UNIVERSITÄT KOBLENZ · LANDAU
Fachbereich 4: Informatik

WeST People and Knowledge Networks
Institute for Web Science and Technologies

Inkonsistenzmessung zur Lösbarkeit von Planungsproblemen

Masterarbeit

zur Erlangung des Grades einer Master of Science (M.Sc.)
im Studiengang Computer Visualistik

vorgelegt von
Eva Kreckel

Erstgutachter: Prof. Dr. Steffen Staab
Institute for Web Science and Technologies

Zweitgutachter: PD Dr. Matthias Thimm
Institute for Web Science and Technologies

Koblenz, im Januar 2019

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

	Ja	Nein
Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden.	☐	☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	☐	☐
Der Text dieser Arbeit ist unter einer Creative Commons Lizenz verfügbar.	☐	☐
Der Quellcode ist unter einer Creative Commons Lizenz verfügbar.	☐	☐
Die erhobenen Daten sind unter einer Creative Commons Lizenz verfügbar.	☐	☐

.....

(Ort, Datum) (Unterschrift)

Inhaltsverzeichnis

Zusammenfassung

In dieser Arbeit wird die Messung der Inkonsistenz auf ein Planungsproblem angewandt. Dafür wird eine Funktion definiert, die das Planungsproblem auf eine Zahl zwischen 0 und 1 abbildet. Ist das Planungsproblem lösbar beziehungsweise konsistent, dann erhält es einen Wert von 0. Unlösbare Planungsprobleme werden hingegen als inkonsistent klassifiziert. Durch die Abbildung des Planungsproblems auf einen Zahlenbereich lassen sich zwei unlösbare Planungsprobleme miteinander vergleichen. Anhand des Inkonsistenzwertes können die Planungsprobleme also geordnet werden und so kann ein Planungsproblem inkonsistenter sein als das Andere. Die Maße zur Berechnung der Inkonsistenz bauen entweder darauf auf die Inkonsistenz schrittweise zu lösen oder die Berechnung findet auf den erreichbaren Zuständen des Planungsproblems statt.

Zusammenfassung

This thesis illustrates the computation of inconsistency in a planning problem. Thus a function like in the inconsistency measure will be used to map the planning problem onto a number ranging from 0 to 1. Is the planning problem solvable, also consistent, then it gets mapped to a 0. For the unsolvable case a planning problem can be represented as inconsistent. While comparing two inconsistent planning problems, one can be found more inconsistent than the other. Thus it can be measured as in inconsistency measure and ordered by their value. The measuring can be done by stepwise erasement of the inconsistency or by directly calculating on the outcome of the search from the planning problem.

1 Einführung

Planungsprobleme spielen sowohl in der Informatik als auch im alltäglichen Leben eine große Rolle, zum Beispiel in Navigationssystemen, in Computerspielen, Robotern und vielen weiteren Bereichen. In lösbaren Planungsproblemen wird oft die Lösung gesucht, die das Planungsproblem am schnellsten oder am kostengünstigsten löst. Bei der Navigation in einem Straßennetz ist dies die schnellste Verbindung um von Punkt A zu Punkt B zu gelangen. Sei dazu der aktuelle Standort München und der Zielort Berlin. Dann wird erwartet, dass das Navigationssystem die Route auswählt, die am wenigsten Zeit benötigt oder von den Kilometern her am geringsten ist. Eine Route über Frankfurt kann zwar auch dazu führen den Zielort Berlin zu erreichen, jedoch ist dies ein Umweg für die Strecke München-Berlin.

Genauso wie die Lösungen in lösbaren Planungsproblemen gemäß der Anwendung sortiert werden können, gibt es intuitiv eine Ordnung in unlösbaren Planungsproblemen. Sei zum Beispiel ein Roboter gegeben, der Objekte zwischen Standorten bewegen soll. Seien dazu einige angeforderte Objekte nicht verfügbar und Standorte auf Grund von Hindernissen nicht erreichbar. So ein Planungsproblem ist unlösbar, da das Ziel nicht vollständig erreicht werden kann. Zwar können Teile des Ziels erreicht werden, jedoch ist ein Planungsproblem erst lösbar, wenn das komplette Ziel erreicht werden kann. Besteht das Ziel aus 6 Objekten an einem zugewiesenen Ort, dann ist ein Planungsproblem, welches zum Beispiel nur 2 Objekte an die richtigen Standorte bewegen kann, unlösbarer als ein Planungsproblem, welches 5 der 6 Objekte an den richtigen Standort bringen kann. Dadurch, dass das erste Planungsproblem unlösbarer ist, ist es möglich eine Ordnung zwischen den beiden Planungsproblemen zu definieren.

Neben den Planungsproblemen gibt es in der künstlichen Intelligenz die Inkonsistenzmessung. Dabei steht eine Wissensbasis zu Verfügung, die das Wissen der realen Welt widerspiegelt. Da sich das Wissen der realen Welt fortlaufend verändert, wandelt sich die Wissensbasis ebenfalls über die Zeit. Soll neues Wissen der Wissensbasis hinzugefügt werden, dann kann es passieren, dass das neue Wissen konsistent oder inkonsistent zur Wissensbasis ist. Ist das neue Wissen konsistent zur Wissensbasis, dann erweitert es die Wissensbasis mit dem neuen Wissen. Ruft das neue Wissen jedoch Inkonsistenzen innerhalb einer Wissensbasis auf, dann wird die ganze Wissensbasis inkonsistent. Dies ist vergleichbar mit Aussagen von mehreren Personen. Beinhalte die Wissensbasis dazu Aussagen über die Farbe eines Autos. Behaupte die ersten Person das Auto ist rot. Anschließend behauptet eine zweite Person, dass das Auto blau sei. Da ein Auto normalerweise einfarbig ist, sind die Aussagen inkonsistent. Die Wissensbasis kann beide Aussagen nicht zu gleichen Zeit erfüllen. Dabei können Wissensbasen entstehen, die inkonsistenter sind als Andere. Um diese Inkonsistenz zu messen, werden Inkonsistenzmaße auf eine Wissensbasis angewandt. Die Maße bilden die Wissensbasen dabei auf eine Zahl ab. Je inkonsistenter eine Wissensbasis, desto größer ist der Wert des Inkonsistenzmaßes. Dadurch lässt sich eine Ordnung zwischen Wissensbasen herstellen.

Sowohl die Wissensbasen aus der Inkonsistenzmessung als auch Planungsprobleme können anhand ihrer Inkonsistenz beziehungsweise Unlösbarkeit geordnet werden. Dadurch, dass Inkonsistenzmessungen auf eine Zahl abbilden, ist es möglich Wissensbasen nach ihrer Schwere der Inkonsistenz zu ordnen oder die Schwere der Inkonsistenz mit anderen Wissensbasen zu vergleichen. So soll es auch möglich sein Planungsprobleme anhand der Schwere ihrer Unlösbarkeit zu ordnen und zu vergleichen. Dafür bietet sich die Inkonsistenzmessung mit einer Vielzahl an Maßen an. Durch die Anwendung der Inkonsistenzmessung auf ein Planungsproblem kann dieses ebenfalls auf eine Zahl abgebildet werden. Somit können Planungsprobleme bezüglich ihrer Unlösbarkeit geordnet und verglichen werden.

Zunächst geben die Kapitel 2 und 3 eine Einführung in die Grundlagen der Planungsprobleme und der Inkonsistenzmessung. Anschließend wird in Kapitel 4 die Inkonsistenzmessung auf die Planungsproblematik definiert. Darauf aufbauend werden vier Inkonsistenzmaße für Planungsprobleme entwickelt. In Kapitel 5 wird anschließend analysiert welche Inkonsistenzmaße der Planungsprobleme die definierten Eigenschaften erfüllen. Weiterhin zeigt Kapitel 6 eine Implementation der vier Maße aus Kapitel 4. Die Diskussion der Ergebnisse findet schließlich in Kapitel 7 statt.

2 Grundlagen der Planungsprobleme

Ein Planungsproblem besteht aus einem Startpunkt und einem gewünschten Ziel. Vergleichbar mit der Navigation im Auto ist der Startpunkt der aktuelle Standort und das Ziel der gewünschte Zielort. Dazu können Operationen die Welt verändern. Dabei soll das Ausführen von Operationen zum Ziel führen. Bei der Navigation zum Beispiel kann eine Operation *driveTo* existieren, die die Welt dahingehend ändert, dass der Standort wechselt. Wird die Operation mehrmals an unterschiedlichen Standorten ausgeführt, dann kann der gewünschte Zielort erreicht werden.

Ist das Ziel durch die gegebenen Operationen erreichbar, dann ist das Planungsproblem lösbar. Jedoch gibt es auch Planungsprobleme, bei denen das Ziel nicht erreicht werden kann. Sei zum Beispiel ein Roboter gegeben, der von seinem aktuellen Standpunkt zu einem Zielpunkt fahren soll. Ist der Weg zum Ziel blockiert, dann ist das Ziel nicht erreichbar. Damit ist das Planungsproblem unlösbar.

Diese Arbeit betrachtet nur Planungsprobleme in Zustandsräumen. Dafür gibt das Kapitel 2.1 zunächst eine Einführung in den Zustandsraum. Anschließend werden die Sprachen STRIPS 2.1.1 und PDDL 2.1.2 erläutert, die Planungsprobleme in einem Zustandsraum darstellen. Die Darstellung erfolgt in dieser Arbeit nach dem Kapitel 2.1.2 mit PDDL in vereinfachter Schreibweise. Zuletzt behandelt das Kapitel 2.1.3 die Lösbarkeit von Planungsproblemen in Zustandsräumen.

