben diese Definitionen an, da in ihnen implizit das Verfahren zur Planextraktion aus einer erfüllenden Belegung enthalten ist und sie die Formulierung einiger weiter unten folgender Sätze erleichtern.

Definition 15. Sei $\Pi = \langle S_0, \ldots, S_{b-1} \rangle$ eine Folge von Mengen von Operatoren und $\pi = s_0, \ldots, s_b$ eine Folge von Zuständen für eine Planungsinstanz $\mathcal{P} = \langle A, I, O, g \rangle$ und $b \in \mathbb{N}$. Wir definieren die Variablenbelegung $v^b_{\Pi,\pi} : \bigcup_{t=0}^{b} A_t \cup \bigcup_{t=0}^{b-1} O_t \to \{\, 0, 1 \,\}$ durch $v^b_{\Pi,\pi}(a_t) = s_t(a)$ für alle $t \in \{\, 0, \ldots, b \,\}$ und alle Zustandsvariablen $a \in A$ sowie $v^b_{\Pi,\pi}(o_t) = 1$ gdw. $o \in S_t$ für alle $t \in \{\, 0, \ldots, b-1 \,\}$ und alle Operatoren $o \in O$.

Definition 16. Sei $v : \bigcup_{t=0}^{b} A_t \cup \bigcup_{t=0}^{b-1} O_t \to \{\, 0, 1 \,\}$ eine Belegung der Variablen in $\bigcup_{t=0}^{b} A_t \cup \bigcup_{t=0}^{b-1} O_t$. Definiere dann $\Pi^v := \langle S^v_0, \ldots, S^v_{b-1} \rangle$ mit $S^v_t = \{\, o \in O \mid v(o_t) = 1 \,\}$ für alle $t \in \{\, 1, \ldots, b-1 \,\}$ und $\pi^v := s^v_0, \ldots, s^v_b$, wobei $s^v_t : A \to \{\, 0, 1 \,\}$ eine Belegung von A mit $s^v_t(a) = v(a_t)$ für alle $t \in \{\, 0, \ldots, b \,\}$ und alle $a \in A$ ist.

Lemma 4. *Sei $\Pi = \langle S_0, \ldots, S_{b-1} \rangle$ eine Folge von Mengen von Operatoren und $\pi = s_0, \ldots, s_b$ eine Folge von Zuständen für eine Planungsinstanz $\mathcal{P} = \langle A, I, O, g \rangle$ und $b \in \mathbb{N}$. Sei ferner $v : \bigcup_{t=0}^{b} A_t \cup \bigcup_{t=0}^{b-1} O_t \to \{\, 0, 1 \,\}$ eine Belegung der Variablen in $\bigcup_{t=0}^{b} A_t \cup \bigcup_{t=0}^{b-1} O_t$. Dann gilt (1.) $v^b_{\Pi^v,\pi^v} = v$, (2.) $\pi^{v^b_{\Pi,\pi}} = \pi$ und (3.) $\Pi^{v^b_{\Pi,\pi}} = \Pi$.*

Beweis. Zu (1): Es gilt $v^b_{\Pi^v,\pi^v}(a_t) = s^v_t(a) = v(a_t)$ für alle $a \in A$ und $t \in \{\, 0, \ldots, b \,\}$ sowie $v^b_{\Pi^v,\pi^v}(o_t) = 1$ gdw. $o \in S^v_t$ gdw. $v(o_t) = 1$ für alle $o \in O$ und $t \in \{\, 0, \ldots, b-1 \,\}$. Also ist $v^b_{\Pi^v,\pi^v} = v$.

Zu (2): Für alle $t \in \{0, \ldots, b-1\}$ gilt $S_t^{v^b_{\Pi,\pi}} = \{ o \in O \mid v^b_{\Pi,\pi}(o_t) = 1 \} = \{ o \in O \mid o \in S_t \}$ $= S_t$, d. h. $\Pi^{v^b_{\Pi,\pi}} = \langle S_0^{v^b_{\Pi,\pi}}, \ldots, S_{b-1}^{v^b_{\Pi,\pi}} \rangle = \langle S_0, \ldots, S_{b-1} \rangle = \Pi$.

Zu (3): Für alle $a \in A$ und $t \in \{0, \ldots, b\}$ gilt $s_t^{v^b_{\Pi,\pi}}(a) = v^b_{\Pi,\pi}(a_t) = s_t(a)$, d. h. $s_t^{v^b_{\Pi,\pi}} = s_t$ für alle $t \in \{0, \ldots, b\}$. Also folgt $\Pi^{v^b_{\Pi,\pi}} = s_0^{v^b_{\Pi,\pi}}, \ldots, s_b^{v^b_{\Pi,\pi}} = s_0, \ldots, s_b = \Pi$. □

Das folgende Lemma enthält die zentrale Aussage über die Korrektheit und Vollständigkeit der oben angegebenen aussagenlogischen Basiskodierung. Wir folgen in seiner Formulierung und seinem Beweis der Arbeit von Rintanen u. a. [2005].

Lemma 5. *Sei $\mathcal{P} = \langle A, I, O, g \rangle$ eine Planungsinstanz. Dann gibt es eine solche Folge $\Pi = \langle S_0, \ldots, S_{b-1} \rangle$ und solche Zustände $s_0, \ldots, s_b$, dass $s_0 = I$, $s_b \models g$ und $s_{t+1} = app_{S_t}(s_t)$ für alle $t \in \{0, \ldots, b-1\}$ genau dann, wenn die Formel $[\![\mathcal{P}]\!]^b_{basic}$ erfüllbar ist.*

Beweis. Sei $\Pi = \langle S_0, \ldots, S_{b-1} \rangle$ und $s_0, \ldots, s_b$ eine Folge mit $s_0 = I$, $s_b \models g$ und $s_{t+1} = app_{S_t}(s_t)$ für alle $t \in \{0, \ldots, b-1\}$. Dann gilt, dass die Belegung $v = v^b_{\Pi,\pi}$ die Formel $[\![\mathcal{P}]\!]^b_{basic}$ erfüllt, d. h. $v \models [\![\mathcal{P}]\!]^b_{basic}$.

Dass $v \models I_0$ und $v \models g_b$, ist klar. Betrachte also noch $precond_t$, $effects_t$, $conditionalEffects_t$, $frameNeg_t$ und $framePos_t$ für $t \in \{0, \ldots, b-1\}$.

$precond_t$: Sei $t \in \{0, \ldots, b-1\}$ und $o = \langle p, e, c \rangle \in O$. Falls $o \notin S_t$, so $v \not\models o_t$ und damit $v \models o_t \rightarrow p_t$. Falls $o \in S_t$, muss $app_{S_t}(s_t)$ definiert und somit die Vorbedingung p in s_t erfüllt sein. Also $v \models o_t \rightarrow p_t$.

$effects_t$: Sei $t \in \{0, \ldots, b-1\}$ und $o = \langle p, e, c \rangle \in O$. Falls $o \notin S_t$, so $v \not\models o_t$ und damit $v \models o_t \rightarrow \bigwedge e_{t+1}$. Falls $o \in S_t$, müssen die unbedingten Effekte von o in $s_{t+1} = app_{S_t}(s_t)$ erfüllt sein. Also $v \models o_t \rightarrow \bigwedge e_{t+1}$.

$conditionalEffects_t$: Sei $t \in \{0, \ldots, b-1\}$, $o = \langle p, e, c \rangle \in O$ und $f \rhd d \in c$. Falls $o \notin S_t$, so $v \not\models o_t$ und damit $v \models (o_t \wedge f_t) \rightarrow \bigwedge d_{t+1}$. Falls $o \in S_t$ und $v \models f$, müssen die Literale in d aktive Effekte von o und damit in $s_{t+1} = app_{S_t}(s_t)$ erfüllt sein, d. h. $v \models \bigwedge d_{t+1}$. Also $v \models (o_t \wedge f_t) \rightarrow \bigwedge d_{t+1}$.

$frameNeg_t$: Sei $t \in \{0, \ldots, b-1\}$ und $a \in A$. Nach der Definition von $s_{t+1} = app_{S_t}(s_t)$ kann a nur dann in s_t wahr und in s_{t+1} falsch sein, falls $\neg a \in [o]_{s_t}$ für einen Operator $o \in S_t$. Nach Lemma 1 ist $\neg a \in [o]_{s_t}$ genau dann, wenn $s_t \models EPC_{\neg a}(o)$. Wenn also die linke Seite von $(a_t \wedge \neg a_{t+1}) \rightarrow \bigvee_{o \in O}(o_t \wedge (EPC_{\neg a}(o))_t)$ wahr ist, dann ist auch eines der Disjunktionsglieder auf der rechten Seite wahr. Also auch $v \models frameNeg_t$.

$framePos_t$: Analog zu $frameNeg_t$.

Für die andere Richtung des Beweises nehmen wir an, dass v eine Belegung ist, die $[\![\mathcal{P}]\!]^b_{basic}$ erfüllt. Die gewünschte Folge von Operatormengen ist dann $\Pi^v = \langle S_0^v, \ldots, S_{b-1}^v \rangle$ zusammen mit der Zustandsfolge $s_0^v, \ldots, s_b^v$.

Damit ist $I = s_0^v$ und $s_b^v \models g$. Es bleibt zu zeigen, dass $s_{t+1}^v = app_{S_t^v}(s_t^v)$ für alle $t \in \{0, \ldots, b-1\}$. Die Vorbedingung p jedes Operators $o \in S_t^v$ ist in s_t^v wahr, da $v \models o_t$ und $v \models o_t \rightarrow p_t$.

Es gilt $s_{t+1}^v \models [o]_{s_t^v}$ für alle $o \in S_t^v$ wegen $v \models o_t$ und $v \models o_t \rightarrow \bigwedge e_{t+1}$ für die unbedingten Effekte e von o und $v \models (o_t \wedge f_t) \rightarrow \bigwedge d_{t+1}$ für die bedingten Effekte $f \rhd d \in c$. Damit ist auch $[S_t^v]_{s_t^v}$ konsistent und $app_{S_t^v}(s_t^v)$ definiert.

Es bleibt nun noch zu zeigen, dass $s_t^v(a) = s_{t+1}^v(a)$ für solche Zustandsvariable $a \in A$, die weder positiv noch negativ in $[S_t^v]_{s_t^v}$ vorkommen. Da a nicht in $[S_t^v]_{s_t^v}$ vorkommt, gilt für jeden Operator $o \in O$, dass $o \notin S_t^v$ oder $\{ a, \neg a \} \cap [o]_{s_t^v} = \varnothing$. Also gilt entweder $v \not\models o_t$ oder nach Lemma 1, dass $v \models \neg(EPC_a(o))_t \wedge \neg(EPC_{\neg a}(o))_t$. Zusammen mit den Annahmen $v \models (a_t \wedge \neg a_{t+1}) \rightarrow \bigvee_{o \in O}(o_t \wedge (EPC_{\neg a}(o))_t)$ und $v \models (\neg a_t \wedge a_{t+1}) \rightarrow \bigvee_{o \in O}(o_t \wedge (EPC_a(o))_t)$ folgt daraus, dass $v \models (a_t \rightarrow a_{t+1}) \wedge (\neg a_t \rightarrow \neg a_{t+1})$. Also bleibt jedes $a \in A$, das nicht in $[S_t^v]_{s_t^v}$ vorkommt, unverändert. Damit ist $s_{t+1}^v = app_{S_t^v}(s_t^v)$. □

Die oben beschriebene Kodierung einer Planungsinstanz garantiert noch nicht, dass der ermittelte Plan eine zulässige Linearisierung besitzt. Um dies sicherzustellen, muss die Formel um weitere Konjunktionsglieder erweitert werden. Die in den Teilabschnitten 2.3.3.2 und 2.3.3.3 angegebenen, im Wesentlichen aus der Arbeit von Rintanen u. a. [2005] entnommenen, Definitionen werden benötigt, um solche Konjunktionsglieder zu definieren.

2.3.3.2. Deaktivierungsgraph

In diesem Teilabschnitt geben wir Definitionen wieder, die beschreiben, wie ein Operator die zu ihm parallele Anwendbarkeit eines anderen Operators unmöglich machen kann. Ein Deaktivierungsgraph ist dann ein Graph über der Menge der Operatoren, in dem zwei Operatoren o und o' durch eine gerichtete Kante miteinander verbunden sind, wenn die Anwendung von o möglicherweise verhindert, dass o' zu einem Zeitpunkt t in einer zulässigen Linearisierung einer Operatormenge S_t nach o auftreten kann.

Definition 17 (Deaktivierung). Sei A eine Menge von Zustandsvariablen und seien $o = \langle p, e, c \rangle$ und $o' = \langle p', e', c' \rangle$ Operatoren über A. Dann **deaktiviert der Operator o den Operator o'**, wenn $o \neq o'$ und es ein solches Literal $\ell \in Lit(A)$ über den Zustandsvariablen gibt, dass

1. $\ell \in [o]_\Diamond$ und
2. a) $\mathrm{neg}(\ell, p')$ oder
   b) es ein $f' \rhd d' \in c'$ gibt, so dass $\ell \in Lit(var(f'))$.

Die Operatoren o und o' **interferieren**, wenn mindestens einer der beiden Operatoren den anderen deaktiviert.

Beispiel 18. Betrachte wieder die Planungsinstanz aus Beispiel 9. Dort deaktiviert der Operator o_2 die Operatoren o_3 und o_4, denn $a_2 \in [o_2]_\Diamond$ und a_2 kommt negativ in der Vorbedingung von o_3 vor, bzw. $a_3 \in [o_2]_\Diamond$ und a_3 kommt negativ in der Vorbedingung von o_4 vor. Entsprechend wird o_2 auch von o_3 deaktiviert. Eine weitere Deaktivierung liegt zwischen o_1 und o_2 vor, denn $a_1 \in [o_1]_\Diamond$ und es gibt den bedingten Effekt $a_1 \rhd \{ a_3 \}$ von o_2, in dem a_1 auf der linken Seite vorkommt. Weitere Deaktivierungen treten nicht auf.

Ein Operator o deaktiviert also einen anderen Operator o', wenn er einen Effekt besitzt, der entweder die Vorbedingung von o' falsifiziert oder potentiell die Menge der aktiven Effekte von o' ändert. Ist dies der Fall, so ist nicht mehr gewährleistet, dass es zu einem Zeitpunkt, zu dem o und o' simultan angewandt werden sollen, eine zulässige Linearisierung der Operatoren gibt, in der o vor o' vorkommt. Gibt es umgekehrt eine Linearisierung der Operatoren in S_t, in der kein Operator einen späteren deaktiviert, so ist dies hinreichend dafür, dass die sequentielle Ausführung der Operatoren in der entsprechenden Reihenfolge definiert ist

und zu demselben Zustand führt wie die simultane Ausführung im Sinne von Definition 4. Um eine solche Reihenfolge zu finden, kann es sinnvoll sein, den unten definierten Deaktivierungsgraphen zu betrachten.

Definition 19 (Deaktivierungsgraph). Sei $\mathcal{P} = \langle A, I, O, g \rangle$ eine Planungsinstanz. Ein gerichteter Graph $G = \langle O, K \rangle$ mit $K \subseteq O \times O$ ist ein **Deaktivierungsgraph** *(Disabling Graph)* für $\mathcal{P}$, falls K alle Kanten (o, o') enthält, so dass

1. es einen Zustand s gibt, der von I mit Operatoren aus O erreichbar ist und in dem $app_{\{o,o'\}}(s)$ definiert ist, und
2. o deaktiviert o'.

Die parallele Anwendbarkeit von Operatoren ist dann nicht mehr gewährleistet – wenn auch nicht ausgeschlossen –, wenn sie sich direkt oder indirekt gegenseitig deaktivieren, d.h. wenn es einen Zyklus im Deaktivierungsgraphen gibt, der beide Operatoren enthält. Einen solchen gibt es genau dann, wenn sie in der gleichen nicht-trivialen starken Zusammenhangskomponente des Deaktivierungsgraphen liegen.

Definition 20 (Starke Zusammenhangskomponente eines gerichteten Graphen). Sei $G = \langle V, K \rangle$ ein gerichteter Graph. Eine Menge $V' \subseteq V$ heißt **starke Zusammenhangskomponente von** G, falls es für alle $v, v' \in V'$ einen gerichteten Pfad π von v nach v' in $G' = \langle V', K' \rangle$ mit $K' = K \restriction_{V' \times V'}$ gibt und dies für keine echte Obermenge von V' gilt. Eine starke Zusammenhangskomponente heißt **nicht-trivial**, wenn sie aus mehr als einem Element besteht oder einelementig ist und für dieses Element v eine Kante (v, v) enthält.

Beispiel 21. Betrachte die Planungsinstanz aus Beispiel 9. Nach Beispiel 18 hat der kleinste Deaktivierungsgraph für $\mathcal{P}$ die Form wie in Abbildung 2.2. Die drei starken Zusammenhangskomponenten sind grau hinterlegt.

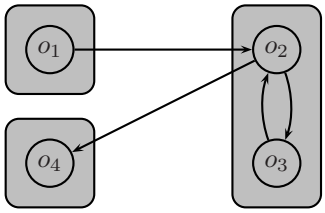


Abbildung 2.2.: Kleinster Deaktivierungsgraph der Planungsinstanz aus Beispiel 9

Definition 22. Sei $\mathcal{P} = \langle A, I, O, g \rangle$ eine Planungsinstanz und $G = \langle O, K \rangle$ ein Deaktivierungsgraph für $\mathcal{P}$. Seien $C_1, \ldots, C_m$ die starken Zusammenhangskomponenten von G. Eine totale Ordnung $\prec$ auf O heißt **mit** G **verträglich**, falls gilt:

1. für jedes $i \in \{1, \ldots, m\}$ gibt es eine totale Ordnung $\prec_i$ auf C_i und
2. es gibt eine totale Ordnung $\prec_C$ auf der Menge der starken Zusammenhangskomponenten, so dass es für alle $i, j \in \{1, \ldots, m\}$ mit $C_i \prec_C C_j$ kein Paar $o \in C_i$ und $o' \in C_j$ mit $(o, o') \in K$ gibt, so dass

für je zwei Operatoren $o, o' \in O$ genau dann $o \prec o'$ gilt, wenn es ein $i \in \{1, \ldots, m\}$ mit $\{o, o'\} \subseteq C_i$ und $o \prec_i o'$ gibt oder wenn $i, j \in \{1, \ldots, m\}$ existieren mit $i \neq j$, $o \in C_i$, $o' \in C_j$ und $C_i \prec_C C_j$.

Lemma 6. *Für jeden Deaktivierungsgraphen G jeder Planungsinstanz $\mathcal{P}$ gibt es mindestens eine mit G verträgliche totale Ordnung $\prec$ der Operatoren.*

Beweis. Sei $G = \langle O, K \rangle$ ein Deaktivierungsgraph für $\mathcal{P}$ mit den starken Zusammenhangskomponenten $C_1, \ldots, C_m$. Starke Zusammenhangskomponenten bilden immer einen gerichteten azyklischen Graphen (DAG). Denn angenommen, der Graph der Komponenten wäre zyklisch, so müssten noch mindestens zwei Komponenten zusammengefasst werden, wären also nicht maximal. Somit lassen sich die starken Zusammenhangskomponenten derart zu einer Ordnung $\prec_C$ topologisch sortieren, dass es für alle $i, j \in \{1, \ldots, m\}$ mit $C_i \prec_C C_j$ kein Paar $o \in C_i$ und $o' \in C_j$ mit $(o, o') \in K$ gibt. Wählt man für jedes $i \in \{1, \ldots, m\}$ eine beliebige Ordnung $\prec_i$ auf C_i, so kann man $\prec$ so definieren, dass genau dann $o \prec o'$ gilt, wenn es ein $i \in \{1, \ldots, m\}$ mit $\{o, o'\} \subseteq C_i$ und $o \prec_i o'$ gibt oder wenn $i, j \in \{1, \ldots, m\}$ existieren mit $i \neq j$, $o \in C_i$, $o' \in C_j$ und $C_i \prec_C C_j$. □

Das folgende Lemma, dessen Beweis in der Arbeit von Rintanen u. a. [2005] geführt wird, beschreibt den Zusammenhang zwischen der Interferenz von Operatoren und der Existenz von 1-Linearisierungen-Plänen.

Lemma 7. *Sei $\mathcal{P} = \langle A, I, O, g \rangle$ eine Planungsinstanz, $\Pi = \langle S_0, \ldots, S_{b-1} \rangle$ eine Folge von Mengen von Operatoren und $\pi = s_0, \ldots, s_t$ eine Folge von Zuständen, so dass (1.) $s_0 = I$, (2.) es für alle $t \in \{0, \ldots, b-1\}$ eine totale Ordnung $\prec$ von S_t gibt, in der kein Operator o' von einem Operator o mit $o \prec o'$ deaktiviert wird, und (3.) $s_{t+1} = app_{S_t}(s_t)$ für alle $t \in \{0, \ldots, b-1\}$. Dann ist Π ein 1-Linearisierungs-Plan für $\mathcal{P}$.* □

2.3.3.3. Parallelitätskodierung

In diesem Teilabschnitt wird eine auf dem oben definierten Deaktivierungsgraphen basierende Kodierung vorgestellt, die garantiert, dass der einer erfüllenden Belegung entsprechende Plan ein 1-Linearisierungs-Plan ist.

Definition 23. Sei O eine Menge von Operatoren und ℓ ein Literal. Definiere dann die Menge aller Operatoren, die ℓ möglicherweise falsifizieren und die Menge aller Operatoren, bei denen ℓ im Antezedens eines bedingten Effektes oder positiv in der Vorbedingung vorkommt, als

$$\begin{aligned} E_\ell &:= \left\{ o \in O \mid \overline{\ell} \in [o]_\diamond \right\} \quad \text{und} \\ R_\ell &:= \left\{ o = \langle p, e, c \rangle \in O \mid \text{pos}(\ell, p) \text{ oder } \ell \in \bigcup \{ \mathit{Lit}(\mathit{var}(f)) \mid f \rhd d \in c \} \right\}. \end{aligned}$$

E_ℓ ist also die Menge aller Operatoren, die durch ihre das Literal ℓ falsifizierenden Effekte potentiell andere Operatoren deaktivieren, während R_ℓ die Menge aller Operatoren ist, die bezüglich ℓ möglicherweise von anderen Operatoren deaktiviert werden.

Die folgende aussagenlogische Formel soll dazu dienen, eine solche Teilmenge C_i' der Operatoren einer starken Zusammenhangskomponente C_i eines Deaktivierungsgraphen G zu finden, dass es bei einer fest vorgegebenen totalen Ordnung $\prec_i$ von C_i keine zwei Operatoren $o, o' \in C_i$ mit den folgenden Eigenschaften gibt: (1.) $o, o' \in C_i'$, (2.) $o \prec_i o'$ und (3.) o deaktiviert o'. Dann ist C_i' eine Menge von Operatoren aus C_i, die zu einem gegebenen Zeitpunkt t parallel angewandt werden dürfen, und $S_t = \bigcup_{i=1}^{m} C_i'$. Eine zulässige Linearisierung

von S_t erhält man durch Restriktion einer mit G verträglichen linearen Ordnung $\prec$, die $\prec|_{C_i} = \prec_i$ für alle $i \in \{1, \ldots, m\}$ erfüllt, auf S_t.

Definition 24. Sei $o^1 \prec \cdots \prec o^n$ eine lineare Ordnung einer Menge von Operatoren, ℓ ein Literal und seien E und R zwei Mengen von Operatoren. Dann sei die aussagenlogische Formel $\mathit{linchain}(o^1, \ldots, o^n; E; R; \ell)$ wie folgt definiert:

$$\begin{aligned}
&\mathit{linchain}(o^1, \ldots, o^n; E; R; \ell) := \\
&\quad \bigwedge \{\, o^i \to a^{j,\ell} \mid 1 \le i < j \le n, o^i \in E, o^j \in R, \{\, o^{i+1}, \ldots, o^{j-1} \,\} \cap R = \varnothing \,\} \wedge \\
&\quad \bigwedge \{\, a^{i,\ell} \to a^{j,\ell} \mid 1 \le i < j \le n, \{\, o^i, o^j \,\} \subseteq R, \{\, o^{i+1}, \ldots, o^{j-1} \,\} \cap R = \varnothing \,\} \wedge \\
&\quad \bigwedge \{\, a^{i,\ell} \to \neg o^i \mid 1 \le i \le n, o^i \in R \,\}.
\end{aligned}$$

mit bisher unbenutzten Hilfsvariablen $a^{i,\ell}, 1 \le i \le n$.

Beispiel 25. Betrachte die Planungsinstanz aus Beispiel 9. Sei $\ell = \neg a_2$, $E = E_\ell = \{\, o_2, o_3 \,\}$ und $R = R_\ell = \{\, o_2, o_3 \,\}$. Dann ist

$$\begin{aligned}
\mathit{linchain}(o_1, o_2, o_3, o_4; E; R; \ell) &= (o_2 \to a^{3,\ell}) \wedge (a^{2,\ell} \to a^{3,\ell}) \wedge (a^{2,\ell} \to \neg o_2) \wedge (a^{3,\ell} \to \neg o_3) \\
&\models o_2 \to \neg o_3
\end{aligned}$$

Definition 26 (Kodierung der 1-Linearisierungs-Eigenschaft). Sei $\mathcal{P} = \langle A, I, O, g \rangle$ eine Planungsinstanz und $G = \langle O, K \rangle$ ein Deaktivierungsgraph für $\mathcal{P}$. Seien $C_1, \ldots, C_m$ die starken Zusammenhangskomponenten von G und sei für alle $i \in \{1, \ldots, m\}$ auf $C_i = \{\, o_{i_1}, \ldots, o_{i_{|C_i|}} \,\}$ eine beliebige totale Ordnung $\prec_i$, etwa $o_{i_1} \prec_i \cdots \prec_i o_{i_{|C_i|}}$, gegeben. Definiere dann

$$[\![\mathcal{P}]\!]^{b,1}_{1lin} := \bigwedge_{t=0}^{b-1} \bigwedge_{\ell \in Lit(A)} \bigwedge_{i=1}^{m} \mathit{linchain}(o_{i_1}, \ldots, o_{i_{|C_i|}}; E_\ell; R_\ell; \ell)_t.$$

Lemma 8. *Die Formel $[\![\mathcal{P}]\!]^{b,1}_{1lin}$ hat Größe $\mathcal{O}(n)$ für $n = |O|$.* □

Lemma 9. *Sei $\mathcal{P} = \langle A, I, O, g \rangle$ eine Planungsinstanz und sei die Formel $[\![\mathcal{P}]\!]^{b}_{basic} \wedge [\![\mathcal{P}]\!]^{b,1}_{1lin}$ erfüllbar, etwa durch die Belegung v. Dann gibt es einen 1-Linearisierungs-Plan der Länge b für $\mathcal{P}$, genauer: $\Pi^v = \langle S^v_0, \ldots, S^v_{b-1} \rangle$ zusammen mit $\pi^v = s^v_0, \ldots, s^v_b$ ist ein 1-Linearisierungs-Plan für $\mathcal{P}$.*

Beweis. Sei G der bei der Kodierung benutzte Deaktivierungsgraph für $\mathcal{P}$, v eine Belegung mit $v \models [\![\mathcal{P}]\!]^{b}_{basic} \wedge [\![\mathcal{P}]\!]^{b,1}_{1lin}$ und $\prec$ eine mit G und den bei der Kodierung benutzten Ordnungen $\prec_i$ verträgliche totale Ordnung von O. Nach Lemma 7 reicht es, neben $s_{t+1} = app_{S_t}(s_t)$ zu zeigen, dass es für alle $t \in \{0, \ldots, b-1\}$ keine zwei Operatoren $o, o' \in S_t$ gibt, so dass o' von o deaktiviert wird und dass $o \prec o'$. Angenommen, es gäbe ein solches t und ein solches Paar von Operatoren. Der Fall, dass $o \in C_i$ und $o' \in C_j$ für $i \neq j$, ist ausgeschlossen, denn nach der Definition einer mit dem Deaktivierungsgraphen verträglichen totalen Ordnung gilt wegen $(o, o') \in K$ auch $C_j \prec_C C_i$ und damit nach der Voraussetzung über $\prec$ auch $o' \prec o$, im Widerspruch zur Annahme. Seien also $o, o' \in C_i$. Da o' von o deaktiviert wird, gibt es ein Literal ℓ mit $\overline{\ell} \in [o]_\Diamond$ und $\mathrm{pos}(\ell, p')$ oder $\ell \in Lit(var(f'))$ für ein $f' \rhd d' \in c'$, wenn $o' = \langle p', e', c' \rangle$. Damit ist nach Definition $o \in E_\ell$ und $o' \in R_\ell$. Nach Voraussetzung gilt $v \models \mathit{linchain}(o_{i_1}, \ldots, o_{i_{|C_i|}}; E_\ell; R_\ell; \ell)_t$ und wegen $o \in S_t$ auch $v \models o_t$. Die Formel

$linchain(o_{i_1}, \ldots, o_{i_{|C_i|}}; E_\ell; R_\ell; \ell)_t$ enthält wegen $o \prec_i o'$ eine Folge von Implikationen $o_t \rightarrow a_t^{j_1,\ell} \rightarrow a_t^{j_2,\ell} \rightarrow \ldots \rightarrow a_t^{j_k,\ell} \rightarrow \neg o'_t$. Daraus folgt, dass $v \models \neg o'_t$, also $o' \notin S_t$ im Widerspruch zur Annahme. Dass $s_{t+1} = app_{S_t}(s_t)$ gilt, ist klar. □

2.4. Modellprüfung

In diesem Abschnitt werden die Grundbegriffe der Modellprüfung, insbesondere der LTL-Modellprüfung, dargestellt. Dazu wird die Logik LTL eingeführt (Teilabschnitt 2.4.1), das Modellprüfungs-Problem für diese Logik definiert (Teilabschnitt 2.4.2), ein symbolisches LTL-Modellprüfungs-Verfahren durch Übersetzung in das aussagenlogische Erfüllbarkeitsproblem dargestellt (Teilabschnitt 2.4.3), das Gegenbeispiele bis zu einer vorgegebenen Maximallänge finden kann, und gezeigt, wie die Effizienz von Modellprüfungs-Verfahren erhöht werden kann, ohne die Korrektheit bzw. Vollständigkeit der Verfahren zu beeinträchtigen, indem parallele Transitionen zugelassen werden (Teilabschnitt 2.4.4).

Auf andere Modellprüfungs-Verfahren wie explizite Modellprüfung, Modellprüfung mit Büchi-Automaten oder symbolische Modellprüfung mit BDDs sowie auf Modellprüfung für andere als in LTL formulierbare Eigenschaften [Clarke u. a., 2002] wird nicht eingegangen.

2.4.1. Linearzeit-Temporallogik

Temporallogiken sind spezielle Modallogiken, mit denen Eigenschaften von Transitionssystemen beschrieben werden können. Während man mit den Logiken CTL* und CTL auch das Verzweigungsverhalten von Transitionssystemen beschreiben kann, ist die Logik LTL nur ausdrucksstark genug, um Aussagen über einzelne Pfade in Transitionssystemen zu machen. Wir wollen hier dennoch lediglich die Logiken LTL und $\mathsf{LTL}_{-\mathbf{X}}$ betrachten, da die Ausführungspfade von Plänen, deren Eigenschaften wir mit Hilfe temporallogischer Formeln spezifizieren wollen, bereits linear sind.

2.4.1.1. Syntax

Ohne die Ausdrucksstärke von LTL einzuschränken, betrachten wir nur LTL-Formeln in Negationsnormalform, d. h. solche Formeln, in denen alle Negationssymbole unmittelbar vor Aussagenvariablen auftreten. Unter Verwendung des **R**-Operators, der durch $\varphi_1 \mathbf{R} \varphi_2 \leftrightarrow \neg(\neg\varphi_1 \mathbf{U} \neg\varphi_2)$ definiert ist, kann jede LTL-Formel in eine äquivalente Formel in Negationsnormalform umgewandelt werden, ohne sie wesentlich zu vergrößern [Clarke u. a., 2002].

Definition 27 (LTL-Formel). Sei A eine Menge von Aussagenvariablen. Die Menge der LTL-Formeln in Negationsnormalform (kurz LTL-Formeln) über A ist dann wie folgt definiert:

1. Für jedes $a \in A$ sind a und $\neg a$ LTL-Formeln in Negationsnormalform.
2. Sind φ_1 und φ_2 LTL-Formeln in Negationsnormalform, so auch $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$, $\mathbf{F}\varphi_1$, $\mathbf{G}\varphi_1$, $\mathbf{X}\varphi_1$, $\varphi_1 \mathbf{U} \varphi_2$ und $\varphi_1 \mathbf{R} \varphi_2$, d. h.

$$\begin{aligned} \mathsf{LTL} \quad ::= \quad & A \mid \neg A \mid \mathsf{LTL} \wedge \mathsf{LTL} \mid \mathsf{LTL} \vee \mathsf{LTL} \\ & \mid \mathbf{F}\,\mathsf{LTL} \mid \mathbf{G}\,\mathsf{LTL} \mid \mathbf{X}\,\mathsf{LTL} \mid \mathsf{LTL}\,\mathbf{U}\,\mathsf{LTL} \mid \mathsf{LTL}\,\mathbf{R}\,\mathsf{LTL}. \end{aligned}$$

Wie in der Aussagenlogik können als Abkürzungen die Symbole $\rightarrow$ und $\leftrightarrow$ eingeführt werden. Mit $var(\varphi)$ bezeichnen wir wie in der Aussagenlogik die Menge der in φ auftretenden Variablen, mit $\|\varphi\|$ ihre Länge.

Definition 28 ($\mathsf{LTL}_{-\mathbf{X}}$-Formel). Sei A eine Menge von Aussagenvariablen. Die Menge der $\mathsf{LTL}_{-\mathbf{X}}$-Formeln in Negationsnormalform (kurz $\mathsf{LTL}_{-\mathbf{X}}$-Formeln) über A ist die Menge aller LTL-Formeln, in denen an keiner Stelle der Operator $\mathbf{X}$ vorkommt.

Obwohl die in dieser Arbeit in Kapitel 3 vorgestellte aussagenlogische Kodierung nur eine Kodierung des Handlungsplanungsproblems für $\mathsf{LTL}_{-\mathbf{X}}$-, nicht für allgemeinere LTL-Zielformeln darstellt, wollen wir uns im Rest dieses Abschnitts noch nicht auf $\mathsf{LTL}_{-\mathbf{X}}$ beschränken. Erzwingt man durch Hinzufügen eines geeigneten Konjunktionsgliedes zu der aussagenlogischen Kodierung *sequentielle* Pläne, so sind auch allgemeine LTL-Formeln mit $\mathbf{X}$-Operator als Zielspezifikationen möglich.

2.4.1.2. Semantik

In diesem Teilabschnitt definieren wir mehrere Gültigkeitsbegriffe einer LTL-Formel φ auf einem Pfad π, kurz $\pi \models \varphi$, nämlich Gültigkeit auf unendlichen Pfaden, Gültigkeit auf endlich langen Pfaden, die auf einen Zustand enden, der bereits zuvor auf dem Pfad aufgetreten ist, und somit einen Zyklus repräsentieren, sowie Gültigkeit auf endlichen Pfaden, die keinen Zyklus repräsentieren. Die Definitionen sind im Wesentlichen aus der Arbeit von Biere u. a. [1999] entnommen.

Definition 29 (Kripke-Semantik für LTL-Formeln auf unendlichen Pfaden). Sei $\mathfrak{M} = \langle \mathfrak{F}, L \rangle$ ein Kripke-Modell, π ein unendlicher Pfad in $\mathfrak{M}$ und φ eine LTL-Formel. Dann ist φ **gültig auf** π, wenn $\pi \models \varphi$, wobei

$$\pi \models a \text{ :gdw. } a \in L(\pi(0)) \qquad \pi \models \neg a \text{ :gdw. } a \notin L(\pi(0))$$

$$\pi \models \varphi_1 \wedge \varphi_2 \text{ :gdw. } \pi \models \varphi_1 \text{ und } \pi \models \varphi_2 \qquad \pi \models \varphi_1 \vee \varphi_2 \text{ :gdw. } \pi \models \varphi_1 \text{ oder } \pi \models \varphi_2$$

$$\pi \models \mathbf{F}\varphi \text{ :gdw. ex. } i \in \mathbb{N}, \text{ so dass } \pi^i \models \varphi \qquad \pi \models \mathbf{G}\varphi \text{ :gdw. f. a. } i \in \mathbb{N} \text{ gilt } \pi^i \models \varphi$$

$$\pi \models \mathbf{X}\varphi \text{ :gdw. } \pi^1 \models \varphi$$

$$\pi \models \varphi_1 \mathbf{U} \varphi_2 \text{ :gdw. ex. } i \in \mathbb{N}, \text{ so dass } \pi^i \models \varphi_2 \text{ und für alle } j \in \mathbb{N} \text{ mit } 0 \leq j < i \text{ gilt } \pi^j \models \varphi_1$$

$$\pi \models \varphi_1 \mathbf{R} \varphi_2 \text{ :gdw. f. a. } i \in \mathbb{N} \text{ gilt, dass } \pi^i \models \varphi_2 \text{ oder ex. } j \in \mathbb{N} \text{ mit } 0 \leq j < i \text{ und } \pi^j \models \varphi_1$$

Definition 30. Seien $k, b \in \mathbb{N}$ und $k \leq b$. Ein Pfad π in $\mathfrak{M} = \langle Q, R, L \rangle$ heißt (b,k)**-Schleife**, falls er die Form $\pi = \pi(0), \ldots, \pi(k-1), \pi(k), \ldots, \pi(b-1), \pi(b)$ mit $\pi(b) = \pi(k-1)$ hat. Sind $u = \pi(0), \ldots, \pi(k-1)$ und $v = \pi(k), \ldots, \pi(b)$, so definieren wir $\pi_k^\infty := u \circ v^\omega$. Anschaulich ist π_k^∞ der *unendliche* Pfad, der entsteht, wenn man die Schleife v nicht nur einmal, wie in π, sondern unendlich oft durchläuft. Der Pfad π heißt b**-Schleife**, falls es ein solches $k \in \mathbb{N}$ gibt, dass π eine (b,k)-Schleife ist.

(a) Pfad ohne Schleife (b) $(4, 2)$-Schleife

Abbildung 2.3.: Beispiele für Pfade mit und ohne Schleifen

Definition 31 (Kripke-Semantik für LTL auf endlichen Pfaden ohne Schleife). Sei $b \in \mathbb{N}$, $\mathfrak{M}$ ein Kripke-Modell, π ein Pfad der Länge mindestens $b+1$ in $\mathfrak{M}$, der *keine* b-Schleife ist, und φ eine LTL-Formel. Dann ist φ **gültig auf** π **mit Schranke** b, kurz $\pi \models_b \varphi$, wenn $\pi \models_b^0 \varphi$, wobei

$$\pi \models_b^t a \text{ :gdw. } a \in L(\pi(t)) \qquad \pi \models_b^t \neg a \text{ :gdw. } a \notin L(\pi(t))$$

$$\pi \models_b^t \varphi_1 \wedge \varphi_2 \text{ :gdw. } \pi \models_b^t \varphi_1 \text{ und } \pi \models_b^t \varphi_2 \qquad \pi \models_b^t \varphi_1 \vee \varphi_2 \text{ :gdw. } \pi \models_b^t \varphi_1 \text{ oder } \pi \models_b^t \varphi_2$$

$$\pi \models_b^t \mathbf{F}\varphi \text{ :gdw. ex. } j \in \mathbb{N}, t \leq j \leq b : \pi \models_b^j \varphi \qquad \pi \not\models_b^t \mathbf{G}\varphi$$

$$\pi \models_b^t \mathbf{X}\varphi \text{ :gdw. } t < b \text{ und } \pi \models_b^{t+1} \varphi$$

$$\pi \models_b^t \varphi_1 \mathbf{U} \varphi_2 \text{ :gdw. ex. } j \in \mathbb{N}, t \leq j \leq b : \pi \models_b^j \varphi_2 \text{ und f. a. } n \in \mathbb{N} \text{ mit } t \leq n < j : \pi \models_b^n \varphi_1$$

$$\pi \models_b^t \varphi_1 \mathbf{R} \varphi_2 \text{ :gdw. ex. } j \in \mathbb{N}, t \leq j \leq b : \pi \models_b^j \varphi_1 \text{ und f. a. } n \in \mathbb{N} \text{ mit } t \leq n \leq j : \pi \models_b^n \varphi_2$$

Ist $t > b$, so gilt $\pi \not\models_b^t \varphi$ für jede LTL-Formel φ.

Definition 32 (Kripke-Semantik für LTL auf endlichen Pfaden mit Schleife). Seien $k, b \in \mathbb{N}$, $\mathfrak{M}$ ein Kripke-Modell, π eine (b, k)-Schleife in $\mathfrak{M}$ und φ eine LTL-Formel. Dann ist φ **gültig auf** π **mit Schleifenanfang** k **und Schranke** b, kurz $\pi \models_{(b,k)} \varphi$, wenn $\pi_k^\infty \models \varphi$. Da ein Pfad zugleich eine (b, k)- und eine (b, k')-Schleife für $k \neq k'$ sein kann, sagen wir, dass φ **auf** π **mit Schranke** b **gültig ist**, kurz $\pi \models_b \varphi$, wenn es ein solches k gibt, dass π eine (b, k)-Schleife ist und dass $\pi \models_{(b,k)} \varphi$.

Beispiel 33. Seien $\pi_1 = 000, 101, 100, 001$, $\pi_2 = 011, 111$, $\pi_3 = \pi_1 \circ \pi_2 \circ 011 = 000, 101, 100, 001, 011, 111, 011$ und $\pi_4 = \pi_1 \circ \pi_2^\omega = 000, 101, 100, 001, 011, 111, 011, 111, \ldots$ Pfade in einem Kripke-Rahmen über den Zustandsvariablen a_1, a_2 und a_3, wobei die Bezeichnungen der Zustände der Konvention aus Beispiel 9 entsprechen. Dann sind π_1, π_2 und π_3 endliche Pfade, darunter π_3 die einzige Schleife, eine $(6, 5)$-Schleife, und π_4 der einzige unendliche Pfad. Seien ferner die folgenden Spezifikationen gegeben:

$$\varphi_1 = (a_1 \leftrightarrow a_3)\mathbf{U}(a_1 \wedge \neg a_3) \quad \text{und} \quad \varphi_2 = (\neg(a_2 \wedge a_3))\mathbf{U}(\mathbf{G}(a_2 \wedge a_3)).$$

Dann gilt etwa $\pi_1 \models_3 \varphi_1$ (wegen $\pi_1 \models_3^0 \varphi_1$) und $\pi_3 \models_6 \varphi_2$ (denn π_3 ist eine $(6, 5)$-Schleife und $\pi_3 \models_{(6,5)} \varphi_2$ wegen $\pi_{3,5}^\infty = \pi_4 \models \varphi_2$).

2.4.2. Das Modellprüfungs-Problem

Das LTL-Modellprüfungs-Problem ist das Problem, für ein Kripke-Modell $\mathfrak{M} = \langle Q, R, L \rangle$, einen Zustand $s \in Q$ und eine LTL-Formel φ zu entscheiden, ob $\mathfrak{M}, s \models \mathbf{E}\varphi$, d. h. ob es einen Pfad π in $\mathfrak{M}$ mit $\pi(0) = s$ und $\pi \models \varphi$ gibt.

2.4.3. Modellprüfung durch Übersetzung in Aussagenlogik

In ihrer Arbeit aus dem Jahr 1999 stellen Biere, Cimatti, Clarke und Zhu einen Algorithmus zur symbolischen beschränkten Modellprüfung *(Symbolic Bounded Model Checking)* vor, der eine Modellprüfungs-Instanz $\mathfrak{M}, s \models_b \mathbf{E}\varphi$, d. h. die Frage, ob es einen Pfad π in $\mathfrak{M}$ mit $\pi(0) = s$ und $\pi \models_b \varphi$ gibt, so in eine aussagenlogische Formel Φ_b übersetzt, dass $\mathfrak{M}, s \models_b \mathbf{E}\varphi$ genau dann gilt, wenn Φ_b erfüllbar ist.

2.4.3.1. Kodierung

Bei der Übersetzung einer LTL-Formel in eine aussagenlogische Formel folgen wir der Arbeit von Latvala u. a. [2004], in der eine etwas einfachere Übersetzung als von Biere u. a. [1999] angegeben wird, die immer eine in der Planlänge und der Größe der LTL-Formel linear große[1] aussagenlogische Formel erzeugt. Dabei benötigen wir Hilfsvariable $loopto_t$ für alle $t \in \{0, \ldots, b-1\}$ und $smallerEx_t$ für alle $t \in \{0, \ldots, b-1\}$, wobei $loopto_t$ nur dann wahr sein soll, wenn es eine Schleife zurück zu $\pi(t)$ gibt, und $smallerEx_t$ genau dann, wenn es eine Schleife zu einem Zustand $\pi(j)$ mit $j < t$ gibt. Neben der eigentlichen Übersetzung der LTL-Formel werden Axiome kodiert, die beschreiben, ob ein erfüllender Pfad eine Schleife ist und zu welchem Zustand in diesem Fall zurückgesprungen wird. Diese Axiome sind

$$
\begin{aligned}
looptoAxioms_b &:= \bigwedge_{t=0}^{b-1} \Big(loopto_t \rightarrow \bigwedge_{a \in A} (a_t \leftrightarrow a_b) \Big) \\
smallerExistsAxioms_b &:= \neg smallerEx_0 \wedge \\
&\qquad \bigwedge_{t=1}^{b-1} \Big(smallerEx_t \leftrightarrow (smallerEx_{t-1} \vee loopto_{t-1}) \Big) \\
atMostOneAxioms_b &:= \bigwedge_{t=0}^{b-1} \Big(smallerEx_t \rightarrow \neg loopto_t \Big) \\
loopConstraints_b &:= looptoAxioms_b \wedge smallerExistsAxioms_b \wedge atMostOneAxioms_b
\end{aligned}
$$

Die Beobachtung, die zu der von Latvala u. a. [2004] und ähnlich von Biere, Cimatti, Clarke, Strichman und Zhu [2003] beschriebenen Kodierung geführt hat, ist, dass es möglich ist, den Ausführungspfad eines Planes nicht nur als Pfad im gesamten dem Zustandsraum entsprechenden Kripke-Modell, sondern in einem neuen Kripke-Modell zu betrachten, das nur aus dem Pfad selbst besteht. Da in diesem neuen Kripke-Modell jeder Knoten höchstens einen Nachfolger besitzt, haben die Pfadquantoren **A** und **E** die gleiche Bedeutung und die zu verifizierende LTL-Formel kann durch Ersetzung jedes Temporaloperators O durch $\mathbf{E}O$ in eine CTL-Formel umgewandelt und dann mit CTL-Modellprüfungs-Methoden behandelt werden. Die in den oben genannten Arbeiten beschriebene Übersetzung beruht auf der Fixpunkt-Charakterisierung der CTL-Modellprüfung [Clarke u. a., 2002], auf die hier nicht näher eingegangen werden soll.

Für Formeln φ der Gestalt $\varphi_1 \mathbf{U} \varphi_2$, $\varphi_1 \mathbf{R} \varphi_2$, $\mathbf{F}\varphi_1$ und $\mathbf{G}\varphi_1$ ist es, wenn $\pi(t)$ innerhalb der Schleife liegt, zur Entscheidung, ob $\pi \models_b^t \varphi$, im Allgemeinen nötig, für alle j, für die $\pi(j)$

[1] Für den Beweis der Linearität verweisen wir auf Latvala u. a. [2004].

ebenfalls innerhalb der Schleife liegt, zu wissen, ob $\pi \models_b^j \varphi_1$ bzw. $\pi \models_b^j \varphi_2$, insbesondere auch für solche j, für die $\pi(j)$ zwischen dem Anfang der Schleife und dem aktuellen Zustand $\pi(t)$ liegt. Um das zu bestimmen, wird eine approximative Übersetzung für die Fixpunktberechnung verwendet, die in das Endergebnis eingeht.

Definition 34. Seien $b, t \in \mathbb{N}$ mit $t \leq b$. Dann sind die Übersetzung $[\![\cdot]\!]_b^t$ und die Hilfsübersetzung $\langle\!\langle \cdot \rangle\!\rangle_b^t$ einer LTL-Formel wie folgt definiert:

	$t < b$	$t = b$
$[\![a]\!]_b^t$	a_t	$\bigvee_{j=0}^{b-1}(loopto_j \wedge a_j)$
$[\![\neg a]\!]_b^t$	$\neg a_t$	$\bigvee_{j=0}^{b-1}(loopto_j \wedge \neg a_j)$
$[\![\varphi_1 \wedge \varphi_2]\!]_b^t$	$[\![\varphi_1]\!]_b^t \wedge [\![\varphi_2]\!]_b^t$	$\bigvee_{j=0}^{b-1}(loopto_j \wedge [\![\varphi_1 \wedge \varphi_2]\!]_b^j)$
$[\![\varphi_1 \vee \varphi_2]\!]_b^t$	$[\![\varphi_1]\!]_b^t \vee [\![\varphi_2]\!]_b^t$	$\bigvee_{j=0}^{b-1}(loopto_j \wedge [\![\varphi_1 \vee \varphi_2]\!]_b^j)$
$[\![\mathbf{X}\varphi]\!]_b^t$	$[\![\varphi]\!]_b^{t+1}$	$\bigvee_{j=0}^{b-1}(loopto_j \wedge [\![\varphi]\!]_b^{j+1})$
$[\![\mathbf{F}\varphi]\!]_b^t$	$[\![\varphi]\!]_b^t \vee [\![\mathbf{F}\varphi]\!]_b^{t+1}$	$\bigvee_{j=0}^{b-1}(loopto_j \wedge \langle\!\langle \mathbf{F}\varphi \rangle\!\rangle_b^j)$
$[\![\mathbf{G}\varphi]\!]_b^t$	$[\![\varphi]\!]_b^t \wedge [\![\mathbf{G}\varphi]\!]_b^{t+1}$	$\bigvee_{j=0}^{b-1}(loopto_j \wedge \langle\!\langle \mathbf{G}\varphi \rangle\!\rangle_b^j)$
$[\![\varphi_1 \mathbf{U} \varphi_2]\!]_b^t$	$[\![\varphi_2]\!]_b^t \vee ([\![\varphi_1]\!]_b^t \wedge [\![\varphi_1 \mathbf{U} \varphi_2]\!]_b^{t+1})$	$\bigvee_{j=0}^{b-1}(loopto_j \wedge \langle\!\langle \varphi_1 \mathbf{U} \varphi_2 \rangle\!\rangle_b^j)$
$[\![\varphi_1 \mathbf{R} \varphi_2]\!]_b^t$	$[\![\varphi_2]\!]_b^t \wedge ([\![\varphi_1]\!]_b^t \vee [\![\varphi_1 \mathbf{R} \varphi_2]\!]_b^{t+1})$	$\bigvee_{j=0}^{b-1}(loopto_j \wedge \langle\!\langle \varphi_1 \mathbf{R} \varphi_2 \rangle\!\rangle_b^j)$
$\langle\!\langle \mathbf{F}\varphi \rangle\!\rangle_b^t$	$[\![\varphi]\!]_b^t \vee \langle\!\langle \mathbf{F}\varphi \rangle\!\rangle_b^{t+1}$	$\bot$
$\langle\!\langle \mathbf{G}\varphi \rangle\!\rangle_b^t$	$[\![\varphi]\!]_b^t \wedge \langle\!\langle \mathbf{G}\varphi \rangle\!\rangle_b^{t+1}$	$\top$
$\langle\!\langle \varphi_1 \mathbf{U} \varphi_2 \rangle\!\rangle_b^t$	$[\![\varphi_2]\!]_b^t \vee ([\![\varphi_1]\!]_b^t \wedge \langle\!\langle \varphi_1 \mathbf{U} \varphi_2 \rangle\!\rangle_b^{t+1})$	$\bot$
$\langle\!\langle \varphi_1 \mathbf{R} \varphi_2 \rangle\!\rangle_b^t$	$[\![\varphi_2]\!]_b^t \wedge ([\![\varphi_1]\!]_b^t \vee \langle\!\langle \varphi_1 \mathbf{R} \varphi_2 \rangle\!\rangle_b^{t+1})$	$\top$

Definition 35. Sei $\mathcal{P} = \langle A, I, O, g \rangle$ eine Planungsinstanz, $b \in \mathbb{N}$, φ eine LTL-Formel und $[\![\mathcal{P}]\!]^b$ eine aussagenlogische Kodierung eines Planes der Länge b für $\mathcal{P}$, etwa $[\![\mathcal{P}]\!]_{basic}^b$. Dann sind die aussagenlogischen Formeln $[\![\mathcal{P}, \varphi]\!]_{BMC}^b$ und $[\![\mathcal{P}, \varphi]\!]^b$ definiert durch

$$\begin{aligned} [\![\mathcal{P}, \varphi]\!]_{BMC}^b &:= loopConstraints_b \wedge [\![\varphi]\!]_b^0 \qquad \text{und} \\ [\![\mathcal{P}, \varphi]\!]^b &:= [\![\mathcal{P}]\!]^b \wedge [\![\mathcal{P}, \varphi]\!]_{BMC}^b. \end{aligned}$$

Beispiel 36. Seien $A = \{ a^1, a^2 \}$, $b = 1$ und $\varphi = \mathbf{F}a^1 \wedge \mathbf{G}a^2$. Dann haben wir

$$\begin{aligned} looptoAxioms_1 &= loopto_0 \rightarrow ((a_0^1 \leftrightarrow a_1^1) \wedge (a_0^2 \leftrightarrow a_1^2)) \\ smallerExistsAxioms_1 &= \neg smallerEx_0 \\ atMostOneAxioms_1 &= smallerEx_0 \rightarrow \neg loopto_0 \\ [\![\mathbf{F}a^1 \wedge \mathbf{G}a^2]\!]_1^0 &= [\![\mathbf{F}a^1]\!]_1^0 \wedge [\![\mathbf{G}a^2]\!]_1^0 \\ &= ([\![a^1]\!]_1^0 \vee [\![\mathbf{F}a^1]\!]_1^1) \wedge ([\![a^2]\!]_1^0 \wedge [\![\mathbf{G}a^2]\!]_1^1) \\ &= ([\![a^1]\!]_1^0 \vee (loopto_0 \wedge \langle\!\langle \mathbf{F}a^1 \rangle\!\rangle_1^0)) \wedge ([\![a^2]\!]_1^0 \wedge (loopto_0 \wedge \langle\!\langle \mathbf{G}a^2 \rangle\!\rangle_1^0)) \\ &= ([\![a^1]\!]_1^0 \vee (loopto_0 \wedge ([\![a^1]\!]_1^0 \vee \langle\!\langle \mathbf{F}a^1 \rangle\!\rangle_1^1)))\wedge \\ &\quad ([\![a^2]\!]_1^0 \wedge (loopto_0 \wedge ([\![a^2]\!]_1^0 \wedge \langle\!\langle \mathbf{G}a^2 \rangle\!\rangle_1^1))) \\ &= (a_0^1 \vee (loopto_0 \wedge (a_0^1 \vee \bot))) \wedge (a_0^2 \wedge (loopto_0 \wedge (a_0^2 \wedge \top))) \\ &\equiv a_0^1 \wedge a_0^2 \wedge loopto_0. \end{aligned}$$

Insgesamt erhält man daraus

$$[\![\mathcal{P}, \varphi]\!]^b_{BMC} \equiv \neg \mathit{smallerEx}_0 \wedge \mathit{loopto}_0 \wedge a^1_0 \wedge a^2_0 \wedge a^1_1 \wedge a^2_1,$$

d. h. eine erfüllende Belegung muss beide Zustandsvariable zu beiden Zeitpunkten erfüllen und damit einem Plan entsprechen, der im Zustandsgraphen eine $(1, 1)$-Schleife erzeugt.

2.4.3.2. Korrektheit

In den folgenden Lemmata dieses Teilabschnitts sei immer $\mathcal{P} = \langle A, I, O, g \rangle$ eine Planungsinstanz, $b, k, t \in \mathbb{N}$, $0 \leq k, t \leq b$, $\Pi = \langle S_0, \ldots, S_{b-1} \rangle$ eine Folge von Mengen von Operatoren, $\pi = s_0, \ldots, s_b$ eine Folge von Zuständen sowie φ eine LTL-Formel. Ferner sei v eine Belegung, deren Restriktion auf $\bigcup_{t=0}^{b} A_t \cup \bigcup_{t=0}^{b-1} O_t$ mit $v^b_{\Pi,\pi}$ übereinstimmt, die zusätzlich für die Variablen in $\bigcup_{t=0}^{b-1} \{ \mathit{loopto}_t, \mathit{smallerEx}_t \}$ definiert ist und für die $v \models \mathit{loopConstraints}_b$ gilt.

Lemma 10. *Falls $v \models \mathit{loopto}_{k-1}$, so ist π eine (b, k)-Schleife.*

Beweis. Gelte $v \models \mathit{loopto}_{k-1}$. Wegen $v \models \mathit{loopConstraints}_b$ folgt, dass $v \models \mathit{loopto}_{k-1} \rightarrow \bigwedge_{a \in A}(a_{k-1} \leftrightarrow a_b)$, d. h. mit Modus Ponens und Definition der Konjunktion, dass $v \models a_{k-1} \leftrightarrow a_b$ für alle $a \in A$. Nach dem Lemma über den Zusammenhang zwischen Plänen und erfüllenden Belegungen (Lemma 4) folgt $\pi(k-1) = \pi(b)$. Da π die Länge $b+1$ besitzt, ist π eine (b, k)-Schleife. □

Lemma 11. *Sei $\pi = u \circ v^\omega$ eine (b, k)-Schleife mit $\|u\| = k$ und $\|v\| = b - k + 1$ und sei φ eine Formel der Gestalt $\mathbf{F}\varphi_1$, $\mathbf{G}\varphi_1$, $\varphi_1 \mathbf{U} \varphi_2$ oder $\varphi_1 \mathbf{R} \varphi_2$. Gelte ferner, dass $v \models [\![\psi]\!]^t_b$ gdw. $\pi \models^t_b \psi$ für alle* echten *Teilformeln ψ von φ und alle $t \in \{0, \ldots, b-1\}$. Dann gilt*

$$\begin{aligned}
v \models \langle\!\langle \mathbf{F}\varphi_1 \rangle\!\rangle^t_b \quad & \textit{gdw.} \quad \textit{es ex. } j \in \mathbb{N},\ t \leq j < b \textit{ mit } \pi \models^j_b \varphi_1, \\
v \models \langle\!\langle \mathbf{G}\varphi_1 \rangle\!\rangle^t_b \quad & \textit{gdw.} \quad \textit{für alle } j \in \mathbb{N} \textit{ mit } t \leq j < b \textit{ gilt: } \pi \models^j_b \varphi_1, \\
v \models \langle\!\langle \varphi_1 \mathbf{U} \varphi_2 \rangle\!\rangle^t_b \quad & \textit{gdw.} \quad \textit{es ex. } j \in \mathbb{N},\ t \leq j < b \textit{ mit } \pi \models^j_b \varphi_2 \textit{ und} \\
& \qquad \textit{für alle } n \in \mathbb{N} \textit{ mit } t \leq n < j \textit{ gilt: } \pi \models^n_b \varphi_1, \\
v \models \langle\!\langle \varphi_1 \mathbf{R} \varphi_2 \rangle\!\rangle^t_b \quad & \textit{gdw.} \quad \textit{für alle } j \in \mathbb{N} \textit{ mit } t \leq j < b \textit{ gilt, dass } \pi \models^j_b \varphi_2 \textit{ oder} \\
& \qquad \textit{dass es ein } n \in \mathbb{N} \textit{ mit } t \leq n < j \textit{ und } \pi \models^n_b \varphi_1 \textit{ gibt.}
\end{aligned}$$

Beweis. Wir beweisen die Behauptung durch Induktion über den Zeitpunkt t, wobei wir bei dem Basisfall $t = b$ beginnen und rückwärts bis $t = 0$ laufen.