2.1 Zustandsraum

Sei $\mathcal{L}$ eine Sprache, die aus endlich vielen Variablen und Konstanten besteht. Dann ist ein Zustand s, aus der Menge aller Zustände S, eine Menge an Atomen aus der Sprache $\mathcal{L}$. Wird zum Beispiel ein Lichtschalter und eine Lampe repräsentiert, dann könnte ein Zustand wie folgt aussehen: $s = \{lightSwitch(on), lamp(on)\}$. Der Lichtschalter ist also betätigt und die Lampe ist ebenfalls an. Ein Zustand s_2 in dem der Lichtschalter an ist, jedoch die Lampe nicht, kann durch $s_2 = \{lightSwitch(on), lamp(off)\}$ dargestellt werden.

Ein Atom α ist in einem Zustand s enthalten, wenn α ein Element des Zustandes s ist: $\alpha \in s$. Dazu erfüllt eine Menge an Atomen g den Zustand s, wenn alle Atome aus g in s enthalten sind.

Um die Welt von einem Zustand s_n in einen neuen Zustand s_{n+1} überführen zu können, besitzt ein Planungsproblem Π Operationen O. Jede Operation $o = \langle c, e \rangle \in O$ hat eine Vorbedingung und Effekte. Eine Operation o ist ausführbar, wenn die Vorbedingung c erfüllt ist, also wenn alle Atome der Vorbedingung c im aktuellen Zustand s_n vorkommen. Anschließend wird der aktuelle Zustand mit den Effekten e aus der Operation o verändert. Daraus lässt sich ein Planungsproblem ableiten.

Definition 1 (Planungsproblem)**.** *Ein Planungsproblem $\Pi = \langle O, s_I, g \rangle$ besteht aus einer Menge von Operationen O, einem Startzustand s_I und einem Ziel g.*

Beispiel 1.
```
turnOff:
c: lightSwitch(on)
e: lightSwitch(off), lamp(off),
   not lightSwitch(on), not lamp(on)
```
Listing 1: Operation *turnOff*

Sei die Operation o aus Listing 1 gegeben, die den Lichtschalter ausstellt. Die Vorbedingung besagt, dass der Lichtschalter vorher an sein muss. Ist dies der Fall, dann wird der Zustand $s_n = \{lightSwitch(on), lamp(on)\}$ so verändert, dass nachfolgend der Lichtschalter ausgeschaltet ist und die Lampe nicht mehr leuchtet, also $s_{n+1} = \{lightSwitch(off), lamp(off)\}$.

Generell kann eine Zustandsänderung $\gamma : S \times O \to S$ durch die Eingabe eines Zustandes und einer Operation definiert werden [11].

Definition 2 (Zustandsänderung γ)**.** *Die Zustandsänderung $\gamma : S \times O \to S$ überführt einen aktuellen Zustand s_n mit einer Operation $o_n = \langle c, e \rangle$ in einen neuen Zustand*

$$s_{n+1} : \gamma(s_n, o_n) = s_{n+1}, \text{ wenn } s_n \text{ die Vorbedingung c erfüllt}$$

Kann ein Zustand das Ziel g eines Planungsproblems erfüllen, dann ist ein Zielzustand erreicht.

Definition 3 (Zielzustand)**.** *Ein Zielzustand s_G ist dann gegeben, wenn ein Zustand s alle Atome des Ziels g des Planungsproblems $\Pi = \langle O, s_I, g \rangle$ erfüllt.*

In dieser Arbeit werden zwei Sprachen, STRIPS und PDDL, vorgestellt, die ein Planungsproblem Π innerhalb des Zustandsraumes darstellen. PDDL baut auf STRIPS auf und erweitert diese Sprache. Daher wird zunächst eine Einführung in die STRIPS-Sprache gegeben und anschließend in PDDL.

2.1.1 STRIPS

Laut [3] besteht ein STRIPS-Problem aus einem Tupel mit drei Einträgen $\Pi = \langle O, s_I, g \rangle$. Dabei beschreibt s_I den initialen Zustand und g entspricht dem Ziel. Weiterhin steht O für eine endliche Anzahl an Operationen, die auf die Zustände wirken und eine Transformation in einen neuen Zustand ermöglichen können. Eine Operation $o \in O$ selbst besteht aus einem Tupel $o = \langle c, a, d \rangle$. Dabei sind c, a und d jeweils Mengen, die aus Atomen bestehen. Die Menge c beschreibt die Vorbedingung, die gelten muss, damit ein Zustand in einen neuen Zustand überführt werden kann. Sind alle Atome aus der Vorbedingung erfüllt, dann treten die Effekte der Mengen a und d ein. Zunächst werden dazu alle Atome aus dem aktuellen Zustand gemäß der *delete*-Menge d entfernt und anschließend die Atome aus der *add*-Menge a dem Zustand hinzugefügt. Dadurch ist ein neuer Zustand entstanden $s_{n+1} = (s_n \setminus d) \cup a$ [5]. Alle weiteren Atome, die sich im alten Zustand befanden und durch die Operation nicht beeinflusst wurden, bleiben unverändert.

Beispiel 2. *Sei dazu ein Supermarkt gegeben, der verschiedene Früchte anbieten möchte. Bananen können aus Ecuador, Orangen aus Spanien und Äpfel aus Deutschland geliefert werden. Das Problem wird als $\Pi = \langle O, s_I, g \rangle$ beschrieben. Die Früchte gelangen über Transportwege in den Supermarkt. Seien dazu die Orte durch market, ecuador, spain und germany und die Früchte durch banana, orange, apple gegeben. Der Startzustand beschreibt das Vorfinden der Obstsorten in den Herkunftsländern und das Ziel ist es diese Obstsorten im Supermarkt anzubieten.*

Generell kann die Welt durch die zwei Aussagen at und transfer, wie in Listing 2, beschrieben werden. Die erste Aussage at beschreibt den Aufenthaltsort der jeweiligen Frucht. Dazu gibt transfer Auskunft über die Möglichkeiten einer Transportverbindung zwischen den Standorten FROM und TO an.

```
1 at(PLACE,FRUIT)
2 transfer(FROM,TO)
```

Listing 2: Aussagen des Supermarkt-Beispiels

Weiterhin gibt es die Operation transport, die den Transport der Frucht zwischen zwei Standorten beschreibt. Die Operation ist nur möglich, wenn sich die Frucht am Standort FROM befindet und eine Transportverbindung zwischen den beiden Standpunkten verfügbar ist, siehe dazu Listing 3.

```
1 transport(FROM,TO,FRUIT):
2     c: at(FROM,FRUIT),transfer(FROM,TO)
3     a: at(TO,FRUIT)
```

```
4       d: at(FROM,FRUIT)
```

Listing 3: Operation *transport* des Supermarkt-Beispiels

Ist die Vorbedingung erfüllt, dann wird zunächst at(FROM,FRUIT) gelöscht und die Frucht ist nicht mehr am Standort FROM verfügbar. Anschließend werden die Atome aus der add-Menge, hier nur at(TO,FRUIT), dem Zustand hinzugefügt.

In Listing 4 ist das Problem in STRIPS zusammen gefasst. Dazu ist ein Startzustand und ein Ziel angegeben.

```
1 Π = ⟨O, s_I, g⟩
2 O={transport(FROM,TO,FRUIT)}
3 s_I={at(ecuador,banana),at(germany,apply),
4       at(spain,orange),transfer(ecuador,germany),
5       transfer(germany,market),transfer(spain,germany)}
6 g={at(market,banana),at(market,apple),at(market,orange)}
```

Listing 4: Supermarkt-Beispiel in STRIPS

Um das Ziel zu erreichen, kann die Abfolge der Operationen aus Listing 5 verwendet werden. Beginnend beim Startzustand erfüllt der erzeugte Zustand nach dem Ausführen der letzten Operation das Ziel des Planungsproblems Π*.*

```
1 transport(ecuador,germany,banana)
2 transport(germany, market, banana)
3 transport(germany, market, apple)
4 transport(spain, germany, orange)
5 transport(germany, market, orange)
```

Listing 5: Abfolge von Operationen um das Ziel des Supermarktbeispiels zu erreichen

2.1.2 PDDL2.1

PDDL (Planning Domain Definition Language) [6] [7] ist eine Sprache, die auf STRIPS basiert und diese erweitert. Dabei wird ein Planungsproblem in zwei Teile gegliedert, die Domäne und die Problemspezifikation. Die Domäne beschreibt die Aktionen und die Art der Atome, wobei die Problemspezifikation die Atome, den Startzustand und das Ziel spezifiziert. Somit ist es möglich verschiedene Planungsprobleme zu erzeugen, in dem unterschiedliche Problemspezifikation auf die gleiche Domäne angewandt wird. Weiterhin existiert die delete-Menge aus STRIPS nicht mehr. Dafür ist es möglich Atome mit *not* zu verneinen [7].

Die Domäne, durch "(:domain ...)" gekennzeichnet, beschreibt die Anforderungen für das Planungsproblem "(:requirements ..)" , Objekttypen "(:types ...)" , Prädikate "(:predicates ...)" und Operationen "(:action ...)". Dabei beschreiben die Objekttypen welche Arten von Typen innerhalb der Domäne auftreten können. Im Beispiel des Supermarktes wären dies die Früchte und die Orte. Die Prädikate sind im

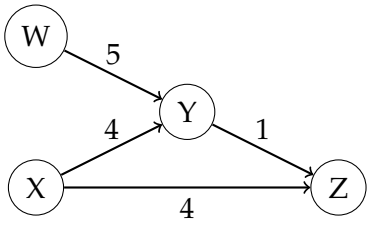

Abbildung 1: Produktionsabhängigkeiten

Supermarktbeispiel at und transfer. Operationen teilen sich in drei gesonderte Spezifikationen auf. Zunächst werden mit ":parameters(...)" die Eingabeparameter der Operation vorgegeben. Die Vorbedingung zur Ausführung der Aktion beschreibt ":precondition(...)". Ist diese wahr, tritt der Effekt der Operation ":effect (...)" ein.