1. Basisfall $t = b$: Wir beweisen die Behauptung nur für Formeln der Gestalt $\mathbf{F}\varphi$. Der Beweis für $\mathbf{G}\varphi$, $\varphi_1 \mathbf{U} \varphi_2$ und $\varphi_1 \mathbf{R} \varphi_2$ erfolgt analog. Für alle LTL-Formeln φ gilt sowohl $v \not\models \langle\!\langle \mathbf{F}\varphi \rangle\!\rangle^b_b = \bot$ als auch, dass es kein $j \in \mathbb{N}$ mit $b \leq j < b$ und $\pi \models^j_b \varphi_1$ gibt.
2. Induktiver Fall $t < b$: Auch hier beweisen wir die Behauptung nur für Formeln der Gestalt $\mathbf{F}\varphi$, da der Beweis für $\mathbf{G}\varphi$, $\varphi_1 \mathbf{U} \varphi_2$ und $\varphi_1 \mathbf{R} \varphi_2$ wiederum analog erfolgt. Nach Definition gilt $v \models \langle\!\langle \mathbf{F}\varphi \rangle\!\rangle^t_b$ genau dann, wenn $v \models [\![\varphi]\!]^t_b$ oder $v \models \langle\!\langle \mathbf{F}\varphi \rangle\!\rangle^{t+1}_b$. Der erste Fall ist nach Voraussetzung äquivalent zu $\pi \models^t_b \varphi$, da φ eine echte Teilformel von $\mathbf{F}\varphi$ ist. Der zweite Fall ist nach Induktionsvoraussetzung äquivalent dazu, dass es ein $j \in \mathbb{N}$ mit $t+1 \leq j < b$ und $\pi \models^j_b \varphi$ gibt. Mindestens einer der beiden Fälle tritt genau dann ein, wenn es ein $j \in \mathbb{N}$ mit $t \leq j < b$ und $\pi \models^j_b \varphi$ gibt. □

Lemma 12. *Sei π* keine *Schleife. Dann gilt $v \models [\![\varphi]\!]_b^t$ gdw. $s_{0:b-1} \models_{b-1}^t \varphi$. Dabei ist $s_{0:b-1} = \pi(0), \ldots, \pi(b-1)$ der Pfad π ohne den letzten Zustand.*

Beweis. Wir beweisen die Behauptung durch Induktion über den Zeitpunkt t, wobei wir sowohl beim Induktionsanfang als auch im Induktionsschritt jeweils einen induktiven Beweis über den Aufbau von LTL-Formeln führen. Die Induktion über die Zeitpunkte beginnt bei $t = b$ und verläuft rückwärts bis $t = 0$.

1. Basisfall $t = b$: Bei der Induktion über den Aufbau von LTL-Formeln beschränken wir uns auf den Fall von Atomen, da für $t = b$ alle Arten von LTL-Formeln im wesentlichen gleich behandelt werden. Nach Definition gilt $v \models [\![a]\!]_b^b$ genau dann, wenn es ein $j \in \mathbb{N}$ mit $0 \leq j < b$, $v \models \mathit{loopto}_j$ und $v \models a_j$ gibt. Das ist immer falsch, da nach Lemma 10 und weil π keine Schleife ist, $v \not\models \mathit{loopto}_j$ für alle $j \in \mathbb{N}$ mit $0 \leq j < b$ gilt. Auch $s_{0:b-1} \models_{b-1}^b \varphi$ ist wegen $b - 1 < b$ immer falsch.
2. Induktiver Fall $t < b$: Bei der Induktion über den Aufbau von LTL-Formeln betrachten wir exemplarisch die Fälle von Atomen, Konjunktion, **F**-Operator und **U**-Operator.
   a) Basisfall $a \in A$: Es gilt $v \models [\![a]\!]_b^t = a_t$ genau dann, wenn $\pi(t) \models a$ genau dann, wenn $a \in L(\pi(t))$ genau dann, wenn $\pi \models_{b-1}^t a$ genau dann, wenn $s_{0:b-1} \models_{b-1}^t a$.
   b) Induktiver Fall $\varphi_1 \vee \varphi_2$: Nach Definition gilt $v \models [\![\varphi_1 \wedge \varphi_2]\!]_b^t$ genau dann, wenn $v \models [\![\varphi_1]\!]_b^t$ und $v \models [\![\varphi_2]\!]_b^t$. Das ist nach Induktionsvoraussetzung genau dann der Fall, wenn $s_{0:b-1} \models_{b-1}^t \varphi_1$ und $s_{0:b-1} \models_{b-1}^t \varphi_2$, d. h. $s_{0:b-1} \models_{b-1}^t \varphi_1 \wedge \varphi_2$
   c) Induktiver Fall $\mathbf{F}\varphi$: Nach Definition gilt $v \models [\![\mathbf{F}\varphi]\!]_b^t$ genau dann, wenn $v \models [\![\varphi]\!]_b^t$ oder $v \models [\![\mathbf{F}\varphi]\!]_b^{t+1}$. Im ersten Fall ist die Induktionsvoraussetzung der inneren Induktion über LTL-Formeln, im zweiten Fall die Induktionsvoraussetzung der äußeren Induktion über die Zeitpunkte anwendbar, d. h. die letzte Aussage ist äquivalent dazu, dass $s_{0:b-1} \models_{b-1}^t \varphi$ oder $s_{0:b-1} \models_{b-1}^{t+1} \mathbf{F}\varphi$. Das gilt genau dann, wenn $s_{0:b-1} \models_{b-1}^t \varphi$ oder es ein $j \in \mathbb{N}$ gibt mit $t + 1 \leq j < b$ und $s_{0:b-1} \models_{b-1}^j \varphi$, d. h. wenn es ein $j \in \mathbb{N}$ gibt mit $t \leq j < b$ und $s_{0:b-1} \models_{b-1}^j \varphi$. Das ist äquivalent dazu, dass $s_{0:b-1} \models_{b-1}^t \mathbf{F}\varphi$.
   d) Induktiver Fall $\varphi_1 \mathbf{U} \varphi_2$: Nach Definition gilt $v \models [\![\varphi_1 \mathbf{U} \varphi_2]\!]_b^t$ genau dann, wenn $[\![\varphi_2]\!]_b^t$ oder sowohl $[\![\varphi_1]\!]_b^t$ als auch $[\![\varphi_1 \mathbf{U} \varphi_2]\!]_b^{t+1}$ von v erfüllt wird. Die Erfülltheit der beiden ersten Formeln kann nach der inneren Induktionsvoraussetzung, die der dritten Formel nach der äußeren Induktionsvoraussetzung durch eine gleichwertige Bedingung ersetzt werden, d. h. $v \models [\![\varphi_1 \mathbf{U} \varphi_2]\!]_b^t$ genau dann, wenn $s_{0:b-1} \models_{b-1}^t \varphi_2$ oder sowohl $s_{0:b-1} \models_{b-1}^t \varphi_1$ als auch $s_{0:b-1} \models_{b-1}^{t+1} \varphi_1 \mathbf{U} \varphi_2$. Das ist genau dann der Fall, wenn $s_{0:b-1} \models_{b-1}^t \varphi_2$ oder sowohl $s_{0:b-1} \models_{b-1}^t \varphi_1$ als auch ein $j \in \mathbb{N}$ mit $t+1 \leq j < b$ existiert, so dass $s_{0:b-1} \models_{b-1}^j \varphi_2$ und dass für alle $n \in \mathbb{N}$ mit $t + 1 \leq n < j$ gilt, dass $s_{0:b-1} \models_{b-1}^n \varphi_1$. Das wiederum gilt genau dann, wenn $s_{0:b-1} \models_{b-1}^t \varphi_2$ oder ein $j \in \mathbb{N}$ mit $t + 1 \leq j < b$ existiert, so dass $s_{0:b-1} \models_{b-1}^j \varphi_2$ und dass für alle $n \in \mathbb{N}$ mit $t \leq n < j$ gilt, dass $s_{0:b-1} \models_{b-1}^n \varphi_1$. Das gilt genau dann, wenn ein $j \in \mathbb{N}$ mit $t \leq j < b$ existiert, so dass $s_{0:b-1} \models_{b-1}^j \varphi_2$ und dass für alle $n \in \mathbb{N}$ mit $t \leq n < j$ gilt, dass $s_{0:b-1} \models_{b-1}^n \varphi_1$. Dies gilt nach Definition genau dann, wenn $s_{0:b-1} \models_{b-1}^t \varphi_1 \mathbf{U} \varphi_2$. □

Lemma 13. *Sei π eine b-Schleife. Dann gilt $v \models [\![\varphi]\!]_b^t$ genau dann, wenn es ein solches $k \in \mathbb{N}$ gibt, dass π eine (b, k)-Schleife ist und $\pi_k^{\infty t} \models \varphi$. Insbesondere gilt $v \models [\![\varphi]\!]_b^0$ genau dann, wenn $\pi \models_b \varphi$.*

Beweis. Die Struktur des Beweises ist dieselbe wie beim Beweis des vorigen Lemmas. Wir beschränken uns sowohl beim Basisfall $t = b$ als auch im Induktionsschritt für $t < b$ exemplarisch auf den Beweis für Formeln der Gestalt $\mathbf{F}\varphi$.

1. Basisfall $t = b$: Nach Definition gilt $v \models [\![\mathbf{F}\varphi]\!]_b^b$ genau dann, wenn es ein $j \in \mathbb{N}$, $0 \leq j \leq b-1$ gibt, so dass $v \models \mathit{loopto}_j$ und $v \models \langle\!\langle \mathbf{F}\varphi \rangle\!\rangle_b^j$. Nach Lemma 10 und Definition 32 gilt das genau dann, wenn es $j \in \mathbb{N}$, $0 \leq j \leq b-1$, mit $\pi(j) = \pi(b)$ und $v \models \langle\!\langle \mathbf{F}\varphi \rangle\!\rangle_b^j$ gibt. Nach Lemma 11 gilt das genau dann, wenn es ein $j \in \mathbb{N}$, $0 \leq j \leq b-1$ gibt, so dass $\pi(j) = \pi(b)$ und dass es weiter ein $n \in \mathbb{N}$ mit $j \leq n \leq b-1$ gibt, so dass $\pi \models_b^n \varphi$. Das gilt genau dann, wenn $\pi(j) = \pi(b)$ und es ein $k \in \mathbb{N}$, nämlich $k = j-1$, gibt, so dass π eine (b,k)-Schleife ist und es ein $n \in \mathbb{N}$ mit $j \leq n \leq b-1$ gibt, so dass $\pi_k^{\infty\, n} \models \varphi$ gilt. Das ist äquivalent dazu, dass es ein $k \in \mathbb{N}$ gibt, so dass π eine (b,k)-Schleife ist und $\pi_k^{\infty\, b} \models \mathbf{F}\varphi$.
2. Induktiver Fall $t < b$: Nach Definition gilt $v \models [\![\mathbf{F}\varphi]\!]_b^t$ genau dann, wenn $v \models [\![\varphi]\!]_b^t$ oder $v \models [\![\mathbf{F}\varphi]\!]_b^{t+1}$. Nach Anwendung der beiden Induktionsvoraussetzungen ist das genau dann der Fall, wenn es ein solches $k \in \mathbb{N}$ gibt, dass π eine (b,k)-Schleife ist und $\pi_k^{\infty\, t} \models \varphi$ sowie $\pi_k^{\infty\,(t+1)} \models \mathbf{F}\varphi$. Nach der Definition des $\mathbf{F}$-Operators gilt zugleich $\pi_k^{\infty\, t} \models \varphi$ und $\pi_k^{\infty\,(t+1)} \models \mathbf{F}\varphi$ genau dann, wenn $\pi_k^{\infty\, t} \models \mathbf{F}\varphi$. □

Lemma 14. *Sei $\mathcal{P} = \langle A, I, O, g\rangle$ eine Planungsinstanz, φ eine* LTL*-Formel und $b \in \mathbb{N}$. Wenn die Formel $[\![\mathcal{P}, \varphi]\!]^b$ wie oben mit $[\![\mathcal{P}]\!]^b = [\![\mathcal{P}]\!]_{basic}^b \wedge [\![\mathcal{P}]\!]_{1lin}^{b,1}$ erfüllbar ist, so gibt es einen solchen 1-Linearisierungs-Plan $\Pi = \langle S_0, \ldots, S_{b-1}\rangle$ mit Ausführung $\pi = s_0, \ldots, s_b$ für $\mathcal{P}$, dass $\pi \models_b \varphi$, falls π eine Schleife ist, bzw. $s_{0:b-1} \models_{b-1} \varphi$, sonst.*

Beweis. Sei v eine Belegung mit $v \models [\![\mathcal{P}]\!]^b \wedge \mathit{loopConstraints}_b \wedge [\![\varphi]\!]_b^0$. Seien $\Pi^v = \langle S_0^v, \ldots, S_{b-1}^v\rangle$ und $\pi^v = s_0^v, \ldots, s_b^v$ wie in Definition 16. Nach Lemma 9 ist Π^v ein 1-Linearisierungs-Plan mit Ausführung π^v für $\mathcal{P}$, denn $v \models [\![\mathcal{P}]\!]^b$. Da $v \models \mathit{loopConstraints}_b \wedge [\![\varphi]\!]_b^0$, ist Lemma 12 (keine Schleife) bzw. Lemma 13 (Schleife) anwendbar, woraus die Behauptung folgt. □

2.4.4. Parallele Transitionen und Äquivalenz von Pfaden

Es ist möglich, die Effizienz von Modellprüfungs-Algorithmen dadurch zu erhöhen, dass man in einem Algorithmus nicht alle Pfade, sondern nur Repräsentanten von Klassen von Pfaden betrachtet *(Partial Order Reduction)*.

Definition 37 (Induzierte Zustandsmarkierung). Sei $\mathcal{P} = \langle A, I, O, g\rangle$ eine Planungsinstanz, $\mathfrak{F}(\mathcal{P}) = \langle Q, R_1, \ldots, R_n\rangle$ der von $\mathcal{P}$ induzierte Kripke-Rahmen und $A' \subseteq A$ eine Menge von aussagenlogischen Variablen. Die Abbildung $L_{A'} : Q \to 2^{A'}$ heißt **von A' induzierte Zustandsmarkierung** für $\mathfrak{F}$, falls $L_{A'}(s) = s \cap A'$ für alle $s \in Q$. Ist φ eine LTL-Formel, so ist $L_\varphi := L_{var(\varphi)}$ die **von φ induzierte Zustandsmarkierung** für $\mathfrak{F}$. Das von A' bzw. φ **induzierte Kripke-Modell** ist dann $\mathfrak{M}(\mathcal{P}, A') = \langle \mathfrak{F}(\mathcal{P}), L_{A'}\rangle$ bzw. $\mathfrak{M}(\mathcal{P}, \varphi) = \langle \mathfrak{F}(\mathcal{P}), L_\varphi\rangle$.

Die zwei folgenden Definitionen und der anschließende Satz sind im Wesentlichen aus dem Lehrbuch von Clarke u. a. [2002] entnommen.

Definition 38 (Äquivalenz von Pfaden). Seien $\pi = s_0, s_1, s_2, \ldots$ und $\tilde{\pi} = \tilde{s}_0, \tilde{s}_1, \tilde{s}_2, \ldots$ zwei unendliche Pfade in einem Kripke-Modell $\mathfrak{M} = \langle Q, R, L\rangle$. Dann heißen π und $\tilde{\pi}$ **äquivalent**, kurz $\pi \sim_L \tilde{\pi}$, falls es zwei unendliche Folgen natürlicher Zahlen $0 = i_0 < i_1 < i_2 < \ldots$

und $0 = j_0 < j_1 < j_2 < \ldots$ so gibt, dass für jedes $k \geq 0$ gilt:

$$L(s_{i_k}) = L(s_{i_k+1}) = \ldots = L(s_{i_{k+1}-1}) = L(\tilde{s}_{j_k}) = L(\tilde{s}_{j_k+1}) = \ldots = L(\tilde{s}_{j_{k+1}-1}).$$

Eine endliche Folge von Zuständen mit gleichen Markierungen heißt **Block**. Zwei Pfade sind äquivalent, wenn sie so in unendliche viele Blöcke zerlegt werden können, dass alle Zustände im k-ten Block des einen Pfades die gleichen Markierungen tragen wie die Zustände im k-ten Block des anderen Pfades. Die einander entsprechenden Blöcke auf den beiden Pfaden müssen dabei nicht gleich lang sein.

Sind $\pi = s_0, s_1, s_2, \ldots, s_{b-1}$ und $\tilde{\pi} = \tilde{s}_0, \tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_{\tilde{b}-1}$ endliche Pfade der Länge b bzw. $\tilde{b}$, so heißen π und $\tilde{\pi}$ äquivalent, kurz $\pi \sim_L^{<\infty} \tilde{\pi}$, falls es zwei endliche Folgen natürlicher Zahlen $0 = i_0 < i_1 < i_2 < \ldots < i_n = b$ und $0 = j_0 < j_1 < j_2 < \ldots < j_n = \tilde{b}$ so gibt, dass für jedes $k \in \mathbb{N}$ mit $0 \leq k \leq n-1$:

$$L(s_{i_k}) = L(s_{i_k+1}) = \ldots = L(s_{i_{k+1}-1}) = L(\tilde{s}_{j_k}) = L(\tilde{s}_{j_k+1}) = \ldots = L(\tilde{s}_{j_{k+1}-1}).$$

Ist die Zustandsmarkierung, bezüglich der zwei Pfade äquivalent sind, klar und ist ersichtlich, ob es sich um endliche oder unendliche Pfade handelt, so schreiben wir statt $\sim_L$ bzw. $\sim_L^{<\infty}$ kurz $\sim$.

Beispiel 39. Die folgenden Pfade π und $\tilde{\pi}$, die mit $p, \neg p, q, \neg q$ markiert sind, sind äquivalent:

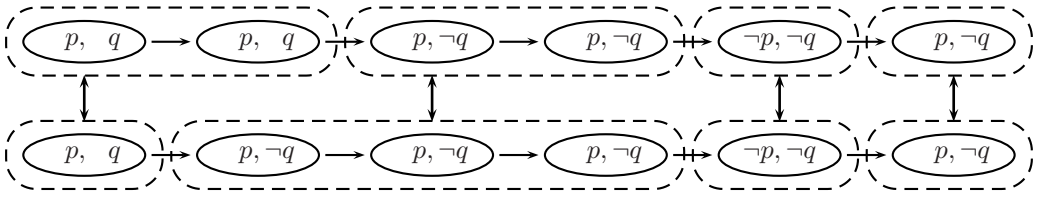


Abbildung 2.4.: Beispiel für zwei äquivalente Pfade

Definition 40 (Invarianz unter Stottern). Eine temporallogische Formel φ heißt **invariant unter Stottern**, wenn für jedes Paar von Pfaden π und $\tilde{\pi}$ mit $\pi \sim \tilde{\pi}$ gilt, dass

$$\pi \models \varphi \quad \text{gdw.} \quad \tilde{\pi} \models \varphi.$$

Satz 15. *Jede Formel $\varphi \in \mathsf{LTL}_{-\mathbf{X}}$ ist invariant unter Stottern. Genauer: Sei φ eine $\mathsf{LTL}_{-\mathbf{X}}$-Formel und $\mathfrak{M} = \langle Q, R, L_\varphi \rangle$ ein Kripke-Modell mit $Q = 2^A$ für eine Menge $A \supseteq var(\varphi)$ von aussagenlogischen Variablen. Seien weiter π und $\tilde{\pi}$ zwei unendliche Pfade in $\mathfrak{M}$ mit $\pi \sim \tilde{\pi}$. Dann gilt:*

$$\pi \models \varphi \quad \textit{gdw.} \quad \tilde{\pi} \models \varphi$$

Beweis. Der Beweis erfolgt durch Induktion über den Aufbau von $\mathsf{LTL}_{-\mathbf{X}}$-Formeln. Dabei genügt es, atomare Formeln, die booleschen Fälle $\neg$ und $\wedge$ sowie die Anwendung des Operators $\mathbf{U}$ zu betrachten (der Beweis für den $\mathbf{R}$-Operator erfolgt analog).

1. $\varphi = a \in A$: Es gilt $\pi \models a$ gdw. $a \in \pi(0)$ gdw. $a \in L_\varphi(\pi(0))$ gdw. $a \in L_\varphi(\tilde{\pi}(0))$ gdw. $a \in \tilde{\pi}(0)$ gdw. $\tilde{\pi} \models a$.

2. $\varphi = \neg a$: Es gilt $\pi \models \neg a$ gdw. $a \notin \pi(0)$ gdw. $a \notin L_\varphi(\pi(0))$ gdw. $a \notin L_\varphi(\tilde{\pi}(0))$ gdw. $a \notin \tilde{\pi}(0)$ gdw. $\tilde{\pi} \models \neg a$.
3. $\varphi = \varphi_1 \wedge \varphi_2$: Es gilt $\pi \models \varphi_1 \wedge \varphi_2$ gdw. $\pi \models \varphi_1$ und $\pi \models \varphi_2$ gdw. $\tilde{\pi} \models \varphi_1$ und $\tilde{\pi} \models \varphi_2$ gdw. $\tilde{\pi} \models \varphi_1 \wedge \varphi_2$.
4. $\varphi = \varphi_1 \mathbf{U} \varphi_2$: Angenommen, $\pi \models \varphi_1 \mathbf{U} \varphi_2$. Dann existiert ein $\ell \geq 0$, so dass $\pi^\ell \models \varphi_2$ und dass für alle m mit $0 \leq m < \ell$ gilt, dass $\pi^m \models \varphi_1$. Wegen $\pi \sim \tilde{\pi}$ gibt es zwei Folgen natürlicher Zahlen $0 = i_0 < i_1 < i_2 < \ldots$ und $0 = j_0 < j_1 < j_2 < \ldots$, so dass für jedes $k \geq 0$: $L(\pi(i_k)) = L(\pi(i_k + 1)) = \ldots = L(\pi(i_{k+1} - 1)) = L(\tilde{\pi}(j_k)) = L(\tilde{\pi}(j_k + 1)) = \ldots = L(\tilde{\pi}(j_{k+1} - 1))$.
   Sei $k \in \mathbb{N}$ so, dass $\pi(\ell)$ im k-ten Block von π liegt, d. h. $i_k \leq \ell < i_{k+1}$. O. B. d. A. ist dann $\ell = i_k$. Wähle nun $\tilde{\ell} \in \mathbb{N}$ so, dass $\tilde{\pi}(\tilde{\ell})$ der erste Zustand des k-ten Blocks von $\tilde{\pi}$ ist, d. h. $\tilde{\ell} = j_k$. Offensichtlich gilt $\pi^\ell \sim \tilde{\pi}^{\tilde{\ell}}$. Nach Induktionsvoraussetzung gilt daher, dass $\pi^\ell \models \varphi_2$ genau dann gilt, wenn $\tilde{\pi}^{\tilde{\ell}} \models \varphi_2$. Nach Voraussetzung folgt also $\tilde{\pi}^{\tilde{\ell}} \models \varphi_2$. Da alle Zustände vor $\pi(\ell)$ im 0-ten bis $(k-1)$-ten Block liegen, liefert eine analoge Argumentation, dass für alle $\tilde{m}$ mit $0 \leq \tilde{m} < \tilde{\ell}$ gilt, dass $\tilde{\pi}^{\tilde{m}} \models \varphi_1$. Also folgt $\tilde{\pi} \models \varphi_1 \mathbf{U} \varphi_2$. Der Schluss von $\tilde{\pi} \models \varphi_1 \mathbf{U} \varphi_2$ auf $\pi \models \varphi_1 \mathbf{U} \varphi_2$ erfolgt analog. □

Lemma 16. *Sei $\mathfrak{M} = \langle Q, R, L \rangle$ ein Kripke-Modell, $\pi = u \circ v$ eine (b,k)-Schleife in $\mathfrak{M}$, wobei $\|u\| = k$, und $\tilde{\pi} = \tilde{u} \circ \tilde{v}$ eine $(\tilde{b}, \tilde{k})$-Schleife in $\mathfrak{M}$, wobei $\|\tilde{u}\| = \tilde{k}$. Gilt $u \sim \tilde{u}$ und $v \sim \tilde{v}$, so gilt auch $\pi_k^\infty \sim \tilde{\pi}_{\tilde{k}}^\infty$.*

Beweis. Seien $u \sim \tilde{u}$ und $v \sim \tilde{v}$ mit den Blockgrenzen

$$0 = i_0^u < i_1^u < \ldots < i_n^u = k \quad \text{bzw.} \quad 0 = j_0^u < j_1^u < \ldots < j_n^u = \tilde{k} \quad \text{für } u \sim \tilde{u} \text{ und}$$
$$0 = i_0^v < i_1^v < \ldots < i_m^v = x \quad \text{bzw.} \quad 0 = j_0^v < j_1^v < \ldots < j_m^v = \tilde{x} \quad \text{für } v \sim \tilde{v},$$

wobei $x = b - k + 1$ und $\tilde{x} = \tilde{b} - \tilde{k} + 1$. Dann ist $\pi_k^\infty \sim \tilde{\pi}_{\tilde{k}}^\infty$ mit den Blockgrenzen

$$\begin{aligned} 0 = i_0^u < i_1^u < \ldots < i_n^u < k + i_1^v < \ldots < k + i_m^v < k + x + i_1^v < \ldots \\ < k + x + i_m^v < \cdots < k + rx + i_1^v < \ldots < k + rx + i_m^v < \ldots \quad \text{in } \pi_k^\infty \text{ bzw.} \\ 0 = j_0^u < j_1^u < \ldots < j_n^u < \tilde{k} + j_1^v < \ldots < \tilde{k} + j_m^v < \tilde{k} + \tilde{x} + j_1^v < \ldots \\ < \tilde{k} + \tilde{x} + j_m^v < \cdots < \tilde{k} + r\tilde{x} + j_1^v < \ldots < \tilde{k} + r\tilde{x} + j_m^v < \ldots \quad \text{in } \tilde{\pi}_{\tilde{k}}^\infty. \end{aligned}$$

□

Lemma 17. *Sei φ eine $\mathsf{LTL}_{-\mathbf{X}}$-Formel und $\mathfrak{M} = \langle Q, R, L_\varphi \rangle$ ein Kripke-Modell mit $Q = 2^A$ für eine Menge $A \supseteq var(\varphi)$ von aussagenlogischen Variablen. Seien ferner $\pi = u \circ v$ eine (b,k)-Schleife in $\mathfrak{M}$, wobei $\|u\| = k$, und $\tilde{\pi} = \tilde{u} \circ \tilde{v}$ eine $(\tilde{b}, \tilde{k})$-Schleife in $\mathfrak{M}$, wobei $\|\tilde{u}\| = \tilde{k}$. Gilt $u \sim \tilde{u}$ und $v \sim \tilde{v}$, so folgt, dass*

$$\pi \models_b \varphi \quad \textit{gdw.} \quad \tilde{\pi} \models_{\tilde{b}} \varphi.$$

Beweis. Nach Lemma 16 gilt $\pi_k^\infty \sim \tilde{\pi}_{\tilde{k}}^\infty$. Da π_k^∞ und $\tilde{\pi}_{\tilde{k}}^\infty$ *unendliche* Pfade sind, ist nun Satz 15 anwendbar. Damit folgt, dass $\pi_k^\infty \models \varphi$ genau dann, wenn $\tilde{\pi}_{\tilde{k}}^\infty \models \varphi$. Nach Definition 32 folgt, dass $\pi \models_b \varphi$ genau dann, wenn $\tilde{\pi} \models_{\tilde{b}} \varphi$. □

Lemma 18. *Sei φ eine $\mathsf{LTL}_{-\mathbf{X}}$-Formel und $\mathfrak{M} = \langle Q, R, L_\varphi \rangle$ ein Kripke-Modell mit $Q = 2^A$ für eine Menge $A \supseteq var(\varphi)$ von aussagenlogischen Variablen. Seien weiter π und $\tilde{\pi}$ zwei endliche Pfade in $\mathfrak{M}$ der Länge b bzw. $\tilde{b}$, die keine Schleifen sind, mit $\pi \sim \tilde{\pi}$. Dann gilt:*

$$\pi(0), \ldots, \pi(b-1) \models_{b-1} \varphi \quad \textit{gdw.} \quad \tilde{\pi}(0), \ldots, \tilde{\pi}(\tilde{b}-1) \models_{\tilde{b}-1} \varphi.$$

Beweis. Analog zum Beweis von Lemma 15. □

2.5. Berechnungskomplexität

Das STRIPS-Planungsproblem mit deterministischen Operatoren [Bylander, 1994] ist, ebenso wie das LTL- und das $\mathsf{LTL}_{-\mathbf{X}}$-Modellprüfungs-Problem [Sistla und Clarke, 1985], **PSPACE**-vollständig. Das Erfüllbarkeitsproblem der Aussagenlogik ist hingegen nur **NP**-vollständig, wobei $\mathbf{NP} \subseteq \mathbf{PSPACE}$ gilt und $\mathbf{NP} \neq \mathbf{PSPACE}$ vermutet wird [Papadimitriou, 1994]. Dass eine Reduktion des Planungs- bzw. des Modellprüfungs-Problems auf das Erfüllbarkeitsproblem dennoch möglich ist, lässt sich dadurch erklären, dass ein kürzester Plan bzw. ein kürzestes Gegenbeispiel exponentielle Länge in der Instanzgröße haben kann und damit auch die Reduktion, die eine exponentiell große aussagenlogische Formel für die entsprechende Planlänge erzeugen muss, im schlimmsten Fall exponentielle Laufzeit hat.

2.6. Ein größeres Beispiel

In diesem Abschnitt wollen wir ein Beispiel aus der LOGISTICS-Planungsdomäne [McDermott, Anderson, Köhler, Selman und Kautz, 1998] etwas genauer betrachten. Instanzen der LOGISTICS-Domäne bestehen aus einer Menge von Städten, die jeweils über eine Menge von Depots sowie je höchstens einen Flughafen verfügen. In jeder Stadt gibt es einen LKW mit unbegrenzter Kapazität, der Pakete innerhalb der Stadt, jedoch nicht über die Stadtgrenzen hinaus transportieren kann. Je zwei Punkte in einer Stadt sind direkt miteinander verbunden, d.h. mit einer `driveTruck`-Aktion voneinander aus zu erreichen. Zum Transport von Paketen zwischen Städten können Flugzeuge mit unbegrenzter Kapazität zur Verfügung stehen, die mit einer `flyAirplane`-Aktion zwischen zwei Flughäfen verkehren können. Die Aktionen `loadTruck`, `unloadTruck`, `loadAirplane` und `unloadAirplane` dienen dazu, einen LKW bzw. das Flugzeug zu be- bzw. entladen. Zustände werden durch die Prädikate $\texttt{at}(\texttt{obj}, \texttt{depot})$, d. h. das Object `obj`, ein LKW oder ein Paket, befindet sich in Depot `depot`, und $\texttt{in}(\texttt{package}, \texttt{truck})$, d. h. das Paket `package` befindet sich in LKW `truck`, beschrieben.

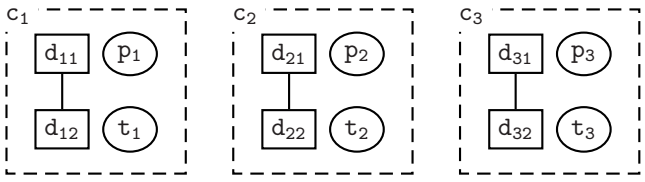


Abbildung 2.5.: LOGISTICS-Beispielinstanz

In unserem Beispiel, das in Abbildung 2.5 dargestellt ist, gibt es die drei Städte $\mathtt{c}_1$, $\mathtt{c}_2$ und $\mathtt{c}_3$. Jede Stadt $\mathtt{c}_i$ besitzt genau zwei Depots, nämlich $\mathtt{d}_{i1}$ und $\mathtt{d}_{i2}$ sowie einen LKW $\mathtt{t}_i$, der sich am Anfang in $\mathtt{d}_{i2}$ befindet. In jedem $\mathtt{d}_{i1}$ gibt es außerdem ein Paket $\mathtt{p}_i$. Das Ziel ist, dass sich unendlich oft gleichzeitig alle Pakete $\mathtt{p}_i$ in ihren jeweiligen Depots $\mathtt{d}_{i1}$ befinden, jedoch auch unendlich oft gleichzeitig alle Pakete $\mathtt{p}_i$ in ihren jeweiligen Depots $\mathtt{d}_{i2}$ sind ($i \in \{1, 2, 3\}$).

Dieses Ziel kann durch die folgende LTL-Formel spezifiziert werden:

$$\begin{aligned}\varphi \quad = \quad & \mathbf{GF}(\mathtt{at}(\mathtt{p}_1, \mathtt{d}_{11}) \wedge \mathtt{at}(\mathtt{p}_2, \mathtt{d}_{21}) \wedge \mathtt{at}(\mathtt{p}_3, \mathtt{d}_{31})) \wedge \\ & \mathbf{GF}(\mathtt{at}(\mathtt{p}_1, \mathtt{d}_{12}) \wedge \mathtt{at}(\mathtt{p}_2, \mathtt{d}_{22}) \wedge \mathtt{at}(\mathtt{p}_3, \mathtt{d}_{32}))\end{aligned}$$

Ein Plan, der dieses Ziel erreicht, muss immer einen zyklischen Ausführungspfad besitzen, da nur so gewährleistet ist, dass bestimmte Aussagen *unendlich* oft gelten. Ein kürzester Plan ohne parallele Aktionen, der die Instanz löst und dessen Ausführung φ erfüllt, ist

$$\begin{aligned}\Pi \quad = \quad & \langle \{\, \mathtt{driveTruck}(\mathtt{t}_1, \mathtt{d}_{12}, \mathtt{d}_{11}) \,\}, \{\, \mathtt{driveTruck}(\mathtt{t}_2, \mathtt{d}_{22}, \mathtt{d}_{21}) \,\}, \{\, \mathtt{driveTruck}(\mathtt{t}_3, \mathtt{d}_{32}, \mathtt{d}_{31}) \,\}, \\ & \{\, \mathtt{loadTruck}(\mathtt{p}_1, \mathtt{t}_1, \mathtt{d}_{11}) \,\}, \{\, \mathtt{driveTruck}(\mathtt{t}_1, \mathtt{d}_{11}, \mathtt{d}_{12}) \,\}, \{\, \mathtt{loadTruck}(\mathtt{p}_2, \mathtt{t}_2, \mathtt{d}_{21}) \,\}, \\ & \{\, \mathtt{driveTruck}(\mathtt{t}_2, \mathtt{d}_{21}, \mathtt{d}_{22}) \,\}, \{\, \mathtt{loadTruck}(\mathtt{p}_3, \mathtt{t}_3, \mathtt{d}_{31}) \,\}, \{\, \mathtt{driveTruck}(\mathtt{t}_3, \mathtt{d}_{31}, \mathtt{d}_{32}) \,\}, \\ & \{\, \mathtt{unloadTruck}(\mathtt{p}_1, \mathtt{t}_1, \mathtt{d}_{12}) \,\}, \{\, \mathtt{unloadTruck}(\mathtt{p}_2, \mathtt{t}_2, \mathtt{d}_{22}) \,\}, \{\, \mathtt{unloadTruck}(\mathtt{p}_3, \mathtt{t}_3, \mathtt{d}_{32}) \,\}, \\ & \{\, \mathtt{loadTruck}(\mathtt{p}_1, \mathtt{t}_1, \mathtt{d}_{12}) \,\}, \{\, \mathtt{driveTruck}(\mathtt{t}_1, \mathtt{d}_{12}, \mathtt{d}_{11}) \,\}, \{\, \mathtt{loadTruck}(\mathtt{p}_2, \mathtt{t}_2, \mathtt{d}_{22}) \,\}, \\ & \{\, \mathtt{driveTruck}(\mathtt{t}_2, \mathtt{d}_{22}, \mathtt{d}_{21}) \,\}, \{\, \mathtt{loadTruck}(\mathtt{p}_3, \mathtt{t}_3, \mathtt{d}_{32}) \,\}, \{\, \mathtt{driveTruck}(\mathtt{t}_3, \mathtt{d}_{32}, \mathtt{d}_{31}) \,\}, \\ & \{\, \mathtt{unloadTruck}(\mathtt{p}_1, \mathtt{t}_1, \mathtt{d}_{11}) \,\}, \{\, \mathtt{unloadTruck}(\mathtt{p}_2, \mathtt{t}_2, \mathtt{d}_{21}) \,\}, \{\, \mathtt{unloadTruck}(\mathtt{p}_3, \mathtt{t}_3, \mathtt{d}_{31}) \,\} \rangle.\end{aligned}$$

Graphisch lässt sich der Plan bzw. seine Ausführung wie folgt darstellen. Beachte, dass wir in dieser und späteren Abbildungen, um die Darstellung übersichtlich zu halten, die folgenden Abkürzungen einführen: $\mathtt{dt}_{ij} := \mathtt{driveTruck}(\mathtt{t}_i, \mathtt{d}_{i(3-j)}, \mathtt{d}_{ij})$, $\mathtt{dt}_j := \mathtt{dt}_{1j}$, $\mathtt{l}_{ij} := \mathtt{loadTruck}(\mathtt{p}_i, \mathtt{t}_i, \mathtt{d}_{ij})$, $\mathtt{l}_j := \mathtt{l}_{1j}$, $\mathtt{u}_{ij} := \mathtt{unloadTruck}(\mathtt{p}_i, \mathtt{t}_i, \mathtt{d}_{ij})$ und $\mathtt{u}_j := \mathtt{u}_{1j}$ für $i \in \{\, 1, 2, 3 \,\}$ und $j \in \{\, 1, 2 \,\}$ sowie $\mathtt{p} := \mathtt{p}_1$, $\mathtt{t} := \mathtt{t}_1$, $\mathtt{d}_1 := \mathtt{d}_{11}$ und $\mathtt{d}_2 := \mathtt{d}_{12}$.

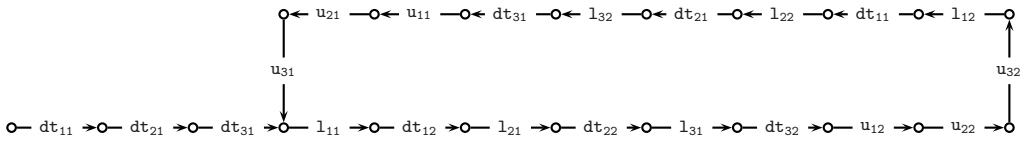


Abbildung 2.6.: Sequentielle Planausführung im LOGISTICS-Beispiel

2.7. SAT-Löser

Bei dem hier vorgestellten Ansatz, Handlungsplanung bzw. Modellprüfung zu betreiben, werden für den Erfüllbarkeitstest bzw. ggf. die Ermittlung einer erfüllenden Belegung für die erzeugten aussagenlogischen Formeln spezielle Programme, sogenannte SAT-Löser (*SAT-Solver*), benötigt. Die wichtigsten Verfahren und Heuristiken, die in den meisten SAT-Lösern implementiert sind, werden z. B. von Ryan [2004] beschrieben.

Bei unseren Experimenten, die in Abschnitt 4.2 beschrieben werden, kam der SAT-Löser SIEGE-V4 [Ryan, 2004] zum Einsatz. Darüber hinaus wurde mit ZCHAFF experimentiert. Wegen der meist schlechteren Ergebnisse wurden diese Experimente jedoch nicht dokumentiert.

Kapitel 3.

Modellprüfung und parallele Transitionen

In diesem Kapitel wollen wir beschreiben, wie es möglich ist, $\mathsf{LTL}_{-\mathbf{X}}$-Modellprüfung durch Übersetzung in das aussagenlogische Erfüllbarkeitsproblem zu betreiben und dabei parallele Transitionen zuzulassen. In den Abschnitten 3.2 und 3.3 geben wir einige Definitionen an und entwickeln darauf aufbauend hinreichende Bedingungen dafür, dass zwei Operatoren parallel angewandt werden dürfen, wenn man garantieren will, dass die parallele Ausführung eines Planes eine vorgegebenen temporallogische Formel genau dann erfüllt, wenn die Ausführung einer zulässigen Linearisierung dies tut. In Abschnitt 3.4 geben wir mehrere Kodierungen dieser hinreichenden Bedingungen als aussagenlogische Formeln an und beweisen in Abschnitt 3.5 die Korrektheit dieser Kodierungen. In Abschnitt 3.7 diskutieren wir die asymptotischen Größen der Kodierungen und das Maß an Parallelität, das sie erlauben.

3.1. Motivation

Bisher haben wir nur beschrieben, wie eine temporallogische Spezifikation einer Planausführung überprüft werden kann, wenn alle Zustände in der Ausführung des Planes explizit vorliegen. Dies kann man bei der im vorigen Kapitel vorgestellten Kodierung erreichen, indem man der Basiskodierung einer Planungsinstanz Konjunktionsglieder hinzufügt, die gewährleisten, dass zu jedem Zeitpunkt nur ein Operator angewandt wird, d. h. dass der berechnete Plan tatsächlich ein sequentieller Plan ist.

Will man jedoch einen Plan finden, dessen Ausführung zugleich eine gegebene temporallogische Spezifikation erfüllt und parallele Operatoren erlaubt, muss man zu den bereits vorgenommenen Einschränkungen an die parallele Anwendbarkeit von Operatoren weitere Restriktionen hinzufügen. Denn ohne weitere Einschränkungen der Parallelität könnte ein Planer durch ungünstige Wahl der Operatormengen S_t korrekte Pläne übersehen oder fälschlicherweise solche Folgen von Operatormengen als Pläne ausgeben, zu denen es keine zulässige Linearisierung gibt, deren Ausführung die gegebene Spezifikation erfüllt.

Betrachte dazu die folgenden Beispiele:

- Sei s_t der aktuelle Zustand und seien in s_t die Variablen a_1 und a_2 wahr. Sei ferner die Spezifikation so, dass in s_t die Formel $\mathbf{F}(\neg a_1 \wedge a_2)$ gelten soll. Werden nun die beiden Operatoren $o_1 = \langle \top, \{ \neg a_1 \}, \varnothing \rangle$ und $o_2 = \langle \top, \{ \neg a_2 \}, \varnothing \rangle$ – evtl. neben anderen Operatoren – simultan angewandt, ergibt sich ein Zustand s_{t+1}, in dem sowohl a_1 als auch a_2 falsch ist. Die Anwendung der Operatoren in der Reihenfolge $\ldots; o_1; \ldots; o_2; \ldots$ ist, sofern die sonstigen Operatoren unproblematisch sind, eine zulässige Linearisierung von S_t. Bei dieser Linearisierung gibt es zwischen der Anwendung von o_1 und o_2 einen

Zwischenzustand q, in dem $\neg a_1 \wedge a_2$ gilt; also ist für mindestens eine Linearisierung in s_t die Formel $\mathbf{F}(\neg a_1 \wedge a_2)$ erfüllt. Diese Tatsache wird jedoch von einem Planer übersehen, wenn nur die Folge $s_1, \ldots, s_t, s_{t+1}, \ldots$ betrachtet wird, denn in s_{t+1} und evtl. auch in folgenden Zuständen gilt $\neg a_1 \wedge \neg a_2$.

- Seien s_t, s_{t+1}, o_1 und o_2 wie oben. Soll s_t die Formel $\mathbf{G}(a_1 \leftrightarrow a_2)$ erfüllen, so wird ein vermeintlicher Plan mit $S_t \supseteq \{ o_1, o_2 \}$ und $S_{t+i} = \varnothing$, $i \in \mathbb{N} \setminus 0$, bisher nicht ausgeschlossen, obwohl er keine zulässige Linearisierung besitzt, deren Ausführung in s_t die Formel $\mathbf{G}(a_1 \leftrightarrow a_2)$ erfüllt.

Die folgenden Definitionen sollen dabei helfen, weitere Einschränkungen an die erlaubte Parallelität von Operatoren zu formulieren und in Aussagenlogik zu kodieren.

3.2. Definitionen

Zunächst definieren wir, wann ein bedingter oder unbedingter Effekt eines Operators für eine $\mathsf{LTL}_{-\mathbf{X}}$-Spezifikation relevant ist. Ist er dies nicht, so ist er im Hinblick auf die oben beschriebenen Schwierigkeiten unproblematisch.

Definition 41. Sei o ein Operator, A eine Menge von Aussagenvariablen, s ein Zustand und φ eine LTL-Formel. Dann definieren wir die **Mengen aller bzw. der unbedingten Effekte von o, die für A (bzw. φ) relevant sind**, als

$$[o]^A_\Diamond := [o]_\Diamond \cap \mathit{Lit}(A), \qquad [o]^A_\Box := [o]_\Box \cap \mathit{Lit}(A),$$
$$[o]^\varphi_\Diamond := [o]^{\mathit{var}(\varphi)}_\Diamond, \qquad [o]^\varphi_\Box := [o]^{\mathit{var}(\varphi)}_\Box.$$

Zusätzlich zu den Effekten von Operatoren kann man die **Änderungen** definieren, die ein Operator in einem gegebenen Zustand verursacht, d. h. die Effekte, die sich von den im Ausgangszustand bereits geltenden Literalen unterscheiden:

$$ {}_s[o]_\Diamond := [o]_\Diamond \setminus l(s), \qquad {}_s[o]_\Box := [o]_\Box \setminus l(s),$$
$$ {}_s[o]^A_\Diamond := [o]^A_\Diamond \setminus l(s), \qquad {}_s[o]^A_\Box := [o]^A_\Box \setminus l(s),$$
$$ {}_s[o]^\varphi_\Diamond := {}_s[o]^{\mathit{var}(\varphi)}_\Diamond, \qquad {}_s[o]^\varphi_\Box := {}_s[o]^{\mathit{var}(\varphi)}_\Box.$$

Beispiel 42. Sei $o = \langle \top, \{ a_2, a_3 \}, \{ a_1 \wedge a_2 \rhd \{ \neg a_1, \neg a_4 \}, a_1 \wedge \neg a_2 \rhd \{ a_4 \} \} \rangle$ ein Operator, $\varphi = \mathbf{F}a_1 \wedge \mathbf{G}a_2$ eine $\mathsf{LTL}_{-\mathbf{X}}$-Formel und $s = \{ a_1 \mapsto 1, a_2 \mapsto 0, a_3 \mapsto 1, a_4 \mapsto 0 \}$ ein Zustand. Dann ist $\mathit{Lit}(\mathit{var}(\varphi)) = \{ a_1, \neg a_1, a_2, \neg a_2 \}$. Damit gilt:

$$[o]_\Diamond = \{ \neg a_1, a_2, a_3, a_4, \neg a_4 \}, \qquad [o]_\Box = \{ a_2, a_3 \},$$
$$[o]^\varphi_\Diamond = \{ \neg a_1, a_2 \}, \qquad [o]^\varphi_\Box = \{ a_2 \},$$
$$ {}_s[o]_\Diamond = \{ \neg a_1, a_2, a_4 \}, \qquad {}_s[o]_\Box = \{ a_2 \},$$
$$ {}_s[o]^\varphi_\Diamond = \{ \neg a_1, a_2 \}, \qquad {}_s[o]^\varphi_\Box = \{ a_2 \}.$$

In den nächsten beiden Lemmata werden die offensichtlichen Tatsachen formalisiert, dass jeder unbedingte Effekt insbesondere ein bedingter *oder* unbedingter Effekt ist (Lemma 19), dass bei STRIPS-Operatoren jeder Effekt ein unbedingter Effekt, d. h. die Menge der unbedingten Effekte bereits die Menge aller Effekte ist (Lemma 20) und dass sich diese Tatsache auch auf *relevante* Effekte und Änderungen überträgt.

Lemma 19. *Ist o ein Operator, so gilt*

$$[o]^A_\Box \subseteq [o]^A_\Diamond \quad \textit{bzw.} \quad [o]^\varphi_\Box \subseteq [o]^\varphi_\Diamond \qquad \textit{und} \qquad {}_s[o]^A_\Box \subseteq {}_s[o]^A_\Diamond \quad \textit{bzw.} \quad {}_s[o]^\varphi_\Box \subseteq {}_s[o]^\varphi_\Diamond$$

für alle Mengen A von Aussagenvariablen, alle LTL*-Formeln φ und alle Zustände s.* □

Lemma 20. *Ist o ein* STRIPS*-Operator, so gilt*

$$[o]_\Diamond = [o]_\Box, \qquad [o]^A_\Diamond = [o]^A_\Box \qquad \textit{und} \qquad [o]^\varphi_\Diamond = [o]^\varphi_\Box \qquad \textit{bzw.}$$
$${}_s[o]_\Diamond = {}_s[o]_\Box, \qquad {}_s[o]^A_\Diamond = {}_s[o]^A_\Box \qquad \textit{und} \qquad {}_s[o]^\varphi_\Diamond = {}_s[o]^\varphi_\Box$$

für alle Mengen A von Aussagenvariablen, alle LTL*-Formeln φ und alle Zustände s.* □

Definition 43. Sei φ eine LTL-Formel und $\mathcal{P} = \langle A, I, O, g\rangle$ eine Planungsinstanz. Dann sei $O(\varphi)$ die Menge aller Operatoren, die eine Variable in φ verändern können, d. h.

$$O(\varphi) := \left\{\, o \in O \mid [o]^\varphi_\Diamond \neq \varnothing \,\right\}.$$

Enthält $\mathcal{P}$ nur STRIPS-Operatoren, so ist $O(\varphi) = \{\, o \in O \mid [o]^\varphi_\Box \neq \varnothing \,\}$.

Die folgende Definition führt verschieden strenge Begriffe gleichen Verhaltens von Operatoren ein. Später soll gezeigt werden, dass gleiches Verhalten von Operatoren hinreichend dafür ist, dass sie simultan angewandt werden dürfen (sofern sie zusätzlich die bereits in Abschnitt 2.3.3 formulierten Einschränkungen an ihre Parallelisierbarkeit erfüllen).

Definition 44. Seien o und o' Operatoren und sei A eine Menge von Aussagenvariablen. Dann

1. **verhalten sich o und o' gleich bezüglich** A, falls

$$[o]^A_\Box = [o']^A_\Box, \qquad \text{und} \qquad [o]^A_\Diamond \setminus [o]^A_\Box = [o']^A_\Diamond \setminus [o']^A_\Box = \varnothing.$$

   Im Fall von STRIPS-Operatoren ist die zweite Bedingung immer erfüllt.
2. **verhalten sich o und o' in s gleich bezüglich** A, falls

$${}_s[o]^A_\Box = {}_s[o']^A_\Box, \qquad \text{und} \qquad {}_s[o]^A_\Diamond \setminus {}_s[o]^A_\Box = {}_s[o']^A_\Diamond \setminus {}_s[o']^A_\Box = \varnothing.$$

   Auch hier ist im Fall von STRIPS-Operatoren ist die zweite Bedingung immer erfüllt.

Wie oben kann diese Definition von Variablenmengen A auf LTL-Formeln φ übertragen werden, indem man $A = var(\varphi)$ setzt. Unter gleichem Verhalten einer Menge O von Operatoren wollen wir paarweise gleiches Verhalten für alle $o, o' \in O$ verstehen.

Für die folgende Definition nehmen wir an, dass die Operatoren bereits – im einfachsten Fall willkürlich, sonst möglicherweise mit Hilfe einer geeigneten Heuristik – vorsortiert sind. Wir definieren dann, unter welchen Umständen Mengen von Operatoren in dieser Vorsortierung zulässig sind. Wie oben beim Begriff des gleichen Verhaltens von Operatoren ist Zulässigkeit einer Operatormenge in einer gegebenen Reihenfolge zusammen mit den Einschränkungen aus Abschnitt 2.3.3 hinreichend dafür, dass die Operatoren simultan angewandt werden können.

Definition 45. Sei $S = \left\{\, o^1, \ldots, o^{|S|} \,\right\}$ und $o^1 \prec \cdots \prec o^{|S|}$ eine totale Ordnung auf S. Seien ferner s ein Zustand und φ eine LTL-Formel. Dann

1. ist S **bezüglich** φ **und** $\prec$ **zulässig**, falls für alle $j, k \in \{1, \ldots, |S|\}$ mit $j < k$ gilt:
$$[o^k]^\varphi_\Diamond \subseteq [o^j]^\varphi_\Box.$$
2. ist S **in** s **bezüglich** φ **und** $\prec$ **zulässig**, falls für alle $j, k \in \{1, \ldots, |S|\}$ mit $j < k$ gilt:
$${}_s[o^k]^\varphi_\Diamond \subseteq {}_s[o^j]^\varphi_\Box.$$

3.3. Hinreichende Bedingungen für Äquivalenz

In diesem Abschnitt geben wir unterschiedlich starke hinreichende Bedingungen dafür an, dass die Ausführung einer zulässigen Linearisierung eines parallelen Planes eine gegebene LTL-Formel genau dann erfüllt, wenn schon die parallele Ausführung, d. h. die Abfolge der expliziten Zwischenzustände, diese erfüllt.

Lemma 21. *Sei O eine Menge von Operatoren, $S \subseteq O$, s ein Zustand über den Zustandsvariablen A, φ eine* LTL*-Formel mit $var(\varphi) \subseteq A$ und seien die folgenden Aussagen gegeben:*

(i) Wenn $S \cap O(\varphi) \neq \varnothing$, dann $|S| = 1$.
(ii) Für alle $o, o' \in S$ gilt: Ist $o \neq o'$, so verhalten sich o und o' bzgl. φ gleich.
(iii) S ist bezüglich φ und bezüglich jeder Ordnung $\prec$ zulässig.
(iv) Für alle $o, o' \in S$ gilt: Ist $o \neq o'$, so verhalten sich o und o' in s bzgl. φ gleich.
(v) S ist in s bezüglich φ und bezüglich jeder Ordnung $\prec$ zulässig.

Dann gilt (i) $\Rightarrow$ (ii) $\Leftrightarrow$ (iii) $\Rightarrow$ (iv) $\Leftrightarrow$ (v)

Beweis. $(i) \Rightarrow (ii)$: Seien $o, o' \in S$ und $o \neq o'$. Dann ist $|S| > 1$ und damit nach (i) $S \cap O(\varphi) = \varnothing$. Wegen $o, o' \in S$ folgt $o, o' \notin O(\varphi)$, d. h. $[o]^\varphi_\Diamond = [o']^\varphi_\Diamond = \varnothing$. Damit ist auch $[o]^\varphi_\Box = [o']^\varphi_\Box = \varnothing$ und $[o]^\varphi_\Diamond \setminus [o]^\varphi_\Box = [o']^\varphi_\Diamond \setminus [o']^\varphi_\Box = \varnothing$, d. h. o und o' verhalten sich bzgl. φ gleich.

$(ii) \Rightarrow (iii)$: Nach Voraussetzung ist $[o]^\varphi_\Diamond \setminus [o]^\varphi_\Box = [o']^\varphi_\Diamond \setminus [o']^\varphi_\Box = \varnothing$ für alle $o, o' \in S$ mit $o \neq o'$. Mit $[o]^\varphi_\Box \subseteq [o]^\varphi_\Diamond$ für alle o folgt, dass $[o]^\varphi_\Box = [o]^\varphi_\Diamond$ für alle o. Mit der zweiten Voraussetzung $[o]^\varphi_\Box = [o']^\varphi_\Box$ für $o, o' \in S$ mit $o \neq o'$ folgt für alle $o, o' \in S$ mit $o \neq o'$: $[o]^\varphi_\Diamond = [o']^\varphi_\Box$, insbesondere $[o]^\varphi_\Diamond \subseteq [o']^\varphi_\Box$.

$(iii) \Rightarrow (ii)$: S ist genau dann bezüglich φ und bezüglich jeder Ordnung $\prec$ zulässig, wenn für alle $o, o' \in S$ mit $o \neq o'$ gilt: $[o]^\varphi_\Diamond \subseteq [o']^\varphi_\Box$. Wegen $[o]^\varphi_\Box \subseteq [o]^\varphi_\Diamond$ für alle o folgt daraus $[o]^\varphi_\Diamond \subseteq [o']^\varphi_\Box \subseteq [o']^\varphi_\Diamond \subseteq [o]^\varphi_\Box \subseteq [o]^\varphi_\Diamond$ und damit Gleichheit aller dieser Mengen für alle $o, o' \in S$ mit $o \neq o'$. Daraus folgt $[o]^\varphi_\Box = [o']^\varphi_\Box$ und $[o]^\varphi_\Diamond \setminus [o]^\varphi_\Box = [o']^\varphi_\Diamond \setminus [o']^\varphi_\Box = \varnothing$ für alle $o, o' \in S$ mit $o \neq o'$.

$(iii) \Rightarrow (iv)$: Nach (iii) ist $[o]^\varphi_\Diamond = [o']^\varphi_\Box = [o']^\varphi_\Diamond = [o]^\varphi_\Box$ für alle $o, o' \in S$ mit $o \neq o'$. Dann ist nach Definition auch ${}_s[o]^\varphi_\Box \subseteq {}_s[o']^\varphi_\Diamond$ und ${}_s[o]^\varphi_\Diamond \setminus {}_s[o]^\varphi_\Box = {}_s[o']^\varphi_\Diamond \setminus {}_s[o']^\varphi_\Box = \varnothing$ für alle $o, o' \in S$ mit $o \neq o'$.

$(iv) \Rightarrow (v)$: Nach (iv) ist nach der zweiten Bedingung ${}_s[o]^\varphi_\Diamond \subseteq {}_s[o]^\varphi_\Box$ und nach der ersten ${}_s[o]^\varphi_\Box \subseteq {}_s[o']^\varphi_\Box$ für alle $o, o' \in S$ mit $o \neq o'$, also auch ${}_s[o]^\varphi_\Diamond \subseteq {}_s[o']^\varphi_\Box$ für alle $o, o' \in S$ mit $o \neq o'$, also Aussage (v).

$(v) \Rightarrow (iv)$: Nach (v) folgt ${}_s[o]^\varphi_\Diamond \subseteq {}_s[o']^\varphi_\Box \subseteq {}_s[o']^\varphi_\Diamond \subseteq {}_s[o]^\varphi_\Box \subseteq {}_s[o]^\varphi_\Diamond$ und damit Gleichheit aller dieser Mengen für alle $o, o' \in S$ mit $o \neq o'$. Dann ist nach Definition auch ${}_s[o]^\varphi_\Box \subseteq {}_s[o']^\varphi_\Diamond$ und ${}_s[o]^\varphi_\Diamond \setminus {}_s[o]^\varphi_\Box = {}_s[o']^\varphi_\Diamond \setminus {}_s[o']^\varphi_\Box = \varnothing$ für alle $o, o' \in S$ mit $o \neq o'$. □

In den beiden folgenden Lemmata zeigen wir, dass sowohl im Fall von schleifenfreien Pfaden (Lemma 22) als auch bei Schleifen (Lemma 23) die Zulässigkeit von S_t in s_t bzgl. φ und bzgl. einer zulässigen Linearisierung $\prec_t$ für alle $t \in \{0, \ldots, b-1\}$ hinreichend dafür ist, dass die parallele Planausführung und die sequentielle Ausführung gemäß den Ordnungen $\prec_t$ äquivalent sind.

Lemma 22. *Sei $\mathcal{P} = \langle A, I, O, g\rangle$ eine Planungsinstanz, $b \in \mathbb{N}$, φ eine* LTL*-Formel und $\Pi = \langle S_0, \ldots, S_{b-1}\rangle$ ein 1-Linearisierungs-Plan der Länge b für $\mathcal{P}$ mit der Ausführung $\pi = s_0, \ldots, s_b$, und sei $\tilde{\pi}$ die Ausführung einer zulässigen Linearisierung $\tilde{\Pi}$ von Π. Gilt für alle $t \in \{0, \ldots, b-1\}$, dass S_t in s_t bezüglich φ und bezüglich der Ordnung $\prec_t$, in der die Operatoren aus S_t in $\tilde{\Pi}$ ausgeführt werden, zulässig ist, so ist $\pi \sim \tilde{\pi}$.*

Beweis. Nach Lemma 2 hat $\tilde{\pi}$ die Form $s_0, q_{0,1}, \ldots, q_{0,|S_0|-1}, s_1, q_{1,1}, \ldots, q_{1,|S_1|-1}, s_2, \ldots, s_{b-1}, q_{b-1,1}, \ldots, q_{b-1,|S_{b-1}|-1}, s_b$, wobei $s_0 = I$ und für jedes $i \in \{0, \ldots, b-1\}$ eine totale Ordnung $o_{i,1} \prec \ldots \prec o_{i,|S_i|}$ von S_i existiert, so dass gilt: $q_{i,1} = app_{o_{i,1}}(s_i)$, $q_{i,j+1} = app_{o_{i,j+1}}(q_{i,j})$ für alle $j \in \{1, \ldots, |S_i| - 2\}$ und $s_{i+1} = app_{o_{i,|S_i|}}(q_{i,|S_i|-1})$.