Je nach Anforderung in *:requirements* erweitert sich die Sprache PDDL. Die Basis stellt *:strips* dar. Anschließend kann PDDL mit bedingten Effekten *:conditional-effects* erweitert werden. Damit kann zum Beispiel ein Effekt nur ausgeführt werden, wenn die spezifizierte Bedingung im Zustand erfüllt ist. Dies kann als eine zweite Vorbedingung [12] im Effekt angesehen werden. Weiterhin lassen sich Objekttypen mit der Anforderung *:typing* hinzufügen. Eine weitere Anforderung kann *:fluents* sein, die arithmetische Operationen bereit stellt [6].

Die zweite Spezifikation, die Problemspezifikation, beinhaltet Objekte "(:objects ...)", einen Initialzustand "(:init ...)" und ein Ziel "(:goal ...)". Die Objekte können nur von einem Typ sein, der bereits in der Domäne spezifiziert wurde.

Beispiel 3. *Sei dazu ein Beispiel einer Firma gegeben, die vier Produkte, also W, X, Y und Z, herstellt. Diese hängen jeweils von einander ab, siehe Abbildung 1. Die Produkte W und X können unabhängig voneinander produziert werden. Hingegen hängt das Produkt Y von W und X ab und Produkt Z von Y und X. Dazu benötigt es fünf mal das Produkt W und vier mal das Produkt X um Y herzustellen. Für das Produkt Z wird X vier mal und Produkt Y einmal benötigt. Listing 6 zeigt das Beispiel in STRIPS ohne die benötigte Anzahl des Produktes mit einzubeziehen.*

```
1 produceY():
2     C: produced(W), produced(X)
3     A: produced(Y)
4     D: produced(W), produced(X)
5 produceZ():
6     C: produced(Y), produced(X)
7     A: produced(Z)
8     D: produced(Y), produced(X)
```

Listing 6: Produktionsfirma mit STRIPS-Operationen

Mit PDDL ist es möglich, die Abhängigkeiten wie in der Abbildung 1 darzustellen. Listing 7 besitzt 4 Zahlvariablen, für jedes Produkt eine Variable. Anschließend kann in den

Effekten die entsprechende Variable um 1 erhöht werden. In PDDL sieht das Firmen-Beispiel dann wie folgt aus.

```
(define (company-example)
   (:requirements :fluents)
   (:domain company)
   (:types product)
   (:functors
      (producedW)
      (producedX)
      (producedY)
      (producedZ)
   )
   (:action produceW
      :parameters ()
      :precondition ()
      :effect ( (increase(producedW) 1) )
   )
   (:action produceX
      :parameters ()
      :precondition ()
      :effect ( (increase(producedX) 1) )
   )
   (:action produceY
      :parameters (?w ?x - product)
      :precondition (and (>= (producedW) 5)
         (>= (producedX) 4))
      :effect (and ((increase(producedY) 1)
            and ((decrease(producedW) 5)
                (decrease(producedX) 4))))
   )
   (:action produceZ
      :parameters (?x ?y - product)
      :precondition (and (>= (producedX) 4)
         (>= (producedY) 1))
      :effect (and ((increase(producedZ) 1)
            and ((decrease(producedY) 1)
                (decrease(producedX) 4))))
   )
)
```

Listing 7: Produktionsfirma-Beispiel in PDDL als Domänen-spezifikation

Dazu ist hier der Startzustand leer, da noch keine Produkte vorliegen. Jedoch müssen die Zahlvariablen auf 0 gesetzt werden. Das Ziel ist es Produkt Z zu produzieren. Listing 8

zeigt die Problemspezifikation.

```
(define (company-example)
   (:domain company)
   (:objects
      w x y z - product
   )
   (:init
      (= (producedW) 0)
      (= (producedX) 0)
      (= (producedY) 0)
      (= (producedZ) 0)
   )
   (:goal (= (producedZ) 1)
   )
)
```

Listing 8: Produktionsfirma-Beispiel in PDDL als Problemspezifikation

Eine Lösung des Planungsproblems ist es zunächst fünfmal die Operation produceW und achtmal die Operation produceX auszuführen. Anschließend wird die Operation produceY ausgeführt. Zum Schluss kann mit der Operation produceZ das Ziel erreicht werden.

Zur besseren Lesbarkeit der PDDL-Spezifikationen werden nachfolgend die Beispiele in einer vereinfachten PDDL-Schreibweise notiert. Die Parameter der Operation stehen in Klammern hinter dem Namen der Operation. Die Vorbedingung, mit " C: " gekennzeichnet, und die Effekte " E: " werden ähnlich wie in der STRIPS-Schreibweise aufgeführt. Vergleiche sind der Form "x VERGLEICH y" angegeben. Zuweisungen werden durch ein " = " dargestellt.

Beispiel 4. *Listing 9 zeigt die Operation producedY in vereinfachter PDDL-Schreibweise.*

```
produceY(w, x):
   C: producedW >= 5 and producedX >=4
   E: producedY+=1 and producedW-=5 and producedX-=4
```

Listing 9: Operation *producedY* in vereinfachter PDDL-Schreibweise

W und x sind die Parameter der Operation produceY. Die Vorbedingung vergleicht, ob producedW größer gleich 5 und producedX größer gleich 4 ist. Der Effekt besteht aus 3 Zuweisung, in dem producedY um eins erhöht wird, producedW um 5 und producedX um 4 subtrahiert wird.

2.1.3 Lösbarkeit in Planungsproblemen

Bei der in Kapitel 2.1 definierten Zustandsänderung γ entsteht wieder ein Zustand. Auf den neuen Zustand lässt sich wieder die Zustandsänderung γ ausführen. Werden mehrere Zustandsänderungen hintereinander ausgeführt, dann entsteht daraus

ein Pfad $p \in P$. Dieser beschreibt eine Abfolge von Operationen $p = \langle o_1, \ldots, o_k \rangle$. Durch diesen Pfad p wird ein Zustand s_n mit k Operationen in einen Zustand s_{n+k} verändert, wobei $k > 0$ ist.

Definition 4 (Pfad). *Ein Pfad $p \in P$ besteht aus einer Abfolge von Operationen $p = \langle o_1, \ldots, o_k \rangle$, die zu einer Abfolge von Zustandsänderungen γ führen $\gamma : S \times P \to S$.*

$$\gamma(s_n, p) = \begin{cases} s_n & \textit{wenn } k = 0 \\ \gamma(\gamma(s_n, o_1), \langle o_2, \ldots, o_k \rangle) & \textit{wenn } k > 0 \textit{ und } o_1 \textit{ bis } o_k \textit{ anwendbar} \\ undefiniert & \textit{sonst} \end{cases} \quad (1)$$

Kann mindestens ein Pfad vom Startzustand s_I eines Planungsproblems bis zu einem Zielzustand s_G gefunden werden, dann ist das Planungsproblem lösbar.

Definition 5 (Lösbarkeit eines Planungsproblems). *Sei das Planungsproblem $\Pi = \langle O, s_I, g \rangle$ gegeben. Eine Lösung des Problems ist ein Pfad $p \in P$ mit $\gamma(s_I, p) = s_G$.*

Beispiel 5. *Wird das Supermarktbeispiel aus Kapitel 2.1.1 betrachtet, dann ist ein möglicher Pfad in Listing 10 gegeben.*

```
1 transport(ecuador,germany,banana)
2 transport(germany, market, banana)
3 transport(germany, market, apple)
4 transport(spain, germany, orange)
5 transport(germany, market, orange)
```

Listing 10: Pfad zu einem Zielzustand des Supermarkt-Beispiels

Nach dem Ausführen der ersten zwei Operationen auf den Startzustand verändert sich der Zustand zu $s_2 = \{at(market, banana), at(germany, apple), at(spain, orange)\}$. Nach dem Ausführen der weiteren Operationen ist ein Zielzustand erreicht. Insgesamt gibt es sechs mögliche Pfade, die zu einem Zielzustand führen. Das Planungsproblem ist als lösbar verifiziert, wenn es einen Pfad gibt, die in einem Zielzustand endet.

Seien nun auf Grund von einer Knappheit keine Bananen mehr in Ecuador vorhanden, so ändert sich das Problem wie folgt ab $\Pi' = \langle O, s_{I2}, g \rangle$. Der neue Startzustand ist dabei $s_{I2} = \{at(germany, apple), at(spain, orange)\}$. Sowohl das Ziel als auch die Operationen ändern sich zum vorherigen Beispiel des Planungsproblems Π nicht. Dadurch, dass das Atom at(ecuador, banana) in keinem Zustand des Planungsproblems Π_2 vorhanden ist, kann mit der Operation transport niemals die Frucht Banane in den Supermarkt gelangen. Dadurch ist es nicht möglich das Ziel g auf einem Pfad zu erreichen. Das Beispiel Π_2 ist damit nicht lösbar.

3 Grundlagen der Inkonsistenzmessung

Die Inkonsistenzmessung berechnet für eine Wissensbasis k die Schwere der Inkonsistenz innerhalb der Wissensbasis k. Zum Beispiel ist eine Wissensbasis $k1 =$

$\{\neg a, a\}$ inkonsistent. Es ist nicht möglich $nega$ und a gleichzeitig zu erfüllen. Hingegen ist eine Wissensbasis $k2 = \{a, b\}$ konsistent, da es möglich ist a und b so zu belegen, dass die Wissensbasis wahr ist.