Wir zeigen, dass $\pi \sim \tilde{\pi}$ mit den Blockgrenzen $0 = i_0 < i_1 < i_2 < \cdots < i_b$ für π und $0 = j_0 < j_1 < j_2 < \cdots < j_b$ für $\tilde{\pi}$, wobei $i_t = t$ für alle $t \in \{0, \ldots, b\}$ sowie $j_0 = 0$, $j_1 = 1$ und $j_t = \sum_{k=0}^{t-2} |S_k| + 1$ für alle $t \in \{2, \ldots, b\}$. Wie die folgende Abbildung zeigt, besteht Block t in π nur aus dem Zustand s_t, während Block t in $\tilde{\pi}$ aus s_t sowie für $t \neq 0$ allen impliziten Zuständen zwischen s_{t-1} und s_t, d. h. $q_{t-1,1}, q_{t-1,2}, \ldots, q_{t-1,|S_{t-1}|-1}$, besteht.

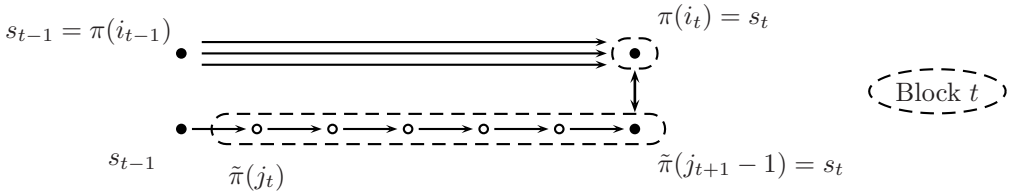


Abbildung 3.1.: Garantierte Äquivalenz am Beispiel eines Zeitschrittes

Mit mehreren Zeitpunkten ergibt sich ein Bild wie das folgende:

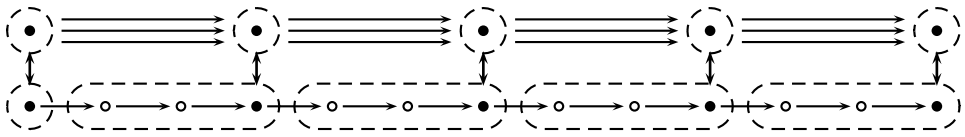


Abbildung 3.2.: Garantierte Äquivalenz bei einem endlichen Pfad ohne Schleife

Es reicht zu zeigen, dass für alle $t \in \{0, \ldots, b-1\}$ gilt:

$$L_\varphi(\pi(i_t)) = L_\varphi(\tilde{\pi}(j_t)) = L_\varphi(\tilde{\pi}(j_t + 1)) = \ldots = L_\varphi(\tilde{\pi}(j_{t+1} - 1)).$$

Dass $L_\varphi(\pi(i_t)) = L_\varphi(\tilde{\pi}(j_{t+1} - 1))$ folgt daraus, dass $\pi(i_0) = s_0 = s_{1-1} = \tilde{\pi}(j_1 - 1)$ und $\pi(i_t) = s_t = \tilde{\pi}(\sum_{k=0}^{t-1} |S_k|) = \tilde{\pi}(\sum_{k=0}^{t+1-2} |S_k| + 1 - 1) = \tilde{\pi}(j_{t+1} - 1)$ für $t \in \{1, \ldots, b\}$.

Es ist noch zu zeigen, dass $L_\varphi(\tilde{\pi}(j_t)) = L_\varphi(\tilde{\pi}(j_t+1)) = \ldots = L_\varphi(\tilde{\pi}(j_{t+1}-1))$.

Angenommen, diese Gleichheit gilt nicht. Dann gilt nicht für alle $r \in \{0, \ldots, |S_t|-1\}$, dass $L_\varphi(\tilde{\pi}(j_t)) = L_\varphi(\tilde{\pi}(j_t+r))$. Sei r minimal mit $L_\varphi(\tilde{\pi}(j_t)) \neq L_\varphi(\tilde{\pi}(j_t+r))$. Dann gibt es ein

$$\ell \in Lit(var(\varphi)) \tag{3.1}$$

mit $\tilde{\pi}(j_t+r') \not\models \ell$ für alle $r' \in \{0, \ldots, r-1\}$, insbesondere für $r'=0$:

$$\tilde{\pi}(j_t) \not\models \ell, \tag{3.2}$$

aber $\tilde{\pi}(j_t+r) \models \ell$. Angenommen, $\ell \in l(s_{t-1})$, d. h. $s_{t-1} \models \ell$. Da s_{t-1} der direkte Vorgängerzustand von $\tilde{\pi}(j_t)$ ist und wegen $\tilde{\pi}(j_t) \not\models \ell$ und $\tilde{\pi}(j_t+r) \models \ell$ gäbe es zwei dann Operatoren in S_{t-1} mit inkonsistenten Effekten, was bei 1-Linearisierungs-Plänen nicht möglich ist. Also gilt

$$\ell \notin l(s_{t-1}). \tag{3.3}$$

Der Operator, der in $\tilde{\pi}(j_t+r-1)$ angewandt wird und ℓ zum ersten Mal wahr macht, ist $o_{t-1,r+1} =: o$. Damit ist

$$\ell \in [o]_\Diamond. \tag{3.4}$$

Nach Definition ist

$${}_{s_{t-1}}[o]^\varphi_\Diamond = [o]^\varphi_\Diamond \setminus l(s_{t-1}) = ([o]_\Diamond \cap Lit(var(\varphi))) \setminus l(s_{t-1}). \tag{3.5}$$

Mit (3.1), (3.3) und (3.4) folgt daraus

$$\ell \in {}_{s_{t-1}}[o]^\varphi_\Diamond. \tag{3.6}$$

Da nach Voraussetzung S_{t-1} in s_{t-1} zulässig ist bezüglich φ und bezüglich der Ordnung $\prec_{t-1}$, in der die Operatoren aus S_{t-1} in $\tilde{\Pi}$ ausgeführt werden, d. h. insbesondere ${}_{s_{t-1}}[o]^\varphi_\Diamond \subseteq {}_{s_{t-1}}[o_{t-1,1}]^\varphi_\Box$ gilt, ist $\ell \in {}_{s_{t-1}}[o_{t-1,1}]^\varphi_\Box$. Da $o_{t-1,1}$ unmittelbar in den Zustand $\tilde{\pi}(j_t)$ führt, gilt dann

$$\tilde{\pi}(j_t) \models \ell, \tag{3.7}$$

im Widerspruch zu (3.2). □

Lemma 23. *Sei $\mathcal{P} = \langle A, I, O, g\rangle$ eine Planungsinstanz, $b \in \mathbb{N}$, φ eine* LTL*-Formel und $\Pi = \langle S_0, \ldots, S_{b-1}\rangle$ ein 1-Linearisierungs-Plan der Länge b für $\mathcal{P}$ mit der Ausführung $\pi = u \circ v$, $\|u\| = k$, die eine (b,k)-Schleife ist, und sei $\tilde{\pi}$ die Ausführung einer zulässigen Linearisierung von Π. Gilt für alle $t \in \{0, \ldots, b-1\}$ und alle $o, o' \in S_t$, dass S_t in s_t bezüglich φ und bezüglich der Ordnung $\prec_t$, in der die Operatoren aus S_t in $\tilde{\Pi}$ ausgeführt werden, zulässig ist, so ist $\tilde{\pi}$ eine $(\tilde{b}, \tilde{k})$-Schleife der Form $\tilde{u} \circ \tilde{v}$ mit solchen $\tilde{u}$ und $\tilde{v}$, dass $u \sim \tilde{u}$ und $v \sim \tilde{v}$.*

Beweis. Nach Lemma 2 hat jede Ausführung einer zulässigen Linearisierung von π die Form

$$\tilde{\pi} = \pi(0), q_{0,1}, \ldots, q_{0,|S_0|-1}, \pi(1), \ldots, \pi(b-1), q_{b-1,1}, \ldots, q_{b-1,|S_{b-1}|-1}, \pi(b),$$

wobei $\pi(b) = \pi(k-1)$. Setze dann

$$\begin{aligned} \tilde{u} &:= \pi(0), q_{0,1}, \ldots, q_{0,|S_0|-1}, \pi(1), q_{1,1}, \ldots, q_{1,|S_1|-1} \ldots, \pi(k-1) \qquad \text{und} \\ \tilde{v} &:= q_{k-1,1}, \ldots, q_{k-1,|S_{k-1}|-1}, \pi(k), q_{k,1}, \ldots, q_{k,|S_k|-1}, \pi(k+1), \ldots, \pi(b). \end{aligned}$$

Dass $\tilde{\pi}$ eine $(\tilde{b}, \tilde{k})$-Schleife der Form $\tilde{u} \circ \tilde{v}$ ist, folgt nun mit $\tilde{b} = |\tilde{\pi}| - 1 = \sum_{t=0}^{b-1} |S_t|$ und $\tilde{k} = |\tilde{u}| = \sum_{t=0}^{k-2} |S_t| + 1$.

Es ist noch zu zeigen, dass $u \sim \tilde{u}$ und $v \sim \tilde{v}$. Es gilt $u \sim \tilde{u}$ mit den Blockgrenzen $0 = i_0 < i_1 < i_2 < \cdots < i_{k-1}$ für u und $0 = j_0 < j_1 < j_2 < \cdots < j_{k-1}$ für $\tilde{u}$, wobei $i_t = t$ sowie $j_0 = 0$, $j_1 = 1$ und $j_t = \sum_{k=0}^{t-2} |S_k| + 1$ für alle $t \in \{ 2, \ldots, k-1 \}$. Dazu reicht es zu zeigen, dass für alle $t \in \{ 0, \ldots, k-1 \}$ gilt:

$$L_\varphi(u(i_t)) = L_\varphi(\tilde{u}(j_t)) = L_\varphi(\tilde{u}(j_t + 1)) = \ldots = L_\varphi(\tilde{u}(j_{t+1} - 1)).$$

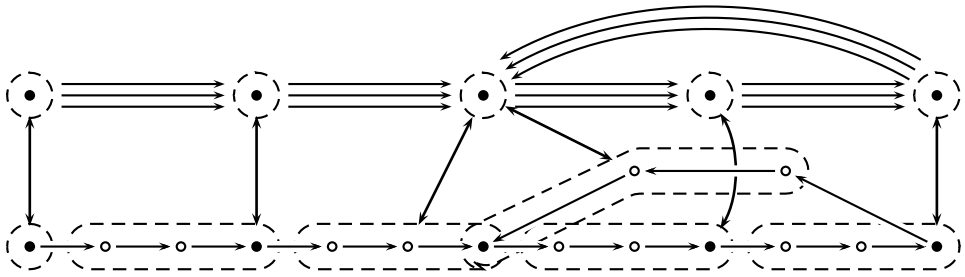


Abbildung 3.3.: Garantierte Äquivalenz bei einem Pfad mit Schleife

Der weitere Beweis erfolgt analog zum Beweis von Lemma 22. Dasselbe gilt für den Nachweis von $v \sim \tilde{v}$. □

3.4. Kodierungen

In diesem Abschnitt geben wir aussagenlogische Kodierungen der in Abschnitt 3.3 entwickelten Bedingungen an parallel angewandte Operatoren an, untersuchen die Größen dieser Kodierungen und zeigen, dass die Bedingungen für Äquivalenz erfüllt sind, wenn die einem parallelen Plan entsprechende Variablenbelegung die zugehörige Kodierung erfüllt.

Zunächst geben wir einige Abkürzungen für aussagenlogische Formeln an, die wir später benötigen werden.

Definition 46. Sei $k \in \mathbb{N}$, $X = \{ x^1, \ldots, x^k \}$ eine Menge von Aussagenvariablen und $C = \{ c^1, \ldots, c^k \}$ eine zu X disjunkte Menge von bisher unbenutzten Hilfsvariablen. Definiere dann

$$\mathit{chain}(x^1, \ldots, x^k) := \bigwedge_{j=1}^{k-1} (x^j \rightarrow \neg c^{j+1}) \wedge \bigwedge_{j=2}^{k-1} (\neg c^j \rightarrow \neg c^{j+1}) \wedge \bigwedge_{j=2}^{k} (\neg c^j \rightarrow \neg x^j).$$

Lemma 24. *Die Größe von $\mathit{chain}(x^1, \ldots, x^k)$ ist linear in k.* □

Definition 47. Sei $k \in \mathbb{N}$ und sei $O = \{ o^1, \ldots, o^k \}$ eine Menge von aussagenlogischen Variablen. Definiere dann

$$\mathit{onlyOneOp}(o^1, \ldots, o^k) := (\mathit{chain}(o^1, \ldots, o^k)) \wedge (\mathit{chain}(o^k, \ldots, o^1)).$$

Ist $O = \{ o^1, \ldots, o^n \}$ eine Menge von Operatoren, so schreiben wir kurz $\mathit{onlyOneOp}(O)$ für $\mathit{onlyOneOp}(o^1, \ldots, o^n)$, da die Formeln $\mathit{onlyOneOp}(\cdot)$ für alle Reihenfolgen der Operatoren logisch äquivalent sind.

Lemma 25. *Die Größe von $\mathit{onlyOneOp}(o^1, \ldots, o^k)$ ist linear in k.* □

Definition 48. Seien $k, m \in \mathbb{N}$ und sei $O = \{ o^1, \ldots, o^k, o^{k+1}, \ldots, o^{k+m} \}$ eine Menge von Aussagenvariablen. Definiere dann

$$\mathit{noConflict}(o^1, \ldots, o^{k+m}; k) := \left(\bigvee_{i=1}^{k} o^i \rightarrow \bigwedge_{j=k+1}^{k+m} \neg o^j \right) \wedge \mathit{onlyOneOp}(o^1, \ldots, o^k).$$

Ist φ eine LTL-Formel und $O = \{ o^1, \ldots, o^n \}$ eine Menge von Operatoren, so sei

$$\mathit{noConflict}(\varphi, O) := \mathit{noConflict}(o^1, \ldots, o^k, o^{k+1}, \ldots, o^{k+m}; k),$$

wobei $k = |O(\varphi)|$, $o^1, \ldots, o^k$ eine beliebige Anordnung von $O(\varphi)$ und $o^{k+1}, \ldots, o^{k+m}$ eine beliebige Anordnung von $O \setminus O(\varphi)$ ist.

Lemma 26. *Die Größe von $\mathit{noConflict}(o^1, \ldots, o^{k+m}; k)$ ist linear in $k + m$.*

Beweis. Für die Teilformel $\bigvee_{i=1}^{k} o_t^i \rightarrow \bigwedge_{j=k+1}^{k+m} \neg o_t^j$ gilt die Behauptung, da jede der $k + m$ Operatorvariablen in ihr genau einmal vorkommt. Da auch $\mathit{chain}(x^1, \ldots, x^k)$ linear groß in k und damit in $k + m$ ist, folgt die Behauptung. □

3.4.1. Direkte Kodierungen

Um einen gemeinsamen Maßstab für die Güte der folgenden Kodierungen zu erhalten, betrachten wir auch eine Kodierung, die fordert, dass zu jedem Zeitpunkt nur ein Operator angewandt wird, und die somit sequentielle Pläne erzwingt.

Definition 49. Sei φ eine LTL-Formel, $b \in \mathbb{N}$ und $\mathcal{P} = \langle A, I, O, g \rangle$ eine Planungsinstanz. Definiere dann

$$[\![\mathcal{P}]\!]^b_{\varphi,0} := \bigwedge_{t=0}^{b-1} \mathit{onlyOneOp}(O)_t.$$

Lemma 27. *Die Größe von $[\![\mathcal{P}]\!]^b_{\varphi,0}$ ist linear in der Anzahl der Operatoren.* □

Die folgenden beiden Kodierungen für paralleles Planen garantieren, dass ein Operator, der den Wert mindestens einer Variablen aus φ ändern kann, in einem Zeitschritt nur allein angewandt werden darf. Die Kodierungen unterscheiden sich lediglich in ihrer Größe, nicht in ihrer Bedeutung.

Definition 50. Sei φ eine LTL-Formel, $b \in \mathbb{N}$ und $\mathcal{P} = \langle A, I, O, g \rangle$ eine Planungsinstanz. Definiere dann

$$[\![\mathcal{P}]\!]^b_{\varphi,1} := \bigwedge_{t=0}^{b-1} \bigwedge_{o \in O(\varphi)} \left(o_t \rightarrow \bigwedge_{o' \in O \setminus \{ o \}} \neg o'_t \right).$$

Lemma 28. *Die Größe von $[\![\mathcal{P}]\!]^b_{\varphi,1}$ ist quadratisch in der Anzahl der Operatoren.* □

Lemma 29. *Sei $\mathcal{P} = \langle A, I, O, g\rangle$ eine Planungsinstanz, φ eine* LTL*-Formel, $b \in \mathbb{N}$ und $\Pi = \langle S_0, \ldots, S_{b-1}\rangle$ ein 1-Linearisierungs-Plan der Länge b für $\mathcal{P}$ mit der Ausführung $\pi = s_0, \ldots, s_b$, wobei $v^b_{\Pi,\pi} \models [\![\mathcal{P}]\!]^b_{\varphi,1}$. Dann gilt für alle $t \in \{0, \ldots, b-1\}$: Ist $S_t \cap O(\varphi) \neq \varnothing$, so ist $|S_t| = 1$.*

Beweis. Angenommen, $S_t \cap O(\varphi) \neq \varnothing$. Dann gilt $v^b_{\Pi,\pi} \models o_t$ für mindestens ein $o \in O(\varphi)$. Da nach Voraussetzung $v^b_{\Pi,\pi} \models o_t \rightarrow \bigwedge_{o' \in O \setminus \{o\}} \neg o'_t$, folgt $v^b_{\Pi,\pi} \models \neg o'_t$ für alle $o' \in O \setminus \{o\}$. Damit ist $S_t = S_t^{v^b_{\Pi,\pi}} = \{o\}$, d. h. $|S_t| = 1$. □

Beachte, dass diese Kodierung für einen Zeitschritt quadratische Größe in der Anzahl der Operatoren hat. Eine lineare Größe kann man erzielen, indem man die folgende Kodierung verwendet.

Definition 51. Sei φ eine LTL-Formel, $\mathcal{P} = \langle A, I, O, g\rangle$ eine Planungsinstanz und etwa $O = \{o^1, \ldots, o^k, o^{k+1}, \ldots, o^{k+m}\}$, wobei $O(\varphi) = \{o^1, \ldots, o^k\}$. Definiere dann

$$[\![\mathcal{P}]\!]^b_{\varphi,2} := \bigwedge_{t=0}^{b-1} \mathit{noConflict}(\varphi, O)_t\,.$$

Lemma 30. *Die Größe von $[\![\mathcal{P}]\!]^b_{\varphi,2}$ ist linear in der Anzahl der Operatoren.* □

Lemma 31. *Sei $\mathcal{P} = \langle A, I, O, g\rangle$ eine Planungsinstanz, φ eine* LTL*-Formel, $b \in \mathbb{N}$ und $\Pi = \langle S_0, \ldots, S_{b-1}\rangle$ ein 1-Linearisierungs-Plan der Länge b für $\mathcal{P}$ mit der Ausführung $\pi = s_0, \ldots, s_b$, wobei $v^b_{\Pi,\pi} \models [\![\mathcal{P}]\!]^b_{\varphi,2}$. Dann gilt für alle $t \in \{0, \ldots, b-1\}$: Ist $S_t \cap O(\varphi) \neq \varnothing$, so ist $|S_t| = 1$.*

Beweis. Angenommen $o^i \in S_t \cap O(\varphi)$. Wir zeigen, dass $S_t = \{o^i\}$. Dazu reicht es, für alle anderen Operatoren o^j mit $j \neq i$ zu zeigen, dass $o^j \notin S_t$. Wir unterscheiden zwei Fälle.

Betrachte zuerst den Fall, dass $o^j \notin O(\varphi)$, d. h., $k+1 \leq j \leq k+m$. Wegen $v^b_{\Pi,\pi} \models \bigvee_{i=1}^k o^i_t \rightarrow \bigwedge_{j'=k+1}^{k+m} \neg o^{j'}_t$ und da nach Definition von $v^b_{\Pi,\pi}$ und wegen $o^i \in S_t$ auch $v^b_{\Pi,\pi} \models o^i_t$, folgt, dass $v^b_{\Pi,\pi} \models \bigwedge_{j'=k+1}^{k+m} \neg o^{j'}_t$, insbesondere also $v^b_{\Pi,\pi} \models \neg o^j_t$, da $k+1 \leq j \leq k+m$. Nach der Definition von $v^b_{\Pi,\pi}$ folgt, dass $o^j \notin S_t$.

Betrachte nun den Fall, dass $o^j \in O(\varphi)$, d. h. $1 \leq j \leq k$, und ohne Beschränkung der Allgemeinheit, dass $j > i$. Falls $j < i$ gilt die folgende Argumentation mit vertauschten i und j. Da $v^b_{\Pi,\pi} \models \mathit{chain}(o^1, \ldots, o^k)_t$, gilt insbesondere $v^b_{\Pi,\pi} \models (o^i \rightarrow \neg c^{i+1})_t \wedge \bigwedge_{\ell=i+1}^{j-1}(\neg c^i \rightarrow \neg c^{i+1})_t \wedge (\neg c^j \rightarrow \neg o^j)_t$. Mit $v^b_{\Pi,\pi} \models o^i_t$ folgt $v^b_{\Pi,\pi} \models \neg c^{i+1}_t$, daraus induktiv für alle $n \in \{i+2, \ldots, j\}$, dass $v^b_{\Pi,\pi} \models \neg c^n_t$, und schließlich $v^b_{\Pi,\pi} \models \neg o^j_t$. Also folgt auch in diesem Fall, dass $o^j \notin S_t$. □

Die dritte Übersetzung kodiert direkt die Forderung, dass sich parallel angewandte Operatoren bezüglich φ gleich verhalten sollen.

Definition 52. Sei φ eine LTL-Formel, $b \in \mathbb{N}$ und $\mathcal{P} = \langle A, I, O, g\rangle$ eine Planungsinstanz. Definiere dann

$$[\![\mathcal{P}]\!]^b_{\varphi,3} := \bigwedge_{t=0}^{b-1} \bigwedge_{\substack{o^i, o^j \in O, \\ o^i, o^j \text{ verhalten sich} \\ \text{nicht gleich bzgl. } \varphi}} \neg(o^i_t \wedge o^j_t).$$

Lemma 32. *Für je zwei Operatoren $o, o' \in O$ und eine LTL-Formel φ kann in polynomieller Zeit entschieden werden, ob sich o und o' bzgl. φ gleich verhalten.* □

Lemma 33. *Die Größe von $[\![\mathcal{P}]\!]^b_{\varphi,3}$ ist quadratisch in der Anzahl der Operatoren.* □

Lemma 34. *Sei $\mathcal{P} = \langle A, I, O, g\rangle$ eine Planungsinstanz, φ eine LTL-Formel, $b \in \mathbb{N}$ und $\Pi = \langle S_0, \ldots, S_{b-1}\rangle$ ein 1-Linearisierungs-Plan der Länge b für $\mathcal{P}$ mit der Ausführung $\pi = s_0, \ldots, s_b$, wobei $v^b_{\Pi,\pi} \models [\![\mathcal{P}]\!]^b_{\varphi,3}$. Dann gilt für alle $t \in \{0, \ldots, b-1\}$: Sind $o, o' \in S_t$, so verhalten sich o und o' gleich bzgl. φ.* □

Die folgenden Kodierungen sorgen dafür, dass es bei fest vorgegebener Reihenfolge der Operatoren nicht vorkommen kann, dass zu einem Zeitpunkt ein Operator angewandt wird, der eine Variable möglicherweise im Effekt hat, nachdem bereits ein Operator angewandt wurde, der dieselbe Variable *nicht* garantiert im Effekt hat.

Definition 53. Sei $O = \{ o^1, \ldots, o^n \}$ eine Menge von Operatoren, ℓ ein Literal und $X = \{ c^{i,\ell} \mid i \in \{1, \ldots, n\} \}$ eine Menge von bisher unbenutzten Hilfsvariablen. Definiere dann

$$\begin{aligned} \mathit{ltlchain}(o^1, \ldots, o^n; \ell) := & \bigwedge \{ o^i \to c^{i-1,\ell} \mid \overline{\ell} \in [o^i]_\Diamond, i \in \{2, \ldots, n\} \} \wedge \\ & \bigwedge \{ c^{i,\ell} \to c^{i-1,\ell} \mid i \in \{2, \ldots, n\} \} \wedge \\ & \bigwedge \{ c^{i,\ell} \to \neg o^i \mid \overline{\ell} \notin [o^i]_\Box, i \in \{1, \ldots, n\} \}. \end{aligned}$$

Definition 54. Sei $\mathcal{P} = \langle A, I, O, g\rangle$ eine Planungsinstanz, $b \in \mathbb{N}$, φ eine LTL-Formel und $\prec$ eine mit einem Deaktivierungsgraphen für $\mathcal{P}$ verträgliche totale Ordnung von O, so dass $o^1 \prec \cdots \prec o^n$. Definiere dann

$$[\![\mathcal{P}]\!]^b_{\varphi,4} := \bigwedge_{t=0}^{b-1} \bigwedge_{\ell \in Lit(var(\varphi))} (\ell \to \mathit{ltlchain}(o^1, \ldots, o^n; \ell))_t .$$

Lemma 35. *Die Formel $[\![\mathcal{P}]\!]^b_{\varphi,4}$ hat lineare Größe in der Anzahl der Operatoren.* □

Lemma 36. *Sei $\mathcal{P} = \langle A, I, O, g\rangle$ eine Planungsinstanz, φ eine LTL-Formel, $b \in \mathbb{N}$ und $\Pi = \langle S_0, \ldots, S_{b-1}\rangle$ ein 1-Linearisierungs-Plan der Länge b für $\mathcal{P}$ mit der Ausführung $\pi = s_0, \ldots, s_b$, wobei $v^b_{\Pi,\pi} \models [\![\mathcal{P}]\!]^b_{\varphi,4}$. Dann ist S_t in s_t bezüglich der Formel φ und der in $\mathit{ltlchain}(o^1, \ldots, o^n; \ell)$ gewählten Ordnung $\prec$ zulässig für alle $t \in \{0, \ldots, b-1\}$.*

Beweis. Sei $t \in \{0, \ldots, b-1\}$ und seien $o^i, o^j \in S_t$ mit $o^i \prec o^j$. Dann ist zu zeigen, dass ${}_{s_t}[o^j]^\varphi_\Diamond \subseteq {}_{s_t}[o^i]^\varphi_\Box$. Sei dazu das Literal $\ell \in {}_{s_t}[o^j]^\varphi_\Diamond$. Dann ist $\ell \in [o^j]_\Diamond$, $\overline{\ell} \in l(s_t)$ und $\ell, \overline{\ell} \in Lit(var(\varphi))$. Wegen $o^j \in S_t$ gilt $v^b_{\Pi,\pi} \models o^j_t$. Wegen $v^b_{\Pi,\pi} \models [\![\mathcal{P}]\!]^b_{\varphi,4}$ und $\ell, \overline{\ell} \in Lit(var(\varphi))$ gilt $v^b_{\Pi,\pi} \models \overline{\ell}_t \to \mathit{ltlchain}(o^1, \ldots, o^n; \overline{\ell})_t$, wegen $\overline{\ell} \in l(s_t)$ gilt $v^b_{\Pi,\pi} \models \overline{\ell}_t$, mit Modus Ponens

also $v^b_{\Pi,\pi} \models \mathit{ltlchain}(o^1, \ldots, o^n; \overline{\ell})_t$. Damit gilt wegen $\overline{\overline{\ell}} = \ell \in [o^j]_\Diamond$ auch $v^b_{\Pi,\pi} \models o^j_t \rightarrow c^{j-1,\ell}_t$. Mit $v^b_{\Pi,\pi} \models o^j_t$ folgt $v^b_{\Pi,\pi} \models c^{j-1,\ell}_t$. Induktiv kann man zeigen, dass $v^b_{\Pi,\pi} \models c^{i',\ell}_t$ für alle $i' < j$ gilt, insbesondere $v^b_{\Pi,\pi} \models c^{i,\ell}_t$. Wäre $\overline{\overline{\ell}} = \ell \notin [o^i]_\Box$, so gälte $v^b_{\Pi,\pi} \models c^{i,\ell}_t \rightarrow \neg o^i_t$, mit Modus Ponens also $v^b_{\Pi,\pi} \models \neg o^i_t$ im Widerspruch zu $v^b_{\Pi,\pi} \models o^i_t$. Also muss $\ell \in [o^i]_\Box$ sei. Mit $\overline{\ell} \in l(s_t)$ und $\ell, \overline{\ell} \in \mathit{Lit}(\mathit{var}(\varphi))$ folgt daraus, dass $\ell \in {}_{s_t}[o^i]^\varphi_\Box$. □

Beachte, dass man in $[\![\mathcal{P}]\!]^b_{\varphi,4}$ jedes Vorkommen von $\mathit{ltlchain}(o^1, \ldots, o^n; \ell)$ durch die Formel $\mathit{linchain}(o^1, \ldots, o^n; E^\sim_\ell; R^\sim_\ell; \ell)$ mit $E^\sim_\ell$ und $R^\sim_\ell$ wie in Definition 56 ersetzen kann und dadurch eine im Allgemeinen kleinere Formel erhält, für die Lemma 36 weiter gilt (ohne Beweis).

3.4.2. Erweiterung des Deaktivierungsgraphen

Bei den bisherigen Kodierungen haben wir lediglich ein weiteres Konjunktionsglied zu der aus Basiskodierung, Kodierung der Parallelitätsaxiome und Kodierung der $\mathsf{LTL}_{-\mathbf{X}}$-Spezifikation bestehenden Übersetzung hinzugefügt. Nun wollen wir unsere Übersetzung enger mit der Kodierung der Parallelitätsaxiome verbinden, die garantieren, dass kein Operator einen späteren parallelen Operator deaktiviert. Dazu fügen wir in den in Definition 19 eingeführten Deaktivierungsgraphen zusätzliche Kanten ein. Bislang muss der Deaktivierungsgraph nur dann eine Kante zwischen Operatoren o und o' besitzen, wenn es ein Literal ℓ über einer Zustandsvariable a gibt, das ein bedingter oder unbedingter Effekt von o ist und zugleich negativ in der Vorbedingung von o' oder im Antezedens eines bedingten Effekts von o' vorkommt, d. h. wenn die Anwendung von o dazu führen könnte, dass o' danach nicht mehr anwendbar wäre oder die nachfolgende Anwendung von o' andere aktive Effekte als im letzten explizit gemachten Zustand hätte.

Da wir, um Äquivalenz zwischen π und $\tilde{\pi}$ zu garantieren, fordern, dass alle für die Spezifikation relevanten Zustandsänderungen bereits vom ersten zu einem Zeitpunkt ausgeführten Operator vorgenommen werden, erscheint es sinnvoll, zu definieren, dass ein Operator o einen anderen Operator o' deaktiviert, wenn o' ein Literal ℓ, das positiv oder negativ in der Spezifikation vorkommt, als bedingten oder unbedingten Effekt besitzt, das nicht auch bereits in o als *un*bedingter Effekt auftritt. Beachte, dass wir den Zustand, in dem o und o' möglicherweise parallel angewandt werden, außer Acht lassen, was dazu führt, dass wir erstens auch solche Literale nicht ignorieren können, die von einem Operator wahr gemacht werden, aber in dem Zustand, in dem der Operator angewandt wird, bereits wahr sind, und dass zweitens bei dem späteren Operator *alle* Effekte betrachtet werden müssen, bei dem früheren Operator aber nur die *unbedingten* Effekte berücksichtigt werden dürfen.

Um den gewünschten erweiterten Deaktivierungsgraphen zu erhalten, genügt es, die Definition der Deaktivierung von Operatoren anzupassen:

Definition 55 (Deaktivierung, verallgemeinert). Sei A eine Menge von Zustandsvariablen, seien o und o' Operatoren über A und sei φ eine LTL-Formel mit $\mathit{var}(\varphi) \subseteq A$. Dann **deaktiviert der Operator o den Operator o'**, wenn $o \neq o'$ und

1. es ein solches Literal $\ell \in \mathit{Lit}(A)$ über den Zustandsvariablen gibt, dass
   a) $\ell \in [o]_\Diamond$ und
   b) i. $\mathrm{neg}(\ell, p')$ oder

ii. es ein $f' \rhd d' \in c'$ so gibt, dass $\ell \in \mathit{Lit}(\mathit{var}(f'))$.

oder

2. $[o']^{\varphi}_{\Diamond} \setminus [o]^{\varphi}_{\Box} \neq \varnothing$.

Ein **erweiterter Deaktivierungsgraph** für $\mathcal{P}$ und φ ist dann wie in Definition 19 definiert, wobei jedoch nun die neue Definition von Deaktivierung zugrunde gelegt wird.

Definition 56. Ist $\mathcal{P} = \langle A, I, O, g\rangle$, so definieren wir wie schon in Def. 23 für jedes Literal $\ell \in \mathit{Lit}(A)$ die Mengen E_ℓ und R_ℓ. Ist φ die LTL-Spezifikation, so definieren wir nun zusätzlich für jedes Literal $\ell \in \mathit{Lit}(\mathit{var}(\varphi))$ die Mengen

$$E^{\sim}_{\ell} := \{\, o \in O \mid \ell \notin [o]_{\Box} \,\} \qquad \text{und} \qquad R^{\sim}_{\ell} := \{\, o \in O \mid \ell \in [o]_{\Diamond} \,\}.$$

Damit können wir auf der Grundlage des erweiterten Deaktivierungsgraphen $G = \langle O, K\rangle$ mit den starken Zusammenhangskomponenten $C_1, \ldots, C_m$ und $C_i = \{\, o_{i_1}, \ldots, o_{i_{|C_i|}} \,\}$ für $i \in \{\, 1, \ldots, m \,\}$ die folgende aussagenlogische Kodierung definieren. Beachte, dass in dieser Kodierung nicht nur ein weiteres Konjunktionsglied zur Basiskodierung und der Kodierung der „alten“ Parallelitätsaxiome hinzukommt, sondern die „alten“ Parallelitätsaxiome selbst sich ändern, da der Deaktivierungsgraph, aus dessen Struktur die Parallelitätsaxiome gewonnen werden, zusätzliche Kanten bekommt.

Definition 57 (Verallgemeinerung von Def. 26). Sei $\mathcal{P} = \langle A, I, O, g\rangle$ eine Planungsinstanz und $G = \langle O, K\rangle$ ein erweiterter Deaktivierungsgraph für $\mathcal{P}$. Seien $C_1, \ldots, C_m$ die starken Zusammenhangskomponenten von G und sei etwa $C_i = \{\, o_{i_1}, \ldots, o_{i_{|C_i|}} \,\}$ für $i \in \{\, 1, \ldots, m \,\}$, wobei $o_{i_1} \prec_i \cdots \prec_i o_{i_{|C_i|}}$ eine beliebige totale Ordnung auf C_i sei. Definiere dann

$$[\![\mathcal{P}]\!]^{b,2}_{1lin} := \bigwedge_{t=0}^{b-1} \bigwedge_{i=1}^{m} \Bigg(\bigwedge_{\ell \in \mathit{Lit}(A)} \mathit{linchain}(o_{i_1}, \ldots, o_{i_{|C_i|}}; E_\ell; R_\ell; \ell)_t \quad \wedge \quad \bigwedge_{\ell \in \mathit{Lit}(\mathit{var}(\varphi))} \mathit{linchain}(o_{i_1}, \ldots, o_{i_{|C_i|}}; E^{\sim}_{\ell}; R^{\sim}_{\ell}; \tilde{\ell})_t \Bigg).$$

Lemma 37. *Die Formel $[\![\mathcal{P}]\!]^{b,2}_{1lin}$ hat Größe $\mathcal{O}(n)$ für $n = |O|$.* □

Lemma 38. *Sei $\mathcal{P} = \langle A, I, O, g\rangle$ eine Planungsinstanz, φ eine* LTL*-Formel, $b \in \mathbb{N}$ und $\Pi = \langle S_0, \ldots, S_{b-1}\rangle$ ein 1-Linearisierungs-Plan der Länge b für $\mathcal{P}$ mit Ausführung $\pi = s_0, \ldots, s_b$, so dass $v^b_{\pi,\Pi} \models [\![\mathcal{P}]\!]^{b,2}_{1lin}$. Sei ferner $t \in \{\, 0, \ldots, b-1 \,\}$ und $G = \langle O, K\rangle$ der erweiterte Deaktivierungsgraph, der bei der Konstruktion der Übersetzung $[\![\mathcal{P}]\!]^{b,2}_{1lin}$ verwendet wurde. Sei dann $\prec^t$ eine mit G und den Ordnungen $\prec_i$ der starken Zusammenhangskomponenten C_i von G verträgliche totale Ordnung von O. Dann ist Π ein 1-Linearisierungs-Plan für $\mathcal{P}$, und S_t ist bzgl. φ und $\prec^t$ zulässig.*

Beweis. Dass Π ein 1-Linearisierungs-Plan für $\mathcal{P}$ ist, folgt aus Lemma 9. Für die Zulässigkeit von S_t bzgl. φ und $\prec^t$ reicht es zu zeigen, dass immer dann, wenn $o, o' \in S_t$ mit $o \prec^t o'$ gilt, auch $[o']^{\varphi}_{\Diamond} \subseteq [o]^{\varphi}_{\Box}$ bzw. $[o']^{\varphi}_{\Diamond} \setminus [o]^{\varphi}_{\Box} = \varnothing$ ist. Wir nehmen an, dass $o \prec^t o'$ und unterscheiden zwei Fälle:

1. Angenommen, o und o' gehören zur gleichen starken Zusammenhangskomponente C_i von G, also $o \prec_i o'$. Wir nehmen an, es gäbe ein $\ell \in \mathit{Lit}(\mathit{var}(\varphi))$ mit $\ell \in [o']^{\varphi}_{\Diamond}$ und $\ell \notin [o]^{\varphi}_{\Box}$ und führen dies zu einem Widerspruch. Nach dieser Annahme ist $o' \in R^{\sim}_{\ell}$ und $o \in E^{\sim}_{\ell}$. Wegen $o, o' \in S_t$ gilt $v^b_{\Pi,\pi} \models o_t$ und $v^b_{\Pi,\pi} \models o'_t$. Nach Voraussetzung gilt $v^b_{\Pi,\pi} \models \mathit{linchain}(o_{i_1}, \ldots, o_{i_{|C_i|}}; E^{\sim}_{\ell}; R^{\sim}_{\ell}; \tilde{\ell})_t$. Diese Formel enthält eine Kette von Implikationen $o_t \rightarrow a^{j_1,\ell}_t \rightarrow a^{j_2,\ell}_t \rightarrow \ldots \rightarrow a^{j_k,\ell}_t \rightarrow \neg o'_t$. Daraus folgt mit $v^b_{\Pi,\pi} \models o_t$, dass $v^b_{\Pi,\pi} \models \neg o'_t$, ein Widerspruch.
2. Angenommen, o und o' gehören zu unterschiedlichen starken Zusammenhangskomponenten von G, etwa $o \in C_i$ und $o' \in C_j$. Wir nehmen zum Widerspruch wie oben an, dass es ein $\ell \in \mathit{Lit}(\mathit{var}(\varphi))$ mit $\ell \in [o']^{\varphi}_{\Diamond}$ und $\ell \notin [o]^{\varphi}_{\Box}$ gäbe. Nach dieser Annahme ist $[o']^{\varphi}_{\Diamond} \setminus [o]^{\varphi}_{\Box} \neq \varnothing$, d. h. o deaktiviert o' und es gibt eine Kante $(o, o') \in K$. Nach Definition der totalen Ordnung auf dem erweiterten Deaktivierungsgraphen ist dann $C_j \prec_C C_i$ bzw. $o' \prec^t o$ im Widerspruch zur Annahme. □

Beispiel 58. Betrachte die Planungsaufgabe aus Abschnitt 2.6 und für diese Aufgabe einen möglichen Deaktivierungsgraphen eingeschränkt auf die Aktionen, die sich in Stadt $\mathtt{c_1}$ abspielen, d. h. $\mathtt{l_1}$, $\mathtt{l_2}$, $\mathtt{u_1}$, $\mathtt{u_2}$, $\mathtt{dt_1}$ und $\mathtt{dt_2}$. Die für die Berechnung dieses Teils des Graphen relevanten Literale sind $\mathtt{at(p, d_1)}$, $\neg\mathtt{at(p, d_1)}$, $\mathtt{at(p, d_2)}$ und $\neg\mathtt{at(p, d_2)}$, da wir die Spezifikation hier mit $\varphi = \mathbf{GF}\mathtt{at(p, d_1)} \wedge \mathbf{GF}\mathtt{at(p, d_2)}$ identifizieren können.

Die Aktionen $\mathtt{l_1}$, $\mathtt{l_2}$, $\mathtt{u_1}$, $\mathtt{u_2}$, $\mathtt{dt_1}$ und $\mathtt{dt_2}$ sind wie folgt definiert[1]:

$$
\begin{aligned}
\mathtt{l_1} &:= \langle\, \mathtt{at(t, d_1)} \wedge \mathtt{at(p, d_1)}, \{\, \neg\mathtt{at(p, d_1)}, \mathtt{in(p, t)} \,\}, \varnothing \,\rangle \\
\mathtt{l_2} &:= \langle\, \mathtt{at(t, d_2)} \wedge \mathtt{at(p, d_2)}, \{\, \neg\mathtt{at(p, d_2)}, \mathtt{in(p, t)} \,\}, \varnothing \,\rangle \\
\mathtt{u_1} &:= \langle\, \mathtt{at(t, d_1)} \wedge \mathtt{in(p, t)}, \{\, \neg\mathtt{in(p, t)}, \mathtt{at(p, d_1)} \,\}, \varnothing \,\rangle \\
\mathtt{u_2} &:= \langle\, \mathtt{at(t, d_2)} \wedge \mathtt{in(p, t)}, \{\, \neg\mathtt{in(p, t)}, \mathtt{at(p, d_2)} \,\}, \varnothing \,\rangle \\
\mathtt{dt_1} &:= \langle\, \mathtt{at(t, d_2)} \wedge \mathtt{inSameCity(d_2, d_1)}, \{\, \neg\mathtt{at(t, d_2)}, \mathtt{at(t, d_1)} \,\}, \varnothing \,\rangle \\
\mathtt{dt_2} &:= \langle\, \mathtt{at(t, d_1)} \wedge \mathtt{inSameCity(d_1, d_2)}, \{\, \neg\mathtt{at(t, d_1)}, \mathtt{at(t, d_2)} \,\}, \varnothing \,\rangle
\end{aligned}
$$

Der erweiterte Deaktivierungsgraph muss keine Kanten zwischen Aktionen enthalten, die in verschiedenen Depots ausgeführt werden, da solche Aktionen niemals gleichzeitig anwendbar sein können. Er muss ferner aus demselben Grund keine Kanten zwischen einer $\mathtt{loadTruck(p, t, d)}$- und einer $\mathtt{unloadTruck(p, t, d)}$-Aktion enthalten. Unter den Aktionen $\mathtt{l_1}$, $\mathtt{l_2}$, $\mathtt{u_1}$, $\mathtt{u_2}$, $\mathtt{dt_1}$ und $\mathtt{dt_2}$ sind also höchstens Kanten zwischen $\mathtt{dt}_i$ und $\mathtt{l}_{3-i}$ bzw. $\mathtt{u}_{3-i}$ für $i \in \{\, 1, 2 \,\}$ nötig. Da eine `loadTruck`- oder `unloadTruck`-Aktion weder die Vorbedingung einer `driveTruck`-Aktion falsifizieren noch die aktiven Effekte einer `driveTruck`-Aktion beeinflussen kann, sind auch Kanten in Richtung von `driveTruck`-Aktionen unnötig. Die noch nicht erwähnten Kanten von `driveTruck`-Aktionen zu `loadTruck`- oder `unloadTruck`-Aktionen müssen tatsächlich in jedem Deaktivierungsgraphen vorhanden sein, da eine `driveTruck`-Aktion, die von einem Depot wegführt, `loadTruck`- und `unloadTruck`-Aktionen in diesem Depot unmöglich macht.

Die oben genannten Kanten sind bereits im nicht-erweiterten Deaktivierungsgraphen enthalten. Die Definition des erweiterten Deaktivierungsgraphen fordert zur Gewährleistung

[1] im Unterschied zu der in Anhang A angegebenen STRIPS-Definition der LOGISTICS-Domäne (nach McDermott u. a. [1998]) lassen wir hier die Typprädikate in den Vorbedingungen weg und führen anstelle des Prädikates $\mathtt{inCity(d, c)}$ ein Prädikat $\mathtt{inSameCity(d_1, d_2)}$ ein.

der Äquivalenz folgende Kanten: Da die `driveTruck`-Aktionen keine Variablen bzw. Literale aus φ enthalten, können sie durch keine anderen vorangehenden Aktionen deaktiviert werden, d. h. es sind keine Kanten zu `driveTruck`-Aktionen nötig. Die von den `driveTruck`-Aktionen ausgehenden Kanten zu den `loadTruck`- und `unloadTruck`-Aktionen werden jedoch benötigt, da diese Aktionen Literale über φ in ihren Effekten haben, die nicht in den Effekten der `driveTruck`-Aktionen vorkommen, nämlich etwa $\mathtt{at}(\mathtt{p}, \mathtt{d}_1) \in [\mathtt{u}_1]^{\varphi}_{\Diamond}$, aber $\mathtt{at}(\mathtt{p}, \mathtt{d}_1) \notin [\mathtt{dt}_2]^{\varphi}_{\Box}$ (bzw. $\mathtt{u}_1 \in E^{\sim}_{\mathtt{at}(\mathtt{p},\mathtt{d}_1)}$ und $\mathtt{dt}_2 \in R^{\sim}_{\mathtt{at}(\mathtt{p},\mathtt{d}_1)}$), also $[\mathtt{u}_1]^{\varphi}_{\Diamond} \setminus [\mathtt{dt}_2]^{\varphi}_{\Box} \neq \varnothing$.

In diesem Beispiel stimmt der kleinste erweiterte Deaktivierungsgraph zufällig mit dem kleinsten nicht-erweiterten Deaktivierungsgraphen überein, während im Allgemeinen die Kantenmenge in einem erweiterten Deaktivierungsgraphen auch eine echte Obermenge der Kantenmenge in einem einfachen Deaktivierungsgraphen sein kann. In der folgenden Abbildung sind die Kanten, die bereits im nicht-erweiterten Deaktivierungsgraphen enthalten sind, durchgezogen dargestellt, Kanten, die durch die Erweiterung des Deaktivierungsgraphen hinzukommen bzw. hinzukommen würden, sind mit unterbrochenen Linien dargestellt.

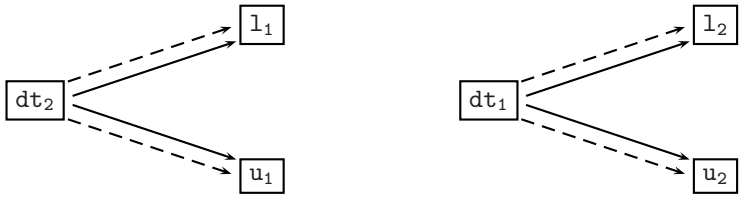


Abbildung 3.4.: Erweiterter Deaktivierungsgraph für LOGISTICS-Beispiel

Da der erweiterte Deaktivierungsgraph azyklisch ist und damit alle starken Zusammenhangskomponenten einelementig sind, müssen in diesem Beispiel keine Axiome kodiert werden, die dafür sorgen, dass Zyklen durch Auslassen mindestens eines Operators durchbrochen werden.

In dem in Abschnitt 2.6 angegebenen Plan Π werden 21 Aktionen zu 21 verschiedenen Zeitpunkten ausgeführt. Lässt man parallele Transitionen zu, so gibt es einen Plan mit denselben 21 Aktionen wie oben, jedoch nur 13 verschiedenen Zeitpunkten, nämlich

$$
\begin{aligned}
\Pi' \;=\; \langle\{\, & \mathtt{driveTruck}(\mathtt{t}_1, \mathtt{d}_{12}, \mathtt{d}_{11}), \mathtt{driveTruck}(\mathtt{t}_2, \mathtt{d}_{22}, \mathtt{d}_{21}), \mathtt{driveTruck}(\mathtt{t}_3, \mathtt{d}_{32}, \mathtt{d}_{31}) \,\}\\
& \{\, \mathtt{loadTruck}(\mathtt{p}_1, \mathtt{t}_1, \mathtt{d}_{11}), \mathtt{driveTruck}(\mathtt{t}_1, \mathtt{d}_{11}, \mathtt{d}_{12}) \,\},\\
& \{\, \mathtt{loadTruck}(\mathtt{p}_2, \mathtt{t}_2, \mathtt{d}_{21}), \mathtt{driveTruck}(\mathtt{t}_2, \mathtt{d}_{21}, \mathtt{d}_{22}) \,\},\\
& \{\, \mathtt{loadTruck}(\mathtt{p}_3, \mathtt{t}_3, \mathtt{d}_{31}), \mathtt{driveTruck}(\mathtt{t}_3, \mathtt{d}_{31}, \mathtt{d}_{32}) \,\},\\
& \{\, \mathtt{unloadTruck}(\mathtt{p}_1, \mathtt{t}_1, \mathtt{d}_{12}) \,\},\\
& \{\, \mathtt{unloadTruck}(\mathtt{p}_2, \mathtt{t}_2, \mathtt{d}_{22}) \,\},\\
& \{\, \mathtt{unloadTruck}(\mathtt{p}_3, \mathtt{t}_3, \mathtt{d}_{32}) \,\},\\
& \{\, \mathtt{loadTruck}(\mathtt{p}_1, \mathtt{t}_1, \mathtt{d}_{12}), \mathtt{driveTruck}(\mathtt{t}_1, \mathtt{d}_{12}, \mathtt{d}_{11}) \,\},\\
& \{\, \mathtt{loadTruck}(\mathtt{p}_2, \mathtt{t}_2, \mathtt{d}_{22}), \mathtt{driveTruck}(\mathtt{t}_2, \mathtt{d}_{22}, \mathtt{d}_{21}) \,\},\\
& \{\, \mathtt{loadTruck}(\mathtt{p}_3, \mathtt{t}_3, \mathtt{d}_{32}), \mathtt{driveTruck}(\mathtt{t}_3, \mathtt{d}_{32}, \mathtt{d}_{31}) \,\},\\
& \{\, \mathtt{unloadTruck}(\mathtt{p}_1, \mathtt{t}_1, \mathtt{d}_{11}) \,\},\\
& \{\, \mathtt{unloadTruck}(\mathtt{p}_2, \mathtt{t}_2, \mathtt{d}_{21}) \,\},\\
& \{\, \mathtt{unloadTruck}(\mathtt{p}_3, \mathtt{t}_3, \mathtt{d}_{31}) \,\}\rangle.
\end{aligned}
$$

Die graphische Darstellung des Planes bzw. seiner Ausführung ähnelt der in Abschnitt 2.6 für den sequentiellen Fall, zeigt aber nun, zu welchen Zeitpunkten Aktionen parallel ausgeführt werden.

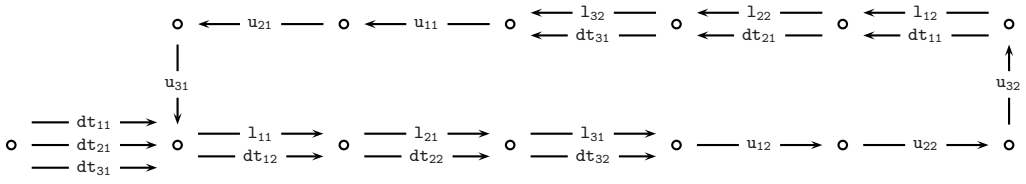


Abbildung 3.5.: Parallele Planausführung im LOGISTICS-Beispiel

Beachte, dass ein solcher Plan im Allgemeinen mehr Operatoren als notwendig enthält. So ist beispielsweise auch ein Plan der Länge 13 möglich, bei dem die Mengen S_0 sowie S_4 bis S_{13} wie oben sind, jedoch

$$\begin{aligned} S_1 &= \{\, \texttt{loadTruck}(\texttt{p}_1, \texttt{t}_1, \texttt{d}_{11}) \,\}, \\ S_2 &= \{\, \texttt{loadTruck}(\texttt{p}_2, \texttt{t}_2, \texttt{d}_{21}) \,\}, \\ S_3 &= \{\, \texttt{loadTruck}(\texttt{p}_3, \texttt{t}_3, \texttt{d}_{31}), \texttt{driveTruck}(\texttt{t}_1, \texttt{d}_{11}, \texttt{d}_{12}), \\ &\qquad \texttt{driveTruck}(\texttt{t}_2, \texttt{d}_{21}, \texttt{d}_{22}), \texttt{driveTruck}(\texttt{t}_3, \texttt{d}_{31}, \texttt{d}_{32}) \,\}. \end{aligned}$$

Dieser Plan enthält noch keine überflüssigen Aktionen. Der Plan bleibt jedoch korrekt und minimal lang, wenn man S_1 und S_2 durch

$$\begin{aligned} S_1' &= \{\, \texttt{loadTruck}(\texttt{p}_1, \texttt{t}_1, \texttt{d}_{11}), \texttt{driveTruck}(\texttt{t}_3, \texttt{d}_{31}, \texttt{d}_{32}) \,\} \text{ und} \\ S_2' &= \{\, \texttt{loadTruck}(\texttt{p}_2, \texttt{t}_2, \texttt{d}_{21}), \texttt{driveTruck}(\texttt{t}_3, \texttt{d}_{32}, \texttt{d}_{31}) \,\} \end{aligned}$$

ersetzt, enthält nun jedoch die beiden überflüssigen und zueinander inversen Leerfahrten $\texttt{driveTruck}(\texttt{t}_3, \texttt{d}_{31}, \texttt{d}_{32})$ und $\texttt{driveTruck}(\texttt{t}_3, \texttt{d}_{32}, \texttt{d}_{31})$ in Stadt $\texttt{c}_3$, die nicht zum Erreichen des spezifizierten Ziels beitragen.

3.4.3. Gesamtkodierungen

Um die oben angegebenen Kodierungen der Bedingungen an parallel angewandte Operatoren bei der Übersetzung von Planungsproblemen zu verwenden, reicht es, sie als weiteres Konjunktionsglied zur Basiskodierung hinzuzufügen.

Gelegentlich, etwa bei Aufrechterhaltungszielen der Form $\mathbf{G}\Phi$, wo $s_0 \models \Phi$, will man garantieren, dass nicht durch $S_t = \varnothing$ für alle t künstliche Null-Operationen eingeführt werden, um Pläne wie den trivialen Plan aus unendlich vielen Null-Operationen auszuschließen. Dazu genügt es, zur Basiskodierung ein Konjunktionsglied $\bigwedge_{t=0}^{b-1} \bigvee_{o \in O} o_t$ hinzuzufügen. Diese Möglichkeit werden wir jedoch bei den folgenden Kodierungen nicht mehr explizit erwähnen.

Definition 59. Sei φ eine LTL-Formel, $\mathcal{P} = \langle A, I, O, g \rangle$ eine Planungsinstanz und $[\![\mathcal{P}]\!]^b_{\varphi,\texttt{enc}}$, $\texttt{enc} \in \{\, 0, \ldots, 5 \,\}$, eine der Kodierungen von oben. Definiere $[\![\mathcal{P}, \varphi]\!]^b_{\texttt{enc}}$ durch

$$\begin{aligned} [\![\mathcal{P}, \varphi]\!]^b_{\texttt{enc}} &:= [\![\mathcal{P}]\!]^b_{basic} \wedge [\![\mathcal{P}]\!]^{b,1}_{1lin} \wedge [\![\mathcal{P}, \varphi]\!]^b_{BMC} \wedge [\![\mathcal{P}]\!]^b_{\varphi,\texttt{enc}} && \text{für } \texttt{enc} \in \{\, 0, \ldots, 4 \,\} \text{ und} \\ [\![\mathcal{P}, \varphi]\!]^b_{\texttt{enc}} &:= [\![\mathcal{P}]\!]^b_{basic} \wedge [\![\mathcal{P}]\!]^{b,2}_{1lin} \wedge [\![\mathcal{P}, \varphi]\!]^b_{BMC} && \text{für } \texttt{enc} = 5. \end{aligned}$$

3.5. Korrektheit

In diesem Abschnitt fügen wir die Ergebnisse der vorangegangenen Kapitel und Abschnitte zusammen und beweisen, dass die oben angegebenen Kodierungen das Gewünschte leisten.

Satz 39. *Sei φ eine $\mathsf{LTL}_{-\mathbf{X}}$-Formel, $\mathcal{P} = \langle A, I, O, g\rangle$ eine Planungsinstanz und $b \in \mathbb{N}$. Wenn $[\![\mathcal{P}, \varphi]\!]^b_{\mathtt{enc}}$ für $\mathtt{enc} \in \{0, \ldots, 5\}$ erfüllbar ist, so gibt es einen 1-Linearisierungs-Plan $\Pi = \langle S_0, \ldots, S_{b-1}\rangle$ der Länge b mit Ausführung $\pi = s_0, \ldots, s_b$ für $\mathcal{P}$ und eine Ausführung $\tilde{\pi}$ einer zulässigen Linearisierung von Π, so dass $\tilde{\pi} \models_{\tilde{b}} \varphi$, falls $\tilde{\pi}$ eine Schleife ist, bzw. $\tilde{\pi}(0), \ldots, \tilde{\pi}(\tilde{b}-1) \models_{\tilde{b}-1} \varphi$, sonst. Dabei ist $\tilde{b} = \sum_{t=0}^{b-1} |S_t|$.*

Beweis. Angenommen, $[\![\mathcal{P}, \varphi]\!]^b_{\mathtt{enc}}$ ist erfüllbar, etwa durch v. Dann gilt insbesondere $v \models [\![\mathcal{P}]\!]^b_{basic} \wedge [\![\mathcal{P}]\!]^{b,1}_{1lin}$ bzw. $v \models [\![\mathcal{P}]\!]^b_{basic} \wedge [\![\mathcal{P}]\!]^{b,2}_{1lin}$. Nach Lemma 14 ist dann $\Pi^v = \langle S^v_0, \ldots, S^v_{b-1}\rangle$ zusammen mit $\pi^v = s^v_0, \ldots, s^v_b$ ein 1-Linearisierungs-Plan der Länge b für $\mathcal{P}$ mit

$$\pi^v \models_b \varphi \quad \text{bzw.} \quad \pi^v(0), \ldots, \pi^v(b-1) \models_{b-1} \varphi. \tag{3.8}$$

Sei $\tilde{\pi}$ eine Ausführung einer zulässigen Linearisierung von Π^v. Nach Lemma 4 ist $v^b_{\Pi^v, \pi^v} = v$, d. h. $v^b_{\Pi^v, \pi^v} = v \models [\![\mathcal{P}]\!]^b_{\varphi, \mathtt{enc}}$. Damit und aus den Lemmata 29, 31 und 34 folgt zusammen mit Lemma 21 für alle Kodierungen $\mathtt{enc} \in \{0, \ldots, 5\}$, dass für alle $t \in \{0, \ldots, b-1\}$ die Menge S_t bzgl. $\prec_t$ für alle totalen Ordnungen $\prec_t$ auf S_t (bzw. für das bei der Kodierung gewählte $\prec_t$) zulässig ist. Nach Lemma 22 gilt dann $\pi^v \sim \tilde{\pi}$. Ist π eine (b,k)-Schleife, so gilt nach Lemma 23 zusätzlich, dass $u \sim \tilde{u}$ und $v \sim \tilde{v}$ mit $u, \tilde{u}, v, \tilde{v}$ wie in Lemma 23. Dann folgt mit Lemma 17 bzw. 18, dass

$$\begin{aligned} \pi^v \models_b \varphi \qquad &\text{gdw.} \qquad \tilde{\pi} \models_{\tilde{b}} \varphi \quad \text{bzw.} \\ \pi^v(0), \ldots, \pi^v(b-1) \models_{b-1} \varphi \qquad &\text{gdw.} \qquad \tilde{\pi}(0), \ldots, \tilde{\pi}(\tilde{b}-1) \models_{\tilde{b}-1} \varphi. \end{aligned} \tag{3.9}$$

Die Aussagen (3.8) und (3.9) zusammen ergeben, dass

$$\tilde{\pi} \models_{\tilde{b}} \varphi \quad \text{bzw.} \quad \tilde{\pi}(0), \ldots, \tilde{\pi}(\tilde{b}-1) \models_{\tilde{b}-1} \varphi. \tag{3.10}$$

Dass $\tilde{b} = \sum_{t=0}^{b-1} |S_t|$, ist klar. □

3.6. Vollständigkeit

Im letzten Abschnitt wurde gezeigt, dass die vorgestellten Kodierungen korrekt sind, d. h. dass ein mit Hilfe dieser Kodierungen berechneter Plan die gegebene Planungsaufgabe tatsächlich löst. Nach Latvala u. a. [2004] gibt es, wenn eine gegebene Planungsinstanz lösbar ist, einen *sequentiellen* Plan, dessen Länge durch einen aus der Planungsinstanz berechenbaren Wert K nach oben beschränkt ist. Aus der Existenz eines sequentiellen Plans der Länge b folgt die Existenz eines parallelen Plans derselben Länge, in dem jede Operatormenge S_t einelementig ist. Dass die einem solchen Plan entsprechende aussagenlogische Variablenbelegung zu einer jede unserer Kodierungen $[\![\mathcal{P}, \varphi]\!]^b_{\mathtt{enc}}$, $\mathtt{enc} \in \{0, \ldots, 5\}$, erfüllenden Belegung erweitert werden kann, ist klar.