Je mehr inkonsistentes Wissen eine Wissensbasis aufweist, desto größer ist der Wert der Messung. Dabei gibt es verschiedene Maße, die die Inkonsistenz auf unterschiedliche Weise berechnen. Generell besteht ein Maß aus einer Funktion, die eine Wissensbasis $k \in K$ auf eine positive Zahl größer gleich 0 abbildet: $I : K \to \mathbb{R}^{\geq 0}$. Ist die Wissensbasis konsistent, so wie $k2$, dann bildet das Maß auf eine 0 ab. Die Wissensbasis $k1$ wird somit auf einen Wert größer 0 abgebildet.

Dieses Kapitel beschreibt die Grundlagen der Inkonsistenzmessung, die im Kapitel 4 aufgegriffen wird. Da die Inkonsistenzmessung auf der Aussagenlogik beruht, werden zunächst die Grundlagen der Aussagenlogik eingeführt und anschließend für die Definitionen der Eigenschaften von Inkonsistenzmaßen verwendet. Kapitel 3.3 führt abschließend einige Inkonsistenzmaße auf.

3.1 Grundlagen der Aussagenlogik

Sei $\mathcal{P}$ eine Menge, die aus endlichen aussagenlogischen Symbolen $a, b, \ldots \in \mathcal{P}$ besteht. Dann definiert $\mathcal{L}(\mathcal{P})$ die dazugehörige aussagenlogische Sprache. Weiterhin können Symbole mit logischen Verknüpfungen wie $\neg$, $\wedge$ und $\vee$ zu Formeln zusammen gesetzt werden. Die Formeln werden nachfolgend mit griechischen Buchstaben wie α, β beschrieben. Eine Wissensbasis k ist dann eine endliche Menge an Formeln aus der Sprache $\mathcal{L}(\mathcal{P})$.

Definition 6 (Wissensbasis)**.** *Eine Wissensbasis k ist eine endliche Menge an Formeln aus der Sprache $\mathcal{L}(\mathcal{P})$.*

$$k \subseteq \mathcal{L}(\mathcal{P})$$

Eine Interpretation $\mathcal{W} : \mathcal{P} \to \{true, false\}$ bildet ein Symbol oder eine Formel auf einen Wert true oder false ab. Weiterhin ist ein Atom $a \in \mathcal{P}$ einer Interpretation ω erfüllt ($\omega \models a$), wenn ω das Atom a auf true abbildet: $\omega(a) = true$. Diese Interpretation wird auch ein Model des Atoms a genannt. Eine Wissensbasis ist konsistent, wenn es mindestens eine Interpretation gibt, die alle Formeln der Wissensbasis erfüllen.

Sei weiterhin $MOD(\phi)$ die Menge an Modellen für eine Formel oder eine Menge von Formeln ϕ: $MOD(\phi) = \{\omega \in \mathcal{W} \mid \omega \models \phi\}$. Kann kein Modell für ϕ gefunden werden, also $MOD(\phi) = \emptyset$, dann wird dies notiert als: $\phi \models \bot$. Dann gilt ϕ als inkonsistent [17]. Daraus lässt sich eine inkonsistente Wissensbasis ableiten.

Definition 7 (Inkonsistente Wissensbasis)**.** *Eine Wissensbasis k ist inkonsistent, wenn es kein Model gibt, welches die Wissensbasis erfüllt.*

$$k \models \bot$$

Ist eine Wissensbasis k inkonsistent, dann gibt es mindestens eine minimal inkonsistente Menge.

Definition 8 (Minimal inkonsistente Menge)**.** *Sei k eine inkonsistente Wissensbasis. Dann ist m eine minimal inkonsistente Menge der Wissensbasis k, wenn*

$$m \subseteq k$$
$$m \models \bot$$
$$\forall m' \subset m : m' \not\models \bot$$

Diese Menge m ist eine Teilmenge der Wissensbasis k und kein Modell der Menge m kann erfüllt werden. Ebenfalls gibt es keine weitere Menge m', die eine echte Teilmenge von m ist und deren Modell ebenfalls nicht erfüllt ist [9], [4], [17]. Sei $MI(k)$ die Menge aller minimal inkonsistenter Mengen der Wissensbasis k.

Innerhalb einer Wissensbasis k können Formeln existieren, die in keiner minimalen inkonsistenten Menge der Wissensbasis k enthalten sind. Diese Formeln werden freie Formeln der Wissensbasis k genannt [9].

Definition 9 (Freie Formel)**.** *Eine freie Formel der Wissensbasis k ist in keiner Menge der minimal inkonsistenten Mengen $MI(k)$ der Wissensbasis k enthalten.*

3.2 Eigenschaften von Inkonsistenzmaßen

Sei K eine Wissensbasis, α und β logische Formeln und die Funktion $I : K \rightarrow \mathbb{R}^{>0}$ als ein Maß zur Berechnung der Inkonsistenz gegeben [10], [4], [17]. Folgende Eigenschaften sollten für Inkonsistenzmaße erfüllt sein.

- **Konsistenz:** Der Funktionswert ist 0, wenn die Wissensbasis konsistent ist: $I(K) = 0$, wenn K konsistent ist.
- **Monotonie:** Wenn die Wissensbasis K zur Wissensbasis K' wächst, kann die Inkonsistenz nicht kleiner werden: Wenn $K \subseteq K'$, dann gilt $I(K) \leq I(K')$.
- **Normalisierung:** Der Inkonsistenzwert liegt zwischen 0 und 1: $0 \leq I(K) \leq 1$.
- **Unabhängigkeit von freien Formeln:** Das Löschen einer freien Formel α verändert die Inkonsistenz nicht: $I(K) = I(K \backslash \{\alpha\})$
- **Dominanz:** Logisch stärkere Formeln, die konsistent sind, führen potenziell zu einer größeren Inkonsistenz: Wenn $\alpha \models \beta$, dann gilt $I(K \cup \{\alpha\}) \geq I(K \cup \{\beta\})$

Die Normalisierung wird nicht als eine erforderliche Eigenschaft angesehen. Jedoch ist sie zur Vergleichbarkeit von Inkonsistenzwerten nützlich.

3.3 Inkonsistenzmaße

In [9] werden Maße vorgestellt, die die Inkonsistenz in Wissensbasen anhand von minimal inkonsistenten Mengen berechnet. Da es intuitiv erscheint eine Inkonsistenz mit minimal inkonsistenen Mengen zu berechnen, definiert [9] das Inkonsistenzmaß I_{MI} anhand der Anzahl an minimal inkonsistenten Mengen. Hier ist die Idee, dass der Inkonsistenzwert ansteigt, je mehr minimal inkonsistente Mengen in einer Wissensbasis existieren [16].

Definition 10 (Inkonsistenzmaß I_{MI})**.** *Das Inkonsistenzmaß I_{MI} entspricht der Anzahl an minimal inkonsistenten Mengen einer Wissensbasis k:*

$$I_{MI}(k) =\mid MI(k) \mid$$

Beispiel 6. *Sei $k1 = \{\neg a, a, a \vee b\}$, $k2 = \{\neg a, a, a \wedge b, d\}$ und $k3 = \{\neg a, a, a \wedge b, a \vee \neg b\}$ gegeben.*

Dann berechnet das Maß I_{MI} die Inkonsistenzwerte wie folgt:

$$\begin{aligned} I_{MI}(k1) &= 1 \\ I_{MI}(k2) &= 2 \\ I_{MI}(k3) &= 2 \end{aligned} \tag{2}$$

Die minimal inkonsistenten Mengen der Wissensbasen $k2$ und $k3$ sind: $\{\neg a, a\}$ und $\{\neg a, a \wedge b\}$. Die Wissensbasis $k1$ besitzt als minimal inkonsistente Menge nur $\{\neg a, a\}$.

Dieses Maß sagt jedoch nichts über die Größe der minimal inkonsistenten Mengen aus oder über die Überlappung von Formeln innerhalb der minimal inkonsistenten Mengen $MI(k)$ [10].

Anstatt die Berechnung auf den minimal inkonsistente Mengen auszuführen, können ebenso die maximal konsistenten Mengen zur Berechnung verwendet werden. Die maximal konsistente Menge m einer Wissensbasis k ist eine Teilmenge der Wissensbasis k, für die es ein Modell gibt. Weiterhin gilt für alle Formeln ϕ, die nicht in der Menge m enthalten sind, dass es für die Vereinigungsmenge von m und ϕ kein Modell gibt [4], [17].

Definition 11 (Maximal konsistente Menge)**.** *Sei k eine inkonsistente Wissensbasis. Dann ist m eine maximal konsistente Menge der Wissensbasis k, wenn*

$$\begin{gathered} m \subseteq k \\ m \nvDash \bot \\ \forall \phi \in k \backslash m : m \cup \{\phi\} \models \bot \end{gathered}$$

Die Menge an maximal konsistenter Mengen wird mit $MC(k)$ notiert. Sei weiterhin die Menge an selbst-widersprechenden Formeln $SC(k)$ einer Wissensbasis k gegeben. Eine Formel ϕ einer Wissensbasis k ist selbst-widersprechend, wenn es kein Modell für die Formel ϕ gibt.

Definition 12 (Selbst-widersprechende Formel)**.** *Sei ϕ eine selbst-widersprechende Formel in einer Wissensbasis k. Dann gilt:*

$$\phi \models \bot$$

Daraus lässt sich ein Inkonsistenzmaß I_{MC} [8], [17] ableiten, welches auf Basis der maximal konsistenten Mengen und selbst-widersprechenden Formeln aufgebaut ist.

Definition 13 (Inkonsistenzmaß I_{MC})**.** *Das Maß I_{MC} berechnet sich aus der Anzahl an maximal konsistenten Mengen $MC(k)$ mit der Anzahl an selbst-widersprechenden Formeln der Menge $SC(k)$:*

$$I_{MC} = (\mid MC(k) \mid + \mid SC(k) \mid) - 1$$

Damit das Inkonsistenzmaß I_{MC} eine konsistente Wissensbasis auf eine 0 abbildet, wird die Addition um 1 reduziert [8].

Beispiel 7. *Sei die Wissensbasis k konsistent. Dann gibt es keine selbst-widersprechenden Formeln und es gilt $\mid SC(k) \mid = 0$. Dazu gibt es nur eine maximal konsistente Menge, nämlich die Wissensbasis k selbst, also $\mid MC(k) \mid = 1$. Damit bildet das Inkonsistenzmaß I_{MC} bei einer konsistenten Wissensbasis auf eine 0 ab.*

$$I_{MC} = (\mid MC(k) \mid + \mid SC(k) \mid) - 1 = (1 + 0) - 1 = 0 \tag{3}$$

Die zwei vorgestellten Inkonsistenzmaße basieren auf den minimal inkonsistenten und maximal konsistenten Mengen. Daher können sie in den Bereich der konsistenz-basierten Analyse kategorisiert werden. Neben der konsistenz-basierten Analyse, kann die Berechnung auch auf probabilistischen Funktionen oder erkenntnistheoretischen Aktionen basieren [10]. Während die wahrscheinlichkeitstheoretische Analyse auf einer Funktion aufgebaut ist, die jeder Menge an Formeln einen Wert zwischen 0 und 1 zuordnet, basieren erkenntnistheoretische Aktionen auf dem Auflösen von Inkonsistenzen. Der Inkonsistenzwert repräsentiert somit die Anzahl oder die Kosten der Aktionen um die Inkonsistenz aufzulösen.

4 Inkonsistenzmessung von Planungsproblemen

Werden die zwei Themengebiete der Planungsprobleme und der Inkonsistenzmessung miteinander verglichen, dann sind Ähnlichkeiten zu erkennen. Ein Planungsproblem kann zu einer Lösung führen, welches vergleichbar ist mit einer konsistenten Wissensbasis. Ebenso kann das Planungsproblem unlösbar sein und entspricht somit einer inkonsistenten Wissensbasis. Lösbare Planungsprobleme würden damit auf eine 0 abgebildet und unlösbare Planungsprobleme auf eine Zahl größer 0. Siehe dazu die *Konsistenz*-Eigenschaft aus Kapitel 3.2.

Genauso wie eine Ordnung zwischen inkonsistenten Wissensbasen durch die Inkonsistenzmessung existiert, gibt es Planungsprobleme, die unlösbarer sind als andere Planungsprobleme. Dadurch kann die Schwere der Unlösbarkeit ebenfalls geordnet werden. Zur Vergleichbarkeit ist die Abbildung der Ordnung auf den Bereich zwischen 0 bis 1 beschränkt. Dies entspricht der *Normalisierung*.

Beispiel 8. *Sei dazu das Beispiel des Supermarkts in vereinfachter PDDL-Schreibweise aus Listing 11 gegeben.*

```
transport(FROM, TO, FRUIT):
    C: at(FROM, FRUIT) and  transfer(FROM,TO)
    E: at(TO, FRUIT) and not at(FROM,FRUIT)
```

Listing 11: Operation *transport* des Supermarkt-Beispiels

Der Startzustand und das Ziel des Planungsproblems Π *ist dabei in Listing 12 zu finden.*

```
init:
    at(germany, apple), at(spain, orange), at(greece, grape)
goal:
    at(market, banana), at(market, apple),
    at(market, orange)
```

Listing 12: Startzustand und Ziel des Planungsproblems Π

Dabei wird vom Supermarkt die Frucht Banane angefordert, jedoch ist sie in keinem Land verfügbar. Das Planungsproblem ist nicht lösbar.

Sei ein Planungsproblem Π' *mit dem Startzustand aus Listing 13 gegeben, in dem nur noch Trauben aus Griechenland verfügbar sind.*

```
init:
    at(greece, grape)
goal:
    at(market, banana), at(market, apple),
    at(market, orange)
```

Listing 13: Startzustand und Ziel des Planungsproblems Π'

Dadurch, dass weder die Bananen, die Äpfel oder die Orangen angeboten werden können, ist damit das Planungsproblem Π' *unlösbarer als das Planungsproblem* Π *in dem wenigstens die Äpfel und die Orangen geliefert werden können. Somit sollte das Planungsproblem* Π' *einen höheren Inkonsistenzwert erhalten.*

Da ein Zustand eine Menge an Atomen ist, können die Operationen aus der Mengenlehre auf die Zustände angewandt werden. Damit ist es möglich die Anzahl an Atomen durch die Kardinalität des Zustandes zu erhalten. Ebenso entsteht mit dem Durchschnitt und der Vereinigung ein neuer Zustand. Der Durchschnitt $\cap$ erzeugt einen Zustand, der alle gemeinsamen Atome enthält, während die Vereinigung $\cup$ einen Zustand mit allen Atomen aus Zustand s_1 und Zustand s_2 erzeugt. Ist ein Atom α in einem Zustand s enthalten, dann wird dies mit $\alpha \in s$ dargestellt.

Dieses Kapitel definiert zunächst grundlegende Begriffe um ein Planungsproblem auf ein Inkonsistenzmaß zu definieren. Weiterhin werden die Eigenschaften der Inkonsistenzmessung auf ein Planungsproblem definiert. Neben den Eigenschaften aus der Inkonsistenzmessung besitzt ein Planungsproblem ebenso Eigenschaften, die im Kapitel 4.1.4 definiert werden. Abschließend werden Maße definiert, die die Inkonsistenz eines Planungsproblems berechnen.

4.1 Definitionen

Um die Inkonsistenzmessung auf ein Planungsproblem Π anwenden zu können, sind Definitionen der Inkonsistenzmessung auf Planungsprobleme notwendig. Dazu werden nachfolgend die Begriffe konsistent und inkonsistent anstatt lösbar und unlösbar verwendet, da es um die Messung der Inkonsistenz im Planungsproblem geht.

Definition 14 (Konsistentes Planungsproblem)**.** *Ein Planungsproblem $\Pi = \langle O, s_I, g \rangle$ ist konsistent, wenn es einen Pfad $p \in P$ vom Startzustand s_I zu einem Zielzustand s_G gibt und dieses damit lösbar ist: $\gamma(s_I, p) = s_G$.*

Gleichermaßen führt das Nichterreichen eines Zielzustandes zu einem inkonsistenten Planungsproblem.

Definition 15 (Inkonsistentes Planungsproblem)**.** *Kann kein Pfad p, beginnend am Startzustand s_I, einen Zielzustand s_G im Planungsproblem $\Pi = \langle O, s_I, g \rangle$ erreichen, dann ist das Planungsproblem inkonsistent.*

Da ein Zielzustand aus mehreren Atomen bestehen kann, ist es möglich diese als Teilziele zu definieren.

Definition 16 (Teilziel)**.** *Ein Teilziel α ist ein Atom eines Zielzustandes s_G, also $\alpha \in s_G$.*

Definition 17 (Unerreichbares Teilziel)**.** *Ein Teilziel $\alpha \in s_G$ kann auf einem Pfad p nicht erreicht werden, wenn dieser Pfad p, ausgehend vom Startzustand s_I, nicht im Teilziel α endet: $\gamma(s_I, p) \neq \alpha$.*

Ähnlich der Inkonsistenzmessung lässt sich nun eine Funktion $I_{Planung}$ definieren, die die Inkonsistenz in einem Planungsproblem Π berechnet.

Definition 18 (Funktionen zur Messung der Inkonsistenz)**.** *Die Inkonsistenz eines Planungsproblems Π kann mit einer Funktionen der Form $I_{Planung} : \Pi \rightarrow \mathbb{R}^{\geq 0}$ realisiert werden.*

Die Eingabe der Funktion $I_{Planung}$ ist ein Planungsproblem Π, welches auf eine Zahl größer gleich 0 abbildet. Zur Vergleichbarkeit zwischen Planungsproblemen, ist der Inkonsistenzwert immer zwischen 0 und 1.

Definition 19 (Normalisierung). *Der Funktionswert von $I_{Planung} : \Pi \to \mathbb{R}^{\geq 0}$ von einem Planungsproblem Π liegt immer zwischen 0 und 1:* $0 \leq I_{Planung}(\Pi) \leq 1$.

$$I_{Planung}(\Pi) = \begin{cases} 0 & \textit{für ein konsistentes Planungsproblem } \Pi \\ 0 < y \leq 1 & \textit{für ein inkonsistentes Planungsproblem } \Pi \end{cases}$$

Definition 20 (Maximale Inkonsistenz). *Bildet ein Maß $I_{Planung}$ ein Planungsproblem auf eine 1 ab, dann ist das Planungsproblem maximal inkonsistent bezüglich des Maßes $I_{Planung}$.*

Weiterhin lassen sich die Eigenschaften aus Kapitel 3.2 auf die Messung der Inkonsistenz in Planungsproblemen, siehe Kapitel 4.2, übertragen.

4.1.1 Eigenschaft Monotonie

Die Inkonsistenz innerhalb einer Wissensbasis kann bei der Hinzunahme von neuem Wissen nur größer werden oder gleich bleiben. Dies beschreibt die Eigenschaft der Monotonie. Bezogen auf ein Planungsproblem kann die Hinzunahme eines Atoms in das Ziel ebenso zu einer größeren Inkonsistenz führen. Das neue Teilziel darf jedoch nicht erreichbar sein, da sonst die Inkonsistenz verringert werden kann.

Theorem 1. *Die Hinzunahme eines Atoms α in das Ziel, welches durch einen Pfad beginnend vom Startzustand erreicht werden kann, verringert die Inkonsistenz.*

Beweis. Sei ein Planungsproblem $\Pi = \langle O, s_I, g \rangle$ für ein beliebiges Maß im maximal inkonsistenten Fall gegeben. Dann wird Π auf eine 1 abgebildet: $I_{Planung}(\Pi) = 1$. Wird nun ein erreichbares neues Teilziel α dem Ziel g hinzugefügt, dann wird das neue Planungsproblem $\Pi' = \langle O, s_I, g \cup \{\alpha\} \rangle$ auf eine Zahl kleiner 1 abgebildet, da der maximal inkonsistente Fall nicht mehr gegeben ist: $I_{Planung}(\Pi') < 1$. Somit kann ein erreichbares neues Teilziel α die Inkonsistenz verringern. □

Definition 21 (Monotonie im Ziel eines Planungsproblems). *Wird dem Ziel g eines Planungsproblems $\Pi = \langle O, s_I, g \rangle$ ein Atom α, welches nicht erreichbar ist, hinzugefügt, dann ist die Inkonsistenz größer oder gleich.*

$$I_{Planung}(\Pi) \leq I_{Planung}(\Pi'), \textit{ wobei } \Pi' = \langle O, s_I, g \cup \{\alpha\} \rangle \textit{ und } \alpha \textit{ ist nicht erreichbar}$$

Wird das Atom hingegen in den Startzustand hinzugefügt, dann bleibt die Inkonsistenz gleich oder verringert sich. Ist das neue Atom für die Lösung des Planungsproblems notwendig, dann kann sich der Grad der Inkonsistenz verringert und der Inkonsistenzwert sinkt.

Definition 22 (Monotonie im Startzustand eines Planungsproblems). *Wird dem Startzustand s_I eines Planungsproblems $\Pi = \langle O, s_I, g \rangle$ ein Atom α hinzugefügt, dann ist die Inkonsistenz kleiner oder gleich.*

$$I_{Planung}(\Pi) \geq I_{Planung}(\Pi'), \textit{ wobei } \Pi' = \langle O, s_I \cup \{\alpha\}, g \rangle$$

Ebenso kann ein Atom α in eine Vorbedingung einer Operation hinzugefügt werden. Mit einem neuen Atom α in der Vorbedingung einer ausführbaren Operation, ist es möglich, dass diese Operation danach nicht mehr ausführbar ist. Ist die Operation auch mit dem zusätzlichen Atom weiterhin ausführbar, dann bleibt der Inkonsistenzwert gleich. Hingegen kann die Inkonsistenz anwachsen, wenn die Operation nicht mehr ausführbar ist. Ist die Operation bereits vor dem Hinzufügen des Atoms nicht ausführbar, dann ist Operation mit dem zusätzlichen Atom α in der Vorbedingung ebenfalls nicht ausführbar und kann zu einem größeren Inkonsistenzwert führen als zuvor.

Definition 23 (Monotonie in Vorbedingungen eines Planungsproblems)**.** *Wird der Vorbedingung einer Operation $o \in O$ eines Planungsproblems $\Pi = \langle O, s_I, g\rangle$ ein Atom α hinzugefügt, dann bleibt die Inkonsistenz gleich oder verringert sich, wenn die Operation o' ausführbar ist. Ist die Operation o' hingegen nicht ausführbar, dann kann die Inkonsistenz anwachsen.*

$$I_{Planung}(\Pi) \geq I_{Planung}(\Pi'),\ \textit{wenn } o' \textit{ ausführbar ist}$$
$$I_{Planung}(\Pi) \leq I_{Planung}(\Pi'),\ \textit{wenn } o' \textit{ nicht ausführbar ist}$$
$$\textit{wobei } \Pi' = \langle O', s_I, g\rangle \textit{ mit } o' = \langle c', e\rangle \in O' \textit{ und } c' = c \cup \{\alpha\}$$

Beispiel 9. *Seien dazu die Operationen O des Planungsproblems $\Pi = \langle O, s_I, g\rangle$ in Listing 14 aufgeführt.*

```
1 func1():
2     C: a, b
3     E: c
4 func2():
5     C: d
6     E: e
7 func3():
8     C: e, c
9     E: f
```

Listing 14: Operationen des Planungsproblems Π

Der Startzustand und das Ziel des Planungsproblems Π ist dem Listing 15 zu entnehmen.

```
1 init:
2     a, b, g
3 goal:
4     f
```

Listing 15: Startzustand und Ziel des Planungsproblems Π

Von den drei Operationen ist nur die Operation func1 ausführbar. Da sowohl func1 als auch func2 benötigt werden um das Ziel zu erreichen, ist ersichtlich, dass das Planungsproblem Π inkonsistent ist.

Wird nun die Monotonie im Startzustand angewandt und das Atom d in den Startzustand hinzugenommen, dann kann das Ziel erreicht werden und das Planungsproblem Π' *ist konsistent. Damit hat sich der Inkonsistenzwert verringert.*

Anstelle des Startzustandes kann das Atom d in das Ziel hinzugefügt werden. Dadurch, dass d unerreichbar ist, muss laut der Monotonie im Ziel der Inkonsistenzwert größer oder gleich bleiben. Dadurch, dass sowohl das Atom f als auch das Atom d nicht zu erreichen ist, besitzt das Planungsproblem Π' *einen größeren oder gleichen Inkonsistenzwert als das Planungsproblem* Π*.*

Für die Monotonie einer Vorbedingung einer Operation wird das Atom g der Vorbedingung der Operation func1 des Planungsproblems Π *hinzugefügt. Die Veränderung der Operation func1 ist dem Listing 16 zu entnehmen. Da das Atom g bereits im Startzustand enthalten ist, bleibt die Operation func1 weiterhin ausführbar. An der Inkonsistenz des Planungsproblems* Π' *ändert sich jedoch nichts.*

```
1 func1():
2     C: a, b, g
3     E: c
```

Listing 16: Operation *func1* des Planungsproblems Π'

Wird die Vorbedingung der Operation func1 jedoch mit dem Atom d erweitert, dann ist die Operation nicht mehr ausführbar. Listing 17 definiert die neue Operation func1.

```
1 func1():
2     C: a, b, d
3     E: c
```

Listing 17: Operation *func1* des Planungsproblems Π'

Da nun auch die Operation func1 nicht mehr ausführbar ist, wächst die Inkonsistenz im Planungsproblem Π' *an.*

4.1.2 Eigenschaft Unabhängigkeit eines Atoms

Die Eigenschaft der Unabhängigkeit von freien Formeln besagt, dass eine Formel, die in keiner inkonsistenten Menge vorkommt, keinen Einfluss auf den Inkonsistenzwert hat. Somit bleibt die Inkonsistenz dieser Menge gleich, egal ob diese Formel in der Menge enthalten ist oder nicht. Auf ein Planungsproblem übertragen bestimmen die Atome des Startzustandes, des Ziels und der Vorbedingung einer Operation den Inkonsistenzwert. Daher ist es möglich hier ebenfalls Atome zu finden, die keinen Einfluss auf den Inkonsistenzwert haben.

Sowohl im Startzustand s_I als auch im Ziel g des Planungsproblems $\Pi = \langle O, s_I, g \rangle$ kann das Entfernen eines Atoms α zu Änderungen im Inkonsistenzwert führen. Bleibt der Inkonsistenzwert jedoch gleich, dann ist das Atom α unabhängig. Ist das Atom α unabhängig, dann kann es auf einem Pfad des Planungsproblems durch eine ausführbare Operation erzeugt worden sein. Dies gilt sowohl für den Startzustand als auch für das Ziel. Für den Startzustand gilt weiterhin, dass ein Atom α

unabhängig ist, wenn es keinen Einfluss auf die Ausführbarkeit des Planungsproblems Π hat.

Definition 24 (Unabhängigkeit eines Atoms im Startzustand)**.** *Bleibt der Inkonsistenzwert eines Planungsproblems $\Pi = \langle O, s_I, g\rangle$ gleich, nach dem ein Atom α aus dem Startzustand s_I entfernt wurde, dann ist das Atom α unabhängig vom Planungsproblem:*

$$I_{Planung}(\Pi) = I_{Planung}(\Pi'),\ \textit{wobei}\ \Pi' = \langle O, s_I \setminus \{\alpha\}, g\rangle\ \textit{und}\ \alpha\ \textit{ist unabhängig}$$

Definition 25 (Unabhängigkeit eines Atoms im Ziel)**.** *Bleibt der Inkonsistenzwert eines Planungsproblems $\Pi = \langle O, s_I, g\rangle$ nach dem Entfernen eines Atoms α aus dem Ziel g gleich, dann ist das Atom α unabhängig vom Ziel g.*

$$I_{Planung}(\Pi) = I_{Planung}(\Pi'),\ \textit{wenn}\ \Pi' = \langle O, s_I, g \setminus \{\alpha\}\rangle\ \textit{und}\ \alpha\ \textit{ist unabhängig}$$

Das Entfernen eines Atoms α aus einer Vorbedingung einer Operation kann zu einer Verringerung des Inkonsistenzwertes führen. Wird das Atom α aus der Vorbedingung einer Operation o entfernt und die Operation o war auf dem Zustand s zuvor ausführbar, dann ändert bezüglich der Ausführbarkeit der Operation o' auf diesen Zustand s nichts. Jedoch könnte die Operation o' nun auf Zustände angewandt werden, bei denen die Operation o fehl geschlagen ist. Ist dies der Fall, dann ist das Atom α nicht unabhängig von der Vorbedingung, da sich der Inkonsistenzwert verringert kann. Bleibt der Inkonsistenzwert jedoch gleich, dann gilt das Atom α als unabhängig.

Definition 26 (Unabhängigkeit eines Atoms in einer Vorbedingung)**.** *Bleibt der Inkonsistenzwert eines Planungsproblems $\Pi = \langle O, s_I, g\rangle$ nach dem Entfernen eines Atoms α aus der Vorbedingung einer Operation $o = \langle c, e\rangle \in O$ gleich, dann ist das Atom α unabhängig von der Vorbedingung der Operation o.*

$$I_{Planung}(\Pi) = I_{Planung}(\Pi'), \textit{wenn}\ \Pi' = \langle O', s_I, g\rangle\ \textit{mit}\ o' = \langle c', e\rangle \in O'$$
$$\textit{und}\ c' = c\backslash\{\alpha\}\ \textit{und}\ \alpha\ \textit{ist unabhängig}$$

Beispiel 10. *Sei dazu ein Planungsproblem $\Pi = \langle O, s_I, g\rangle$ mit den Operationen aus Listing 18 gegeben.*

```
1 func1():
2     C: a, b
3     E: c
4 func2():
5     C: d, e
6     E: f
7 func3():
8     C: c, f
9     E: i
```

Listing 18: Operationen des Planungsproblems Π

Der Startzustand und das Ziel des Planungsproblems Π *ist dabei dem Listing 19 zu entnehmen.*

```
1 init:
2    a, b, c, g, d
3 goal:
4    i, f
```

Listing 19: Startzustand und Ziel des Planungsproblems Π

Wird dem Startzustand s_I *sowohl das Atom c als auch das Atom g entfernt, dann ändert sich die Inkonsistenz nicht, da das Atom c durch die Operation func1 erzeugt werden kann. Das Atom g ist für die Ausführung des Planungsproblems* Π *nicht relevant. Somit sind sowohl das Atom c als auch das Atom g im Startzustand* s_I *unabhängig.*

Das Teilziel f wird benötigt um das Teilziel i erreichen zu können. Wird es daher aus dem Ziel g entfernt, dann bleibt die Inkonsistenz gleich. Damit ist das Teilziel f unabhängig.

Das Atom a aus der Vorbedingung der Operation func1 hat keinen Einfluss auf die Ausführbarkeit des Planungsproblems Π*. Der Inkonsistenzwert bleibt nach dem Entfernen des Atoms a gleich. Daher ist das Atom a unabhängig von der Operation func1.*

Würde in der Operation func2 das Atom e aus der Vorbedingung entfernt, dann wäre das Planungsproblem Π *lösbar und die Inkonsistenz hätte sich verringert. Daher ist das Atom e nicht unabhängig. Hingegen führt das Entfernen des Atoms d zur gleichen Inkonsistenz wie zuvor und ist somit unabhängig.*

4.1.3 Eigenschaft Dominanz

Die Dominanz beschreibt, dass eine konsistente stärkere Formel α potentiell größere Inkonsistenzen bringt als eine Formel β. Dies ist vergleichbar mit einem Atom α, welches in einer Vorbedingung enthalten ist und einem Atom β, welches in einem Effekt vorhanden ist. Dabei ist die Vorbedingung erfüllt und die Operation ist ausführbar.

Da die Atome einer Vorbedingung die Ausführbarkeit einer ganzen Operation verhindern können, haben diese Atome ein größeres Potential Inkonsistenzen hervorzurufen. Atome aus Effekten können andere Operationen hingegen ausführbar machen und können eine Inkonsistenz verringern. Dazu hängen die Atome aus der Vorbedingung und den Effekt zusammen, da ohne die Erfüllbarkeit der Vorbedingung, die Atome der Effekte nicht realisierbar sind. Somit dominieren die Atome der Vorbedingung die der Effekte.

Definition 27 (Dominanz in einer Operation)**.** *Ein Atom* α *dominiert über das Atom* β*, wenn* α *in der Vorbedingung der ausführbaren Operation* $o = \langle c, e\rangle$ *und das Atom* β *im Effekt der Operation o hinzugefügt wird. Dabei führt das Atom* α *potentiell zu einer größeren Inkonsistenz als das Atom* β*. Die Planungsprobleme sind:* $\Pi = \langle O, s_I, g\rangle$ *und* $\Pi' = \langle O', s_I, g\rangle$*, wobei* $o = \langle c \cup \{\alpha\}, e\rangle \in O$ *und* $o' = \langle c, e \cup \{\beta\}\rangle \in O'$*.*

$$I_{Planung}(\Pi) \geq I_{Planung}(\Pi')$$

Beispiel 11. *Sei dazu ein Planungsproblem* $\Pi = \langle O, s_I, g \rangle$ *mit den Operationen aus Listing 20 und dem Startzustand und Ziel aus Listing 21 gegeben.*

```
1 func1():
2     C: a, b
3     E: e
4 func2():
5     C: a, c
6     E: f
```

Listing 20: Operationen des Planungsproblems Π

```
1 init:
2     a, b
3 goal:
4     e, f
```

Listing 21: Startzustand und Ziel des Planungsproblems Π

Von den zwei Teilzielen ist nur das Teilziel e über die Operation func1 erreichbar. Dazu ist die Operation func1 als einzige Operation ausführbar. Wird der Operation func1 jedoch ein Atom zur Vorbedingung hinzugefügt, zum Beispiel das Atom c, dann ist auch dieses Teilziel e nicht mehr erreichbar. Das liegt daran, dass die Operation func1 nicht mehr ausführbar ist. Sei das Planungsproblem mit dem Atom c in der Vorbedingung der Operation func1 als Π' *gegeben.*

Wird aber das Atom f in den Effekt der Operation func1 hinzugefügt, dann ist das Planungsproblem anschließend konsistent. Selbst wenn andere Atome in den Effekt der Operation func1 hinzugefügt werden würden, dann steige die Inkonsistenz nicht an, sondern bliebe gleich. Sei das Planungsproblem mit dem Atom f im Effekt der Operation func1 als Π'' *gegeben.*

$$I_{Planung}(\Pi') > I_{Planung}(\Pi'') \tag{4}$$

Da das Planungsproblem Π' *zum vorherigen Planungsproblem* Π *inkonsistenter geworden ist und das Planungsproblem* Π'' *konsistent ist, gilt die obige Gleichung. Das Atom c dominiert somit über das Atom f.*

4.1.4 Kritische Atome

In einem Planungsproblem gibt es Atome, die neben der Dominanz Atome die Inkonsistenz stärker beeinflussen können als andere Atome. Potenziell verändert ein Atom in einer Vorbedingung die Inkonsistenz stärker als ein Atom, welches in einem Effekt oder im Startzustand vorhanden ist. Das liegt daran, dass die Atome in Vorbedingungen die Ausführung von Operationen verhindern können. Hingegen werden die Atome aus den Effekten erzeugt, was dazu führen kann, dass Operationen ausführbar werden.

Atome können daher geordnet und auf eine Zahl zwischen 0 und 1 abgebildet werden. Die Wertigkeit des Atoms gibt Aufschluss darüber wie stark das Atom die Inkonsistenz im Planungsproblem beeinflussen kann. Dafür wird die Funktion $\delta : \alpha \rightarrow \{0, 1\}$ definiert, die auf eine 0 abbildet, falls sich das Atom in einem Effekt befindet. Ist das Atom in einer Vorbedingung vorhanden, dann bildet die Funktion δ auf eine 1 ab.

Definition 28 (Funktion δ)**.** *Die Funktion δ bekommt ein Atom übergeben und bildet dieses auf eine 0 oder 1 ab.*

$$\delta : \alpha \rightarrow \{0, 1\}$$
$$\delta(\alpha) = \begin{cases} 1 & \textit{wenn } \alpha \textit{ in einer Vorbedingung} \\ 0 & \textit{wenn } \alpha \textit{ in einem Effekt} \end{cases}$$

Kommt ein Atom häufiger im Planungsproblem vor, dann werden dessen Werte der Funktion δ addiert. Um die Wertigkeit des Atoms zu berechnen, wird abschließend durch die Häufigkeit des Atoms in allen Operationen geteilt. Damit ist die Wertigkeit immer zwischen 0 und 1.

Definition 29 (Wertigkeit eines Atoms)**.** *Die Wertigkeit ω eines Atoms α berechnet sich aus der Summe der Funktion δ durch die Häufigkeit des Atoms in Vorbedingungen und Effekten:*

$$\omega = \frac{\sum \delta(\alpha)}{\textit{Häufigkeit des Atoms } \alpha}$$

Je größer die Wertigkeit eines Atoms, desto kritischer ist das Atom im Planungsproblem, da es den Inkonsistenzwert stärker verändern kann als ein Atom mit einer geringeren Wertigkeit.

Definition 30 (Kritische Atome)**.** *Ein Atom α ist kritischer als β, wenn es eine höhere Wertigkeit als das Atom β hat.*

Beispiel 12. *Sei dazu ein Beispiel gegeben, in dem es zwei Räume gibt. Diese zwei Räume sollen die Küche und das Wohnzimmer sein und sind durch eine Tür miteinander verbunden. Das Ziel ist es mit einem Roboter Getränke aus der Küche in das Wohnzimmer zu bringen. Listing 22 stellt die Operationen des Planungsproblems $\Pi = \langle O, s_I, g \rangle$ dar und Listing 23 zeigt den Startzustand und das Ziel.*

```
1 move(FROM, TO):
2     C: door(open), at(FROM, robot)
3     E: at(TO, robot), not at(FROM, robot)
```

Listing 22: Operationen des Planungsproblems Π

```
1 init:
2     at(kitchen, water), at(kitchen, tea), at(kitchen, coffee),
3     at(livingRoom, robot), door(open)
```

door(open)	at(kitchen,robot)	at(livingRoom,robot)
1	0,33	0,33

Abbildung 2: Wertigkeiten der Atome aus der Operation *move*

```
4 goal:
5     at(livingRoom, water), at(livingRoom, juice),
6     at(livingRoom, lemonade), at(livingRoom, tea)
```

Listing 23: Startzustand und Ziel des Planungsproblems Π

Die Wertigkeiten für die Atome aus der Operation sind der Abbildung 2 zu entnehmen. Alle weiteren Atome haben die Wertigkeit 0.

$$\begin{aligned} \omega(\textit{door(open)}) &= \frac{1}{1} = 1 \\ \omega(\textit{at(kitchen, robot)}) &= \frac{1+0+0}{2} \approx 0,33 \\ \omega(\textit{at(livingRoom, robot)}) &= \frac{1+0+0}{3} \approx 0,33 \end{aligned} \tag{5}$$

Das Atom door(open) ist in der Vorbedingung enthalten und wird daher mit der Funktion δ auf eine 1 abgebildet. Insgesamt kommt es einmalig in den Operationen des Planungsproblems Π vor und erhält dadurch den Wert 1. Die Inkonsistenz verändert sich also potentiell stärker, falls die Tür geschlossen ist.

Ist die Tür offen, dann ist die Berechnung der Inkonsistenz davon abhängig, welche Getränke in der Küche vorhanden sind. Hingegen kann kein einziges Getränk in das Wohnzimmer gebracht werden, wenn die Tür geschlossen ist. Somit führt das Atom door(open) zu einer größeren Inkonsistenz und ist kritischer als eines der fehlenden Getränke.

4.2 Inkonsistenzmaße

Die Inkonsistenz in einem Planungsproblem kann entstehen, wenn Vorbedingungen nicht erfüllt sind und somit Teilziele nicht erreicht werden können. Anhand dessen ist es möglich die Inkonsistenz auf die Anzahl der erreichten Teilziele abzubilden. Weiterhin ist es möglich, dass Atome so lange zum Startzustand hinzugefügt werden bis schließlich ein Zielzustand erreicht werden kann. Ebenso kann die Inkonsistenz aufgelöst werden, in dem Operationen angewandt werden, obwohl die Vorbedingung nicht erfüllt ist.

Während ein Algorithmus eine Lösung für ein Planungsproblem sucht, entstehen durch Zustandsänderungen neue Zustände. Diese neuen Zustände sind potentiell näher an einem Zielzustand als die vorherigen Zustände. Jedoch kann es passieren, dass die neuen Zustände weniger Teilziele aus dem Ziel erfüllen als der vorherige Zustand. Daher erfolgt die Berechnung des Inkonsistenzwertes immer auf den Zuständen, die am meisten Teilziele des Ziels erreichen. Daher gilt je mehr ein Zustand an Teilzielen erreicht, desto günstiger ist der Zustand.

Definition 31 (Günstigster Zustand)**.** *Ein günstigster Zustand s_F ist vom Startzustand s_I erreichbar und enthält am meisten erreichbare Teilziele des Planungsproblems* $\Pi = \langle O, s_I, g \rangle$.

Im konsistenten Fall enthält der günstigste Zustand s_F alle Teilziele des Ziels g. Ist der Zustand s_F günstig, dann gibt es keinen weiteren Zustand s_n, so dass dieser mehr Teilziele erreicht: $\mid s_n \cap g \mid \leq \mid s_F \cap g \mid \leq \mid g \mid$.

Wird für die Berechnung des Inkonsistenzwertes nicht der günstigste Zustand gewählt, dann variiert der Inkonsistenzwert innerhalb eines Planungsproblems Π.

Einige Maße lösen die Inkonsistenz schrittweise auf. Dabei entstehen Pfade, die im ursprünglichen Planungsproblem nicht vorhanden sind. Diese Pfade werden nachfolgend als Pseudo-Pfade bezeichnet.

Definition 32 (Pseudo-Pfad)**.** *Ein Pseudo-Pfad in einem Planungsproblem* $\Pi = \langle O, s_I, g \rangle$ *beginnt am Startzustand s_I und besitzt Zustandsänderungen γ, die im Planungsproblem* Π *nicht möglich sind.*

Endet der Pseudo-Pfad in einem Zielzustand s_G, dann ist eine Pseudo-Lösung gefunden.

Definition 33 (Pseudo-Lösung)**.** *Eine Pseudo-Lösung eines Planungsproblems* $\Pi = \langle O, s_I, g \rangle$ *ist ein Pseudo-Pfad, der vom Startzustand s_I in einen Zielzustand s_G führt.*

4.2.1 Maß der Teilziele

Das Ziel eines Planungsproblems kann mehrere Teilziele enthalten. Daher ist es intuitiv, dass die Inkonsistenz über die Anzahl der unerreichten Teilziele gebildet wird. Zur Normalisierung wird die Anzahl an unerreichten Teilzielen durch die Anzahl aller Teilziele dividiert. Um unterschiedliche Ergebnisse pro Zustand zu vermeiden, muss die Berechnung auf einem günstigsten Zustand ausgeführt werden.

Zu Beginn wird die Anzahl aller Teilziele m ermittelt. Dies geschieht durch die Kardinalität des Ziels g.

Definition 34 (Anzahl aller Teilziele)**.** *Die Anzahl aller Teilziele m ergibt sich aus der Kardinalität des Ziels g:*

$$m = \mid g \mid$$

Anschließend muss der günstigste Zustand ermittelt werden. Gibt es mehrere günstigste Zustände, dann kann jeder dieser günstigen Zustände für die weitere Berechnung verwendet werden. Die Anzahl an erreichten Teilzielen ergibt sich anschließend aus der Kardinalität der Vereinigung des günstigsten Zustandes mit dem Ziel.

Definition 35 (Anzahl erreichter Teilziele)**.** *Die Anzahl an erreichten Teilzielen l eines Zustands s ergibt sich aus der Kardinalität der Vereinigung des Ziels g mit dem Zustand s.*

$$l = \mid s \cap g \mid$$

Abschließend berechnet sich die Anzahl an unerreichten Teilzielen aus der Differenz der Anzahl aller Teilziele m mit der Anzahl der erreichten Teilziele l.

Definition 36 (Anzahl unerreichter Teilziele)**.** *Die Anzahl der unerreichten Teilziele n ergibt sich aus der Differenz der Anzahl aller Teilziele m mit der Anzahl der erreichten Teilziele im günstigsten Zustand s_F:*

$$n = m - \mid s_F \cap g \mid$$

Um die Normalisierung zu realisieren, wird die Anzahl an unerreichbaren Teilzielen n schließlich durch die Anzahl aller Teilziele m dividiert. Dabei muss die Anzahl an Teilzielen größer $m > 0$ sein, um eine Division durch 0 zu verhindern. Das Ziel darf also nicht leer sein.

Definition 37 (Maß der Teilziele)**.** *Das Inkonsistenzmaß der Teilziele lässt sich durch die Division der Anzahl an unerreichten Teilziele n mit der Anzahl aller Teilziele m berechnen:*

$$\begin{aligned} &I_{Teilziele} : \Pi \to \mathbb{R}^{\geq 0} \\ &I_{Teilziele}(\Pi) = \frac{\textit{Anzahl unerreichter Teilziele}}{\textit{Anzahl aller Teilziele}} = \frac{n}{m}, \\ &\textit{für } 0 \leq n \leq m \end{aligned}$$

Der konsistente Fall wird damit auf eine 0 abgebildet, da alle Teilziele erreicht werden können und $n = 0$ ist. Hingegen ist der maximal inkonsistente Fall gegeben, wenn alle Teilziele unerreichbar sind: $n = m$.

$$\begin{aligned} I_{Teilziele}(\Pi) = \frac{n}{m} = \frac{0}{m} = 0, \text{ für } n = 0 \text{ und } m > 0 \\ I_{Teilziele}(\Pi) = \frac{n}{m} = \frac{m}{m} = 1, \text{ für } n = m \text{ und } m > 0 \end{aligned} \tag{6}$$

Beispiel 13. *Sei dazu ein Planungsproblem $\Pi = \langle O, s_I, g \rangle$ mit 3 Operationen gegeben, siehe Listing 24.*

```
1 func1():
2     C: a
3     E: not c, b
4 func2():
5     C: b
6     E: not b, e
7 func3():
8     C: a, not b
9     E: not e
```

Listing 24: Operationen des Planungsproblems Π

Der Startzustand und das Ziel für das Planungsproblem Π ist in Listing 25 definiert.

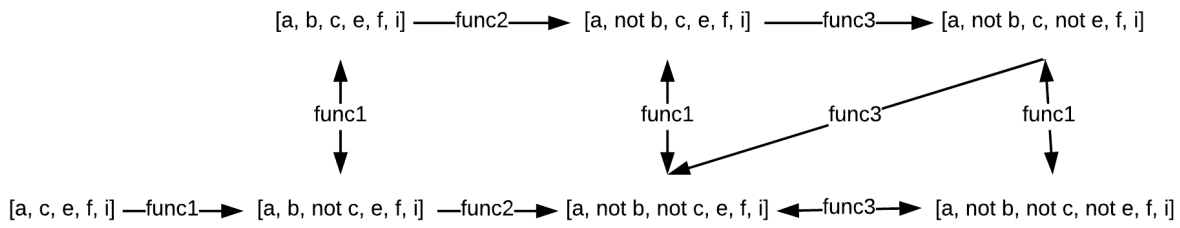

Abbildung 3: Diese Abbildung zeigt die möglichen Zustände des Planungsproblems Π

```
1 init:
2    a, c, e, f, i
3 goal:
4    a, b, not c, not e, f, not i
```

Listing 25: Startzustand und Ziel des Planungsproblems Π

Alle möglichen