Betrachtet man also Formeln Φ_b für alle b zwischen 1 und K, so findet man für lösbare Planungsaufgaben immer in endlicher Zeit einen Plan. Findet man umgekehrt keinen Plan

der Länge kleiner oder gleich K, so ist garantiert, dass es für die gegebene Planungsaufgabe keinen Plan, gleich welcher Länge, gibt.

Dies ist vor allem dann wichtig, wenn man das Planungsproblem mit temporal erweiterten Zielen nicht als Planungsproblem, sondern als Modellprüfungsproblem betrachtet, da in diesem Fall die Tatsache, dass es keinen Plan gibt, der Tatsache entspricht, dass kein Gegenbeispiel gegen die gegebene Spezifikation existiert und das überprüfte System damit korrekt ist. Wäre der Planungs- bzw. Modellprüfungsalgorithmus unvollständig, könnten Fehler unerkannt bleiben.

Der Wert $|Q| \cdot 2^{\|\varphi\|} = 2^{(|A|+\|\varphi\|)}$ kann als obere Schranke K dienen [Latvala u. a., 2004].

3.7. Diskussion

3.7.1. Größen der Kodierungen

Die in Abschnitt 3.4 angegebenen Kodierungen unterscheiden sich nicht nur in dem Maß an Parallelität, die sie in einem Plan zulassen, sondern auch hinsichtlich ihrer asymptotischen Größen. Da die Laufzeit eines SAT-Lösers im Allgemeinen exponentiell in der Größe der Eingabe wächst, ist es wichtig, möglichst kleine Formeln zu erzeugen.

In der folgenden Tabelle ist immer b die Anzahl der Zeitschritte, n die Anzahl der Operatoren, v die Anzahl der Zustandsvariablen und $F = \max(\{\, \|f\| \mid f \rhd d \in c \,\} \cup \{\, |d| \mid f \rhd d \in c \,\} \cup \{\, \|p\|, |e|, |c| \,\})$. Bei den Angaben zur Größe der Kodierung einer LTL-Formel in Aussagenlogik gehen wir von der in den Abschnitten 4.1.2.1 und 4.1.2.2 beschriebenen linearen Kodierung aus.

Kodierung	**Kodierungslänge**	**Aussagenvariable**
$[\![\mathcal{P}]\!]^b_{basic}$	$\mathcal{O}(b \cdot n \cdot v \cdot F^2)$	$\mathcal{O}(b \cdot (v+n))$
$[\![\mathcal{P}]\!]^{b,1}_{1lin}$	$\mathcal{O}(b \cdot v \cdot n)$	$\mathcal{O}(b \cdot v \cdot n)$
$[\![\mathcal{P}]\!]^{b,2}_{1lin}$	$\mathcal{O}(b \cdot v \cdot n)$	$\mathcal{O}(b \cdot v \cdot n)$
$[\![\mathcal{P}, \varphi]\!]^b_{BMC}$	$\mathcal{O}(b \cdot (\|\varphi\| + v))$	$\mathcal{O}(b \cdot (\|\varphi\| + v))$
$[\![\mathcal{P}]\!]^b_{\varphi,0}$	$\mathcal{O}(b \cdot n)$	$\mathcal{O}(b \cdot n)$
$[\![\mathcal{P}]\!]^b_{\varphi,1}$	$\mathcal{O}(b \cdot n^2)$	$\mathcal{O}(b \cdot n)$
$[\![\mathcal{P}]\!]^b_{\varphi,2}$	$\mathcal{O}(b \cdot n)$	$\mathcal{O}(b \cdot n)$
$[\![\mathcal{P}]\!]^b_{\varphi,3}$	$\mathcal{O}(b \cdot n^2)$	$\mathcal{O}(b \cdot n)$
$[\![\mathcal{P}]\!]^b_{\varphi,4}$	$\mathcal{O}(b \cdot \|var(\varphi)\| \cdot n)$	$\mathcal{O}(b \cdot \|var(\varphi)\| \cdot n)$

Tabelle 3.1.: Vergleich der Kodierungen hinsichtlich Länge und Anzahl der vorkommenden Aussagenvariablen.

3.7.2. Parallelität

Die folgenden beiden Beispiele sollen zeigen, dass mit unseren Kodierungen im besten Fall die Parallelisierbarkeit von Operatoren gleich hoch bleibt wie im Fall von Erreichbarkeitszielen, und dass im schlechtesten Fall keine Parallelisierbarkeit mehr gegeben ist, obwohl eine gleichwertige Formulierung der Planungsaufgabe mit einem Erreichbarkeitsziel anstelle eines temporal erweiterten Ziels maximale Parallelisierbarkeit erlaubt.

Bester Fall: Sei $(\langle \mathcal{P}_n, \varphi_n \rangle)_{n \in \mathbb{N}}$ die folgende Familie von Planungsaufgaben zusammen mit temporal erweiterten Zielen: $\mathcal{P}_n = \langle A_n, I_n, O_n, g_n \rangle$, wobei $A_n = \{ a_1, \ldots, a_n, a_{\text{goal}} \}$, $I_n = \bigwedge_{i=1}^{n} \neg a_i \wedge \neg a_{\text{goal}}$, $O_n = \{ o_1, \ldots, o_n, o_{\text{goal}} \}$, $o_i = \langle \top, \{ a_i \}, \varnothing \rangle$ für alle $i \in \{ 1, \ldots, n \}$, $o_{\text{goal}} = \langle \bigwedge_{i=1}^{n} a_i, \{ a_{\text{goal}} \}, \varnothing \rangle$ und $g_n = \top$ sowie $\varphi_n = \mathbf{F} a_{\text{goal}}$.

Dann gilt für jedes $n \in \mathbb{N}$: Ein kürzester sequentieller Plan Π_{seq} für $\mathcal{P}_n$ und φ_n hat die Länge $n+1$, etwa $\Pi_{\text{seq}} = o_1; o_2; \ldots; o_n; o_{\text{goal}}$, während der kürzeste parallele Plan $\Pi_{\text{par}} = S_0; S_1$ mit $S_0 = \{ o_1, \ldots, o_n \}$ und $S_1 = \{ o_{\text{goal}} \}$ nach einer der Kodierungen enc_j, $j \in \{ 1, \ldots, 5 \}$, die Länge 2 besitzt. Das Verhältnis der Planlängen ist also in $\Theta(n)$ für Instanzgröße n.

Schlechtester Fall: Sei wieder $(\langle \mathcal{P}_n, \varphi_n \rangle)_{n \in \mathbb{N}}$ eine Familie von Planungsaufgaben zusammen mit temporal erweiterten Zielen: $\mathcal{P}_n = \langle A_n, I_n, O_n, g_n \rangle$, wobei $A_n = \{ a_1, \ldots, a_n \}$, $I_n = \bigwedge_{i=1}^{n} \neg a_i$, $O_n = \{ o_1, \ldots, o_n \}$ mit $o_i = \langle \top, \{ a_i \}, \varnothing \rangle$ für alle $i \in \{ 1, \ldots, n \}$ und $g_n = \top$ sowie $\varphi_n = \mathbf{F}(\bigwedge_{i=1}^{n} a_i)$.

Dann hat für jedes $n \in \mathbb{N}$ sowohl der kürzeste sequentielle als auch der kürzeste mit den Übersetzungen enc_j, $j \in \{ 1, \ldots, 5 \}$, gefundene parallele Plan für $\mathcal{P}_n$ und φ_n die Länge n, da je zwei unterschiedliche Operatoren bezüglich φ_n interferieren. Diese Kodierungen erlauben hier also keine Parallelität mehr.

Die Planungsaufgabe $\langle \mathcal{P}_n, \varphi_n \rangle$ lässt sich auch ohne temporal erweitertes Ziel als $\langle \mathcal{P}'_n, \varphi'_n \rangle$ formulieren, wobei $\mathcal{P}'_n = \langle A_n, I_n, O_n, g'_n \rangle$ mit A_n, I_n, O_n wie oben und $g'_n = \bigwedge_{i=1}^{n} a_i$ sowie $\varphi'_n = \top$. Der Planungsalgorithmus ohne temporal erweiterte Ziele findet für diese Aufgabe einen parallelen Plan der Länge 1, da alle Operatoren vollständig voneinander unabhängig sind. Durch unsere Art, temporal erweiterte Ziele zuzulassen, erhöht sich bei dieser Familie von Planungsaufgaben also die Anzahl der benötigten Zeitpunkte um einen Faktor $\Theta(n)$ für Instanzgröße n.

Die Beispiele zeigen, dass die Kodierungen vor allem dann gut sind, wenn nur ein kleiner Teil der Zustandsvariablen in der Zielspezifikation vorkommt und wenn in einem Plan, der die gegebene Planungsaufgabe löst, viele Operatoren angewandt werden müssen, die keine der in der Spezifikation vorkommenden Variablen positiv oder negativ als Effekt haben.

Kapitel 4.

Implementierung und Experimente

In diesem Kapitel stellen wir eine Implementierung des im vorigen Kapitel beschriebenen Planungs- bzw. Modellprüfungs-Verfahrens vor.

4.1. Implementierung

Die Implementierung erfolgte durch Erweiterung des von Rintanen in SML [Paulson, 1996] entwickelten Planungssystems SMLSATPLANNER um ein Modul für beschränkte Modellprüfung, d. h. ein Modul, das die Kodierung aus Abschnitt 2.4.3 und die neuen Kodierungen aus Abschnitt 3.4 durchführt.

4.1.1. Planer

Parallele Transitionen, der Aufbau eines Deaktivierungsgraphen und dessen in Abschnitt 2.3.3 beschriebene Übersetzung in eine aussagenlogische Formel waren in dem Planer bereits implementiert, d. h. es genügte, dem Deaktivierungsgraphen wie in Abschnitt 3.4.2 beschrieben zusätzliche Kanten hinzuzufügen und die resultierende aussagenlogische Formel um das Konjunktionsglied zu erweitern, das der Übersetzung des beschränkten Modellprüfungs-Problems entspricht.

4.1.2. Modellprüfer

Um zu garantieren, dass der Übersetzungsvorgang in linearer Zeit möglich ist und die Übersetzung lineare Größe in der Größe der LTL-Formel besitzt, müssen zwei Dinge beachtet werden: Mehrfach vorkommende Teilformeln der LTL-Formel dürfen nur einmal übersetzt werden, und die Transformation der resultierenden aussagenlogischen Formel in konjunktive Normalform muss so durchgeführt werden, dass sie nur lineare Zeit benötigt und sie die Formel nicht wesentlich, d. h. nur um einen kleinen konstanten Faktor, vergrößert.

4.1.2.1. Hilfsvariable für gemeinsame Teilformeln

Um Teilformeln nur einmal zu übersetzen, genügt es, wie von Latvala u. a. [2004] beschrieben, für jede Teilformel φ der gegebenen LTL-Formel und jeden Zeitpunkt t bei der ersten Berechnung von $[\![\varphi]\!]_b^t$ eine neue aussagenlogische Hilfsvariable $a_{\varphi,t,b}$ einzuführen, die Formel $a_{\varphi,t,b} \leftrightarrow [\![\varphi]\!]_b^t$ als Konjunktionsglied der Übersetzung hinzuzufügen und jedesmal, wenn $[\![\varphi]\!]_b^t$ berechnet werden soll, $a_{\varphi,t,b}$ zurückzugeben.

4.1.2.2. Lineare Transformation in konjunktive Normalform

Da die resultierende Formel in konjunktiver Normalform (KNF) an den SAT-Löser übergeben werden soll, muss sie noch in diese Form überführt werden. Im Allgemeinen führt diese Transformation zu einer exponentiellen Vergrößerung der Formel [Schöning, 2000]. Diese Vergrößerung ist bei der Basiskodierung unter Umständen noch hinnehmbar, wird jedoch bei der Übersetzung der LTL-Spezifikation problematisch, da in dieser Übersetzung häufig Disjunktionen über Konjunktionen vorkommen, wie etwa durch die Disjunktion über das Sprungziel einer möglichen Schleife oder bei der Übersetzung von Teilformeln der Gestalt $\mathbf{FG}\varphi$.

In unserem Fall genügt jedoch eine nicht logisch äquivalente, sondern nur erfüllbarkeitsäquivalente Transformation, wie sie von Tseitin [1968] beschrieben wird.

Bei dieser Umformung werden in der Formel zunächst die Symbole $\rightarrow$ und $\leftrightarrow$ eliminiert, so dass nur noch Variable, Konjunktionen, Disjunktionen und Negationen in der Formel vorkommen. Danach wird die Formel in Negationsnormalform [Schöning, 2000] gebracht. Schließlich wird als Endergebnis $cnf(\Phi, aux_\Phi)$ berechnet, wobei

$$
\begin{aligned}
cnf(\ell, aux) &:= \ell, \qquad \text{falls } \ell \text{ positives oder negatives Literal,} \\
cnf(\Phi_1 \wedge \Phi_2, aux) &:= aux \wedge cnf(\Phi_1, aux_{\Phi_1}) \wedge cnf(\Phi_2, aux_{\Phi_2}) \wedge \\
&\quad\ cnf'(aux \leftrightarrow aux_{\Phi_1} \wedge aux_{\Phi_2}), \\
cnf(\Phi_1 \vee \Phi_2, aux) &:= aux \wedge cnf(\Phi_1, aux_{\Phi_1}) \wedge cnf(\Phi_2, aux_{\Phi_2}) \wedge \\
&\quad\ cnf'(aux \leftrightarrow aux_{\Phi_1} \vee aux_{\Phi_2}), \\
cnf'(\ell_1 \leftrightarrow \ell_2 \wedge \ell_3) &:= (\overline{\ell_1} \vee \ell_2) \wedge (\overline{\ell_1} \vee \ell_3) \wedge (\ell_1 \vee \overline{\ell_2} \vee \overline{\ell_3}) \quad \text{und} \\
cnf'(\ell_1 \leftrightarrow \ell_2 \vee \ell_3) &:= (\overline{\ell_1} \vee \ell_2 \vee \ell_3) \wedge (\ell_1 \vee \overline{\ell_2}) \wedge (\ell_1 \vee \overline{\ell_3}).
\end{aligned}
$$

Dabei ist aux_Φ gleich Φ, falls Φ ein Literal ist, und eine neue, bisher unbenutzte Hilfsvariable, sonst.

Diese Umformung ist in linearer Zeit möglich und erzeugt eine Formel, die nur unwesentlich größer als die Ausgangsformel ist. Die neue Formel enthält im Allgemeinen zusätzliche neue Hilfsvariable, hat aber zugleich die Eigenschaft, dass eine erfüllende Belegung für sie auch eine erfüllende Belegung für die ursprüngliche Formel ist und dass eine erfüllende Belegung für die ursprüngliche Formel zu einer erfüllenden Belegung für die KNF-Formel erweitert werden kann. Diese Eigenschaft ist für unsere Zwecke ausreichend.

4.2. Experimente

4.2.1. Aufbau

In Ermangelung einer größeren Zahl von PDDL-kodierten Testdomänen und -instanzen wurden nur Versuche mit einzelnen handkodierten Planungsaufgaben durchgeführt, bei denen darauf geachtet wurde, unterschiedliche Typen von temporal erweiterten Zielen zu spezifizieren. Alle Planungsaufgaben gehören der LOGISTICS-Domäne an und enthalten die gleichen Objekte wie die Instanz in Abschnitt 2.6. Sie unterscheiden sich lediglich in der Wahl des Anfangszustands und der $\mathsf{LTL}_{-\mathbf{X}}$-Spezifikation. Tabelle 4.1 gibt die gewählten Spezifikationen

und Anfangszustände an. Dabei ist

$$p_{ij} = \mathtt{at}(\mathtt{p}_i, \mathtt{d}_{ij}), \qquad \hat{p}(k,l,m) = (p_{1k} \wedge p_{2l} \wedge p_{3m}), \qquad \check{p}(k,l,m) = (p_{1k} \vee p_{2l} \vee p_{3m}),$$
$$t_{ij} = \mathtt{at}(\mathtt{t}_i, \mathtt{d}_{ij}), \qquad \hat{t}(k,l,m) = (t_{1k} \wedge t_{2l} \wedge t_{3m}) \qquad \text{und}$$
$$L = \{\, \neg\mathtt{in}(\mathtt{p}_2, \mathtt{t}_2), \neg\mathtt{in}(\mathtt{p}_3, \mathtt{t}_3), \mathtt{at}(\mathtt{t}_2, \mathtt{d}_{22}), \mathtt{at}(\mathtt{t}_3, \mathtt{d}_{32}) \,\}\,.$$

Nr.	$\mathsf{LTL}_{-\mathbf{X}}$-Spezifikation	Anfangszustand	Bemerkung
φ_1	$\mathbf{GF}\hat{p}(1,1,1) \wedge \mathbf{GF}\hat{p}(2,2,2)$	$\hat{p}(1,1,1) \wedge \hat{t}(2,2,2)$	Erfordert Schleife
φ_2	$\mathbf{F}\Big(\hat{p}(1,2,2) \wedge \mathbf{F}\big(\hat{p}(1,1,2) \wedge \mathbf{F}\hat{p}(1,1,1)\big)\Big)$	$\hat{p}(2,2,2) \wedge \hat{t}(2,2,2)$	Zwischenziele
φ_3	$\mathbf{F}\hat{p}(1,1,1)$	$\hat{p}(2,2,2) \wedge \hat{t}(2,2,2)$	Erreichbarkeitsziel
φ_4	$\mathbf{FG}\hat{p}(1,1,1)$	$\hat{p}(2,2,2) \wedge \hat{t}(2,2,2)$	Aufrechterhaltungsziel
φ_5	$\mathbf{FG}p_{11} \wedge \bigwedge_{\ell \in L} (\ell \rightarrow \mathbf{G}\ell)$	$\hat{p}(2,2,2) \wedge \hat{t}(1,1,1)$	Vermeidung überflüssiger Aktionen
φ_5	$\bigwedge_{i=1}^{3} \bigwedge_{j=1}^{2} \mathbf{G}(p_{ij} \rightarrow \mathbf{F}p_{i(3-j)})$	$\hat{p}(1,1,1) \wedge \hat{t}(2,2,2)$	Reaktion auf Zustand

Tabelle 4.1.: Für Experimente verwendete Instanzen bzw. $\mathsf{LTL}_{-\mathbf{X}}$-Spezifikationen

Neben den genannten Planungsaufgaben wurde als Beispiel für eine Anwendung in der Modellprüfung eine Lösung des Mutex-Problems (vgl. z. B. Clarke u. a. [2002]) in PDDL kodiert. Dabei gibt es für jeden Prozess p je eine Variable, die angibt, ob sich p in seinem kritischen oder nicht-kritischen Abschnitt befindet oder ob er darauf wartet, den kritischen Abschnitt zu betreten ($\mathtt{critical}(p)$, $\mathtt{noncritical}(p)$ bzw. $\mathtt{trying}(p)$). Weiter gibt es eine Variable `turn`, die angibt, welcher Prozess im Konfliktfall seinen kritischen Abschnitt betreten darf. Es gibt die folgenden Aktionen: (a) ein Prozess p kann in den Wartezustand übergehen, wenn er sich im nicht-kritischen Abschnitt aufhält, (b) er kann den kritischen Abschnitt betreten, wenn er bereits wartet und der andere Prozess seinen kritischen Abschnitt nicht betreten will, oder wenn beide Prozesse ihre kritischen Abschnitte betreten wollen, aber p an der Reihe ist, d. h. $\mathtt{turn} = i$ für $p = \mathtt{proc}_i$, und (c) er kann den kritischen Abschnitt verlassen, wenn er sich darin befindet. (d) Befindet sich ein Prozess $\mathtt{proc}_i$ ($i = 0, 1$) in seinem kritischen Abschnitt, kann $\mathtt{turn} = 1 - i$ gesetzt werden. Zu Beginn sind die Prozesse $\mathtt{proc}_0$ und $\mathtt{proc}_1$ in ihren nicht-kritischen Abschnitten und es gilt $\mathtt{turn} = 0$. Es wird nach einem Gegenbeispiel gegen die Spezifikation $\mathbf{G}(\mathtt{trying}(\mathtt{proc}_0) \rightarrow \mathbf{F}\,\mathtt{critical}(\mathtt{proc}_0))$ gesucht. Da keine Fairness garantiert ist, gibt es ein Gegenbeispiel der Länge 8. Prozess $\mathtt{proc}_0$ betritt dabei einmal seinen kritischen Abschnitt, damit die `turn`-Variable auf 1 gesetzt werden kann, und verlässt den kritischen Abschnitt wieder. Danach gehen beide Prozesse in den Wartezustand. Nun kann sich $\mathtt{proc}_1$ wegen $\mathtt{turn} = 1$ beliebig lange zwischen seinem kritischen und nicht-kritischen Abschnitt sowie dem Wartezustand bewegen, ohne dass $\mathtt{proc}_0$ wieder seinen kritischen Abschnitt betritt.

Alle Experimente, bei denen Laufzeiten angegeben sind, wurden auf einem Rechner mit 2-GHz-AMD-Athlon-Prozessor mit 512 MB RAM unter dem Betriebssystem SuSE Linux 10.0 durchgeführt. Als SAT-Löser kam Siege in der Version 4 zum Einsatz. Wurde die Erfüllbarkeit einer Formel nicht innerhalb von 10 Sekunden entschieden, so wurde die Berechnung abgebrochen und zur nächsten Formel übergegangen.

Die neben den mittleren Laufzeiten angegebenen Konfidenzintervalle wurden mit einem Bootstrapping-Verfahren [Efron und Tibshirani, 1993] ermittelt, wobei für jedes Konfidenzintervall 4000-mal eine 100-elementige Stichprobe mit Zurücklegen aus den jeweils 100 gemessenen Stichproben entnommen und von dieser der Mittelwert gebildet wurde. Die Grenzen des 5-Prozent-Konfidenzintervalls sind dann der 100ste und der 3900ste Wert unter den aufsteigend sortierten Mittelwerten.

4.2.2. Ergebnisse

In den folgenden Tabellen ist $enc_{\texttt{enc}} = [\![\mathcal{P}, \varphi]\!]^b_{\texttt{enc}}$ für $\texttt{enc} \in \{0, \ldots, 5\}$.

In Tabelle 4.2 ist für die in Abschnitt 3.4.3 vorgestellten Kodierungen und die in Tabelle 4.1 angegebenen Planungsinstanzen dargestellt, wieviele Zeitpunkte ein kürzester paralleler Plan für die jeweilige Instanz bei gegebener Kodierung enthält. Entsprechend stellt Tabelle 4.3 die Anzahl der Operatoren in den jeweils kürzesten Plänen dar.

φ	enc_0	enc_1	enc_2	enc_3	enc_4	enc_5
φ_1	21	15	15	15	13	13
φ_2	10	9	9	9	7	7
φ_3	10	5	5	5	5	5
φ_4	10	5	5	5	5	5
φ_5	5	4	4	4	4	4
φ_6	21	15	15	15	13	13

Tabelle 4.2.: Anzahlen von Zeitpunkten in kürzesten Plänen

φ	enc_0	enc_1	enc_2	enc_3	enc_4	enc_5
φ_1	21	21	21	21	40	34
φ_2	10	11	11	9	11	21
φ_3	10	10	10	9	11	16
φ_4	9	9	9	9	11	10
φ_5	4	4	4	4	4	4
φ_6	21	21	21	21	40	32

Tabelle 4.3.: Anzahlen von Operatoren in kürzesten Plänen

Tabelle 4.4 zeigt die Laufzeiten von SIEGE bei der Berechnung dieser Pläne zusammen mit den zugehörigen 5-Prozent-Konfidenzintervallen. Dabei wurde von einer sequentiellen Auswertung der Formeln ausgegangen, d. h. es wurde mit der Auswertung der Formel Φ_1 für Planlänge 1 begonnen, von Φ_n zu Φ_{n+1} übergegangen, wenn Φ_n nicht erfüllbar war, und das Verfahren beendet, wenn Φ_n erfüllbar war. Die angegebenen Laufzeiten sind die Summen der einzelnen Laufzeiten für die Entscheidung der Erfüllbarkeit von Φ_i für i zwischen 1 und dem jeweils ersten n, für das Φ_n erfüllbar ist.

In Abbildungen 4.1 und 4.2 ist exemplarisch dargestellt, wie die kumulierten Laufzeiten aus Tabelle 4.4 für φ_1 (Abb. 4.1) und φ_6 (Abb. 4.2) zustande kommen. Im Fall von φ_1 sind die

φ	enc_0		enc_1		enc_2		enc_3		enc_4		enc_5	
φ_1	5,83	5,70 5,95	0,20	0,20 0,20	0,20	0,20 0,20	0,20	0,19 0,20	0,16	0,16 0,16	0,13	0,13 0,13
φ_2	0,09	0,09 0,09	0,08	0,08 0,08	0,08	0,08 0,08	0,08	0,08 0,08	0,06	0,06 0,06	0,06	0,06 0,06
φ_3	0,09	0,09 0,09	0,04	0,04 0,04	0,04	0,04 0,04	0,04	0,04 0,04	0,04	0,04 0,04	0,04	0,04 0,04
φ_4	0,09	0,09 0,09	0,04	0,04 0,04	0,04	0,04 0,04	0,04	0,04 0,04	0,04	0,04 0,04	0,04	0,04 0,04
φ_5	0,04	0,04 0,04	0,03	0,03 0,03	0,03	0,03 0,03	0,03	0,03 0,03	0,03	0,03 0,03	0,03	0,03 0,03
φ_6	51,45	51,10 51,79	2,02	1,97 2,08	2,23	2,16 2,30	2,07	2,02 2,13	2,85	2,79 2,92	2,31	2,26 2,35

Tabelle 4.4.: Kumulierte Laufzeiten von SIEGE bis zur ersten erfüllbaren Formel

Kodierungen enc_0 und enc_5 durch hellgraue bzw. weiße Balken dargestellt, im Fall von φ_6 die Kodierungen enc_0, enc_1 und enc_5 durch hellgraue, dunkelgraue und weiße Balken. Die Laufzeiten für die restlichen Kodierungen sind nicht angegeben. Sie liegen in beiden Fällen zwischen den beiden Extremen enc_0 und enc_5, im Fall von φ_6 sind die Laufzeitkurven von enc_1, enc_2, enc_3 und enc_4 beinahe identisch.

In beiden Diagrammen sind Phasenübergänge zwischen unerfüllbaren und erfüllbaren Formeln zu erkennen. Die höchsten Laufzeiten treten häufig bei den letzten unerfüllbaren und den ersten erfüllbaren Formeln auf.

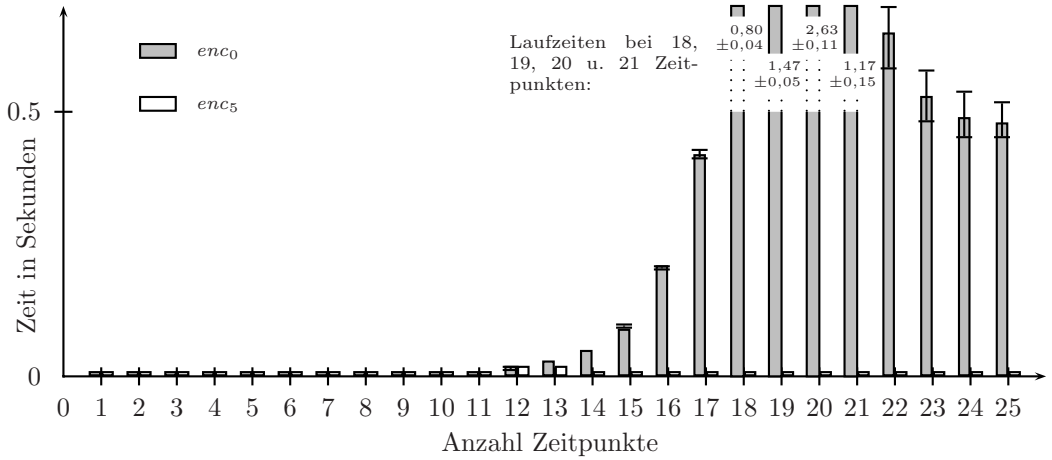


Abbildung 4.1.: Laufzeiten von SIEGE bei Spezifikation φ_1 für Kodierungen enc_0 und enc_5

In Tabellen 4.6 und 4.7 werden die durch die Kodierungen enc_0 bis enc_5 erzeugten Formeln hinsichtlich ihrer Größe verglichen. Insbesondere werden die Anzahlen *vars* der in ihnen vorkommenden Variablen, die Anzahlen *clauses* der Klauseln, aus denen ihre konjunktive Normalform besteht, wenn die Umformung wie in Abschnitt 4.1.2.2 beschrieben erfolgt, die mit ihren Häufigkeiten gezählten Anzahlen *occurr* der vorkommenden Variablen sowie die Dateigrößen *fs* der Kodierungen im DIMACS-Format [DIMACS, 1993] gegenübergestellt.

Während in Tabelle 4.6 für feste Spezifikationen φ_i die Formelgrößen jeweils für die Anzahl von Zeitpunkten verglichen werden, bei der mit Kodierung enc_0 erstmals Erfüllbarkeit auf-

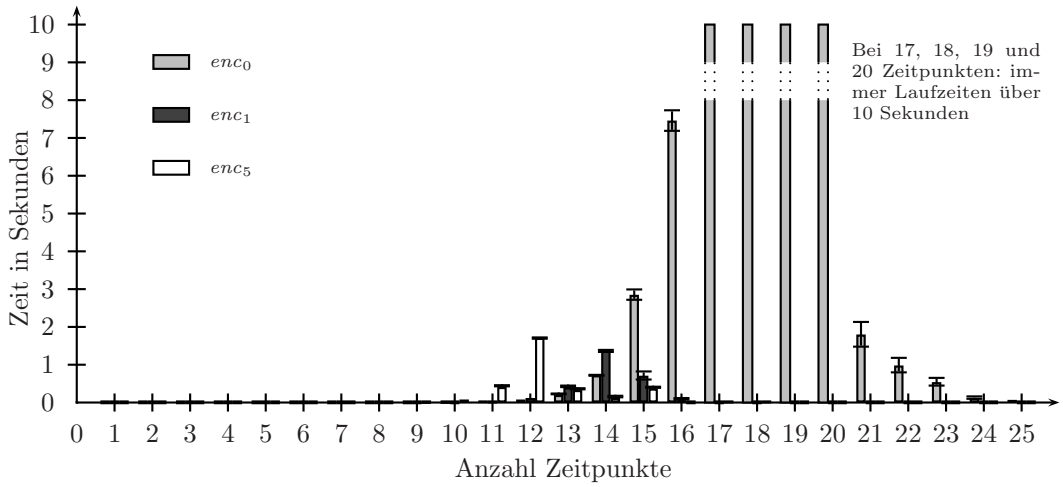


Abbildung 4.2.: Laufzeiten von SIEGE bei Spezifikation φ_6 für Kodierungen enc_0, enc_1, enc_5

tritt, werden in Tabelle 4.7 die Formelgrößen der für alle Paare aus Spezifikation φ_i und Kodierung enc_j jeweils ersten erfüllbaren Formeln gegenübergestellt.

φ	Starke Zusammenhangskomponenten
φ_1	$1 \times 12 + \ 6 \times 1$
φ_2	$1 \times 10 + \ 8 \times 1$
φ_3	$1 \times \ 6 + 12 \times 1$
φ_4	$1 \times \ 6 + 12 \times 1$
φ_5	$1 \times 14 + \ 4 \times 1$
φ_6	$1 \times 12 + \ 6 \times 1$

Tabelle 4.5.: Anzahlen und Größen starker Zusammenhangskomponenten im erweiterten Deaktivierungsgraphen

Die mögliche Anzahl paralleler Operatoren hängt bei den Kodierungen enc_4 und enc_5 entscheidend von der Zahl und den Größen der starken Zusammenhangskomponenten des erweiterten Deaktivierungsgraphen ab. Tabelle 4.5 zeigt die Anzahlen und Größen starker Zusammenhangskomponenten in der Form $\sum_n (scc(n) \times n)$, wobei $scc(n)$ die Anzahl der starken Zusammenhangskomponenten mit genau n Elementen ist.

Im Mutex-Beispiel wurden nur noch die Kodierungen enc_0 und enc_5 miteinander verglichen. Beide führten zu Plänen der Länge 8, d. h. das Zulassen von Parallelität brachte keinen Gewinn, vielmehr benötigte SIEGE für die Entscheidung der Erfüllbarkeit der mit Kodierung enc_5 erzeugten Formeln rund doppelt so lange wie bei Formeln, die mit Kodierung enc_0 erzeugt wurden.

φ	enc	b	$vars/10^3$	$clauses/10^3$	$occurr/10^3$	fs/kB
φ_1	enc_0	21	4,38	12,52	29,03	179,72
φ_1	enc_1	21	3,58	14,56	33,14	210,01
φ_1	enc_2	21	4,12	13,34	30,66	191,69
φ_1	enc_3	21	3,58	16,07	36,17	231,18
φ_1	enc_4	21	7,86	18,57	49,46	307,51
φ_1	enc_5	21	3,79	11,87	27,77	171,01
φ_2	enc_0	10	2,09	5,98	13,85	82,24
φ_2	enc_1	10	1,71	6,60	15,11	91,76
φ_2	enc_2	10	1,93	6,32	14,53	87,32
φ_2	enc_3	10	1,71	7,40	16,71	102,96
φ_2	enc_4	10	3,41	8,20	21,61	130,61
φ_2	enc_5	10	1,79	5,43	12,77	74,94
φ_3	enc_0	10	1,47	4,26	9,73	56,33
φ_3	enc_1	10	1,09	4,18	9,59	53,79
φ_3	enc_2	10	1,23	4,26	9,73	55,91
φ_3	enc_3	10	1,09	4,90	11,03	62,44
φ_3	enc_4	10	2,11	5,14	13,49	79,76
φ_3	enc_5	10	1,13	3,35	7,93	44,07
φ_4	enc_0	10	1,62	4,68	10,73	62,48
φ_4	enc_1	10	1,24	4,60	10,59	61,57
φ_4	enc_2	10	1,38	4,68	10,73	62,75
φ_4	enc_3	10	1,24	5,32	12,03	71,22
φ_4	enc_4	10	2,26	5,56	14,49	85,68
φ_4	enc_5	10	1,28	3,77	8,93	50,39
φ_5	enc_0	5	1,24	3,47	8,04	45,01
φ_5	enc_1	5	1,05	4,12	9,35	52,03
φ_5	enc_2	5	1,20	3,64	8,38	47,22
φ_5	enc_3	5	1,05	4,36	9,83	54,91
φ_5	enc_4	5	1,90	4,58	11,94	69,56
φ_5	enc_5	5	1,11	3,36	7,84	43,19
φ_6	enc_0	21	7,82	21,92	51,46	322,02
φ_6	enc_1	21	7,02	23,95	55,58	352,30
φ_6	enc_2	21	7,56	22,74	53,10	333,99
φ_6	enc_3	21	7,02	25,47	58,60	373,47
φ_6	enc_4	21	11,30	27,96	71,90	466,60
φ_6	enc_5	21	7,23	21,27	50,20	313,30

Tabelle 4.6.: Formelgrößen bei unterschiedlichen Kodierungen I

φ	enc	b	$vars/10^3$	$clauses/10^3$	$occurr/10^3$	fs/kB
φ_1	enc_0	21	4,38	12,52	29,03	179,72
φ_1	enc_1	15	2,57	10,43	23,73	148,52
φ_1	enc_2	15	2,96	9,56	21,96	135,44
φ_1	enc_3	15	2,57	11,51	25,89	163,64
φ_1	enc_4	13	4,89	11,53	30,69	188,33
φ_1	enc_5	13	2,36	7,39	17,26	103,83
φ_2	enc_0	10	2,09	5,98	13,85	82,24
φ_2	enc_1	9	1,55	5,94	13,61	82,01
φ_2	enc_2	9	1,74	5,69	13,08	78,02
φ_2	enc_3	9	1,55	6,66	15,05	92,09
φ_2	enc_4	7	2,40	5,76	15,16	89,61
φ_2	enc_5	7	1,27	3,82	8,97	50,62
φ_3	enc_0	10	1,47	4,26	9,73	56,33
φ_3	enc_1	5	0,55	2,10	4,81	25,98
φ_3	enc_2	5	0,62	2,14	4,88	26,29
φ_3	enc_3	5	0,55	2,46	5,53	30,30
φ_3	enc_4	5	1,06	2,58	6,76	37,69
φ_3	enc_5	5	0,57	1,69	3,98	20,93
φ_4	enc_0	10	1,62	4,68	10,73	62,48
φ_4	enc_1	5	0,63	2,32	5,33	28,75
φ_4	enc_2	5	0,70	2,36	5,40	29,06
φ_4	enc_3	5	0,63	2,68	6,05	33,08
φ_4	enc_4	5	1,14	2,80	7,28	40,79
φ_4	enc_5	5	0,65	1,91	4,50	23,70
φ_5	enc_0	5	1,24	3,47	8,04	45,01
φ_5	enc_1	4	0,86	3,34	7,59	41,43
φ_5	enc_2	4	0,98	2,96	6,82	36,56
φ_5	enc_3	4	0,86	3,53	7,98	43,73
φ_5	enc_4	4	1,54	3,71	9,66	55,27
φ_5	enc_5	4	0,91	2,74	6,38	33,89
φ_6	enc_0	21	7,82	21,92	51,46	322,02
φ_6	enc_1	15	5,05	17,18	39,85	250,83
φ_6	enc_2	15	5,44	16,31	38,08	237,75
φ_6	enc_3	15	5,05	18,26	42,01	265,95
φ_6	enc_4	13	7,05	17,40	44,71	277,28
φ_6	enc_5	13	4,52	13,26	31,28	192,79

Tabelle 4.7.: Formelgrößen bei unterschiedlichen Kodierungen II

Kapitel 5.

Zusammenfassung

5.1. Ergebnisse

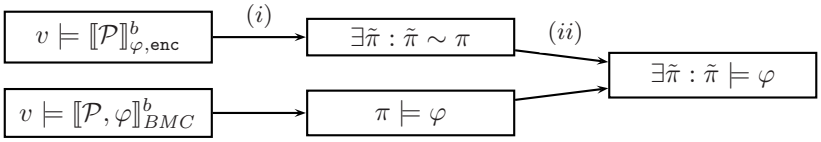
Ziel dieser Arbeit war es, einen gegebenen korrekten und vollständigen Handlungsplanungsalgorithmus so zu erweitern, dass er neben Plänen für Erreichbarkeitsziele auch solche für temporal erweiterte Ziele finden kann. Diese Erweiterung ist durch die Verwendung der von Biere u. a. [2003] bzw. Latvala u. a. [2004] angegebenen Übersetzung temporallogischer Formeln in Aussagenlogik möglich. Darüber hinaus sollte untersucht werden, inwieweit es möglich ist, die im Basisalgorithmus mit Erfolg eingesetzte Verwendung paralleler Operatoren in den um temporale Ziele erweiterten Algorithmus zu übertragen. Wie erwartet nimmt die Parallelisierbarkeit von Operatoren durch die Einführung temporal erweiterter Ziele ab. Im besten Fall bleibt sie im Wesentlichen gleich hoch wie bei Erfüllbarkeitszielen, während im schlechtesten Fall parallele Pläne zu sequentiellen Plänen degenerieren, obwohl bei äquivalenter Formulierung des temporal erweiterten Ziels als Erreichbarkeitsziel maximale Parallelität möglich ist.

Der in dieser Arbeit vorgestellte Planungsalgorithmus für temporal erweiterte Ziele ist wie der Algorithmus für Erreichbarkeitsziele korrekt und vollständig. Häufig liegt seine Laufzeit wegen der geringeren Anzahl von zu betrachtenden Zeitpunkten unterhalb der Laufzeit eines entsprechenden Planers ohne Verwendung paralleler Operatoren.

Die erzielten Ergebnisse können dazu beitragen, erfüllbarkeitsbasiertes Planen mit temporal erweiterten Zielen bzw. beschränkte Modellprüfung durch die Berücksichtigung paralleler Transitionen effizienter zu gestalten.

5.2. Ausblick

Wir haben gezeigt, dass die angegebenen Bedingungen an die Parallelität von Operatoren (gleiches Verhalten, Zulässigkeit in einer bestimmten Reihenfolge) hinreichend dafür sind, dass es zu einem sequentiellen Plan einen entsprechenden parallelen Plan mit äquivalenter Ausführung gibt. Weiter wissen wir, dass aus der Äquivalenz zweier Ausführungspfade folgt, dass sie dieselben $\mathsf{LTL}_{-\mathbf{X}}$-Formeln mit gegebener Variablenmenge erfüllen. Schematisch haben wir unter der Voraussetzung, dass $v \models [\![\mathcal{P}]\!]^b_{basic} \wedge [\![\mathcal{P}]\!]^{b,1}_{1lin}$, also



Die Bedingungen (i) und (ii) sind jedoch nur hinreichend, nicht aber notwendig, wie die folgenden Beispiele zeigen:

Zu (i): Sei φ eine $\mathsf{LTL}_{-\mathbf{X}}$-Formel mit $var(\varphi) = \{ a_1 \}$, seien $o_1 = \langle a_1, \{ \neg a_2 \}, \varnothing \rangle$ und $o_2 = \langle \top, \{ \neg a_1 \}, \varnothing \rangle$ Operatoren und s ein Zustand mit $s \models a_1 \wedge a_2$. Dann sind o_1 und o_2 in s simultan und in der Reihenfolge o_1 vor o_2 anwendbar. Die Anwendung von o_2 vor o_1 zu einem Zeitpunkt t ist durch die Parallelitätsaxiome ausgeschlossen, da o_2 die Vorbedingung von o_1 falsifiziert. Unsere Kodierungen lassen wegen $[o_2]^{\varphi}_{\Diamond} = \{ \neg a_1 \} \not\subseteq \varnothing = [o_1]^{\varphi}_{\Box}$ auch die Reihenfolge o_1 vor o_2 nicht zu, obwohl mit $s'' = app_{o_1}(s)$ und $s' = app_{o_2}(s'')$, d. h. $s'' \models a_1 \wedge \neg a_2$ und $s' \models \neg a_1 \wedge \neg a_2$, die Pfade s, s' und s, s'', s' bezüglich der Variablen in φ äquivalent sind.

Zu (ii): Sei $\varphi = \mathbf{F}(a_1 \wedge a_2 \wedge a_3)$ und s ein Zustand mit $s \models \neg a_1 \wedge \neg a_2 \wedge \neg a_3$. Seien ferner $o_i = \langle \top, \{ a_i \}, \varnothing \rangle$, $i \in \{ 1, 2, 3 \}$, Operatoren. Dann sind o_1, o_2 und o_3 in s sowohl simultan als auch in jeder der sechs möglichen Reihenfolgen sequentiell anwendbar und ihre Anwendung führt in einen Zustand s' mit $s' \models a_1 \wedge a_2 \wedge a_3$. Keiner der Ausführungspfade, der bei der Anwendung von o_1, o_2 und o_3 in einer der sechs möglichen Anordnungen entsteht, ist bezüglich der Variablen in φ zu der parallelen Ausführung s, s' äquivalent. Dennoch erfüllt jeder diese Pfade ebenfalls φ.

Um mehr Parallelität und somit kürzere Pläne und evtl. geringere Laufzeiten des Planers zu erhalten, liegt es somit nahe,

- schwächere hinreichende Bedingungen für die Äquivalenz von paralleler und sequentieller Ausführung als gleiches Verhalten von Operatoren bzw. deren Zulässigkeit in einer bestimmten Reihenfolge oder
- schwächere hinreichende Bedingungen für die Existenz einer sequentiellen Ausführung, die die Spezifikation erfüllt, als die Äquivalenz zu einer parallelen Ausführung

zu entwickeln.

Im ersten Fall ist zu beachten, dass gleiches Verhalten von Operatoren bzw. deren Zulässigkeit in einer gegebenen Reihenfolge nicht nur die Äquivalenz von paralleler und sequentieller Ausführung zur Folge hat, sondern sogar garantiert, dass jede Blockgrenze der Äquivalenz in der sequentiellen Planausführung auf eine Grenze zwischen zwei Zeitpunkten fällt. Es ist möglich, auch Kodierungen anzugeben, die Blockgrenzen an beliebigen Stellen in der sequentiellen Ausführung zulassen, indem man geeignete Hilfsvariable und Axiome über diese Hilfsvariablen einführt.

Im zweiten Fall sollte man im Unterschied zum bisherigen Vorgehen die Struktur der LTL-Formel bei der Formulierung der Bedingungen an parallel anwendbare Operatoren berücksichtigen. In Spezialfällen, etwa $\varphi = \mathbf{F}g$ für eine aussagenlogische Formel g, ist dies offenbar möglich, im genannten Fall etwa dadurch, dass man neben der Parallelitätskodierung aus Abschnitt 2.3.3.3 keine weiteren Bedingungen an parallele Operatoren stellt und dafür wie bei Erreichbarkeitszielen das Konjunktionsglied g_b zu der aussagenlogischen Übersetzung

hinzufügt. Auch eine Berücksichtigung der Polarität, mit der eine Variable in der Spezifikation vorkommt, könnte zur Verbesserung der angegebenen Kodierungen beitragen. Eine allgemeine Definition von Parallelitätsaxiomen in Abhängigkeit von der Struktur der LTL-Formel, etwa eine rekursive Definition wie in Abschnitt 2.4.3.1, erscheint dagegen schwierig und ist im Rahmen dieser Arbeit nicht zufriedenstellend gelungen.

Schließlich fehlt es noch an aussagekräftigen Vergleichen der Implementierung in SMLSAT-PLANNER mit bereits existierender Planungs- bzw. Modellprüfungssoftware (etwa NuSMV [Cimatti, Clarke, Giunchiglia und Roveri, 1999] für einen Vergleich mit einer bestehenden Implementierung von beschränkter symbolischer Modellprüfung). Auch wurden noch keine Vergleiche mit nicht-erfüllbarkeitsbasierten Planungs- oder Modellprüfungsalgorithmen angestellt.

Anhang A.

STRIPS-Version der Logistics-Domäne

```
(define (domain logistics-strips)
  (:requirements :strips)
  (:predicates  (OBJ ?obj) (TRUCK ?truck) (LOCATION ?loc) (AIRPLANE ?airplane)
                (CITY ?city) (AIRPORT ?airport) (at ?obj ?loc) (in ?obj1 ?obj2)
                (in-city ?obj ?city))

(:action LOAD-TRUCK
  :parameters (?obj ?truck ?loc)
  :precondition (and (OBJ ?obj) (TRUCK ?truck) (LOCATION ?loc)
                     (at ?truck ?loc) (at ?obj ?loc))
  :effect (and (not (at ?obj ?loc)) (in ?obj ?truck)))

(:action LOAD-AIRPLANE
  :parameters (?obj ?airplane ?loc)
  :precondition (and (OBJ ?obj) (AIRPLANE ?airplane) (LOCATION ?loc)
                     (at ?obj ?loc) (at ?airplane ?loc))
  :effect (and (not (at ?obj ?loc)) (in ?obj ?airplane)))

(:action UNLOAD-TRUCK
  :parameters (?obj ?truck ?loc)
  :precondition (and (OBJ ?obj) (TRUCK ?truck) (LOCATION ?loc)
                     (at ?truck ?loc) (in ?obj ?truck))
  :effect (and (not (in ?obj ?truck)) (at ?obj ?loc)))

(:action UNLOAD-AIRPLANE
  :parameters (?obj ?airplane ?loc)
  :precondition (and (OBJ ?obj) (AIRPLANE ?airplane) (LOCATION ?loc)
                     (in ?obj ?airplane) (at ?airplane ?loc))
  :effect (and (not (in ?obj ?airplane)) (at ?obj ?loc)))

(:action DRIVE-TRUCK
  :parameters (?truck ?loc-from ?loc-to ?city)
  :precondition (and (TRUCK ?truck) (LOCATION ?loc-from) (LOCATION ?loc-to)
                     (CITY ?city) (at ?truck ?loc-from)
                     (in-city ?loc-from ?city) (in-city ?loc-to ?city))
  :effect (and (not (at ?truck ?loc-from)) (at ?truck ?loc-to)))
```

```
(:action FLY-AIRPLANE
  :parameters (?airplane ?loc-from ?loc-to)
  :precondition (and (AIRPLANE ?airplane) (AIRPORT ?loc-from)
                     (AIRPORT ?loc-to) (at ?airplane ?loc-from))
  :effect (and (not (at ?airplane ?loc-from)) (at ?airplane ?loc-to)))
)
```

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[Bacchus und Ady 2001] Bacchus, Fahiem ; Ady, Michael: Planning with Resources and Concurrency: A Forward Chaining Approach. In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2001, S. 417–424

[Bacchus und Kabanza 2000] Bacchus, Fahiem ; Kabanza, Froduald: Using Temporal Logics to Express Search Control Knowledge for Planning. In: *Artificial Intelligence* 116 (2000), S. 123–191

[Biere u. a. 2003] Biere, Armin ; Cimatti, Alessandro ; Clarke, Edmund ; Strichman, Ofer ; Zhu, Yunshan: Bounded Model Checking. In: *Advances in Computers* 58 (2003)

[Biere u. a. 1999] Biere, Armin ; Cimatti, Alessandro ; Clarke, Edmund ; Zhu, Yunshan: Symbolic Model Checking without BDDs. In: *Lecture Notes in Computer Science* 1579 (1999), S. 193–207

[Blackburn u. a. 2001] Blackburn, Patrick ; de Rijke, Maarten ; Venema, Yde: *Modal Logic*. New York, NY, USA : Cambridge University Press, 2001

[Blum und Furst 1995] Blum, Avrim L. ; Furst, Merrick L.: Fast Planning Through Planning Graph Analysis. In: *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI 95)*, 1995, S. 1636–1642

[Bonet und Geffner 2001] Bonet, Blai ; Geffner, Hector: Planning as Heuristic Search. In: *Artificial Intelligence* (2001)

[Bylander 1994] Bylander, Tom: The Computational Complexity of Propositional STRIPS Planning. In: *Artificial Intelligence* 69 (1994), Nr. 1–2, S. 165–204

[Cimatti u. a. 1999] Cimatti, Alessandro ; Clarke, Edmund M. ; Giunchiglia, Fausto ; Roveri, Marco: NuSMV: a new Symbolic Model Verifier. In: Halbwachs, N. (Hrsg.) ; Peled, D. (Hrsg.): *Proceedings Eleventh Conference on Computer-Aided Verification (CAV'99)*. Trento, Italy : Springer-Verlag, July 1999 (Lecture Notes in Computer Science 1633), S. 495–499

[Clarke u. a. 2002] Clarke, Edmund M. ; Grumberg, Orna ; Peled, Doron A.: *Model Checking*. Cambridge, Massachusetts : The MIT Press, 2002

[DIMACS 1993] DIMACS: *Satisfiability Suggested Format*. 1993. – URL `http://www.satlib.org/Benchmarks/SAT/satformat.ps`

[Doherty und Kvarnström 2001] Doherty, Patrick ; Kvarnström, Jonas: TALplanner: A Temporal Logic Based Planner. In: *AI Magazine* 22 (2001), Nr. 3, S. 95–102

[Doherty u. a. 2001] Doherty, Patrick ; Kvarnström, Jonas ; Doherty, Patrick: TALplanner: A Temporal Logic Based Forward Chaining Planner. In: *Annals of Mathematics and Artificial Intelligence* 30 (2001), S. 119–169

[Ebbinghaus u. a. 1998] Ebbinghaus, Heinz-Dieter ; Flum, Jörg ; Thomas, Wolfgang: *Einführung in die mathematische Logik*. Heidelberg, Berlin : Spektrum Akademischer Verlag, 1998

[Edelkamp 2003] Edelkamp, Stefan: Limits and Possibilities of PDDL for Model Checking Software. In: *Workshop on the Competition: Impact Organization, Evaluation, Benchmarks, International Conference on Automated Planning and Scheduling (ICAPS)*. Trento, Italy, 2003, S. 53–63

[Efron und Tibshirani 1993] Efron, Bradley ; Tibshirani, Robert J.: *An introduction to the bootstrap*. New York : Chapman & Hall, 1993 (Monographs on statistics and applied probability 57)

[Gerevini und Long 2005] Gerevini, Alfonso ; Long, Derek: Plan Constraints and Preferences in PDDL3 / Department of Electronics for Automation, University of Brescia. August 2005. – Forschungsbericht

[Godefroid 1996] Godefroid, Patrice: *Partial-Order Methods for the Verification of Concurrent Systems: An Approach to the State-Explosion Problem*. Berlin, Heidelberg : Springer-Verlag, 1996 (Lecture Notes in Computer Science 1032)

[Kautz und Selman 1992] Kautz, Henry ; Selman, Bart: Planning as Satisfiability. In: *Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI'92)*, 1992, S. 359–363

[Kautz und Selman 1996] Kautz, Henry ; Selman, Bart: Pushing the Envelope: Planning, Propositional Logic, and Stochastic Search. In: Shrobe, Howard (Hrsg.) ; Senator, Ted (Hrsg.): *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*. Menlo Park, California : AAAI Press, 1996, S. 1194–1201

[Lamport 1983] Lamport, Leslie: What Good is Temporal Logic? In: Mason, R. E. A. (Hrsg.): *Information Processing 83*, Elsevier Science Publishers B. V. (North-Holland), 1983, S. 657–668

[Latvala u. a. 2004] Latvala, Timo ; Biere, Armin ; Heljanko, Keijo ; Junttila, Tommi: Simple Bounded LTL Model Checking. In: *Proceedings of the 5th International Conference on Formal Methods in Computer-Aided Design (FMCAD 2004)*. Austin, Texas, USA : Springer-Verlag, November 2004 (Lecture Notes in Computer Science 3312), S. 186–200

[Manna und Pnueli 1992] Manna, Zohar ; Pnueli, Amir: *The Temporal Logic of Reactive and Concurrent Systems: Specification*. New York, NY, USA : Springer-Verlag New York, Inc., 1992

[Manna und Pnueli 1995] Manna, Zohar ; Pnueli, Amir: *Temporal Verification of Reactive Systems: Safety*. New York, NY, USA : Springer-Verlag New York, Inc., 1995

[McDermott 1998] McDermott, Drew: *PDDL – The Planning Domain Definition Language*. 1998

[McDermott 2000] McDermott, Drew: The 1998 AI Planning Systems Competition. In: *AI Magazine* 21 (2000), Nr. 2, S. 35–55

[McDermott u. a. 1998] McDermott, Drew ; Anderson, Corin ; Köhler, Jana ; Selman, Bart ; Kautz, Henry: *AIPS98 Planning Competition Domänen und Ergebnisse*. 1998. – URL `ftp://ftp.cs.yale.edu/pub/mcdermott/papers/aipscomp.tar.gz`. – letzter Zugriff am 29. November 2005

[Nau u. a. 2004] Nau, Dana ; Ghallab, Malik ; Traverso, Paolo: *Automated Planning: Theory and Practice*. San Fransisco, CA : Morgan Kaufmann Publishers, 2004

[Papadimitriou 1994] Papadimitriou, Christos H.: *Computational Complexity*. Addison-Wesley, 1994

[Paulson 1996] Paulson, Lawrence C.: *ML for the Working Programmer*. New York, NY, USA : Cambridge University Press, 1996

[Peled und Wilke 1997] Peled, Doron ; Wilke, Thomas: Stutter-Invariant Temporal Properties are Expressible Without the Next-time Operator. In: *Inf. Process. Lett.* 63 (1997), Nr. 5, S. 243–246

[Rintanen 2004] Rintanen, Jussi: *Introduction to Automated Planning*. Vorlesungsmanuskript. 2004

[Rintanen 2005] Rintanen, Jussi: *Automated Planning: Algorithms and Complexity*, Albert-Ludwigs-Universität Freiburg, Habilitationsschrift, Juli 2005

[Rintanen u. a. 2004] Rintanen, Jussi ; Heljanko, Keijo ; Niemelä, Ilkka: Parallel Encodings of Classical Planning as Satisfiability. In: Alferes, J. J. (Hrsg.) ; Leite, J. (Hrsg.): *Logics in Artificial Intelligence: 9th European Conference, JELIA*, 2004 (Lecture Notes in Computer Science 3229), S. 307–319

[Rintanen u. a. 2005] Rintanen, Jussi ; Heljanko, Keijo ; Niemelä, Ilkka: Planning as Satisfiability: Parallel Plans and Algorithms for Plan Search / Albert-Ludwigs-Universität Freiburg, Institut für Informatik. 2005 (216). – Forschungsbericht

[Rintanen und Wölfl 2004] Rintanen, Jussi ; Wölfl, Stefan: *Modal Logics: Theory and Applications*. Vorlesungsmanuskript. 2004

[Russell und Norvig 2003] Russell, Stuart ; Norvig, Peter: *Artificial Intelligence – A Modern Approach*. Prentice Hall, 2003 (Series in Artificial Intelligence)

[Ryan 2004] Ryan, Lawrence: *Efficient Algorithms for Clause-Learning SAT Solvers*, Simon Fraser University, Master thesis, 2004

[Schöning 2000] Schöning, Uwe: *Logik für Informatiker*. Heidelberg, Berlin : Spektrum Akademischer Verlag, 2000

[Sistla und Clarke 1985] Sistla, A. P. ; Clarke, Edmund M.: The Complexity of Propositional Linear Temporal Logics. In: *Journal of the ACM* 32 (1985), Nr. 3, S. 733–749

[Tseitin 1968] Tseitin, G. S.: On the complexity of derivation in the propositional calculus. In: *Zapiski nauchnykh seminarov LOMI* 8 (1968), S. 234–259. – Englische Übersetzung: Consultants Bureau, N.Y., 1970, S. 115–125

[Weld und Etzioni 1994] Weld, Daniel ; Etzioni, Oren: The First Law of Robotics (a call to arms). In: *AAAI '94: Proceedings of the Twelfth National Conference on Artificial Intelligence (vol. 2)*. Menlo Park, CA, USA : American Association for Artificial Intelligence, 1994, S. 1042–1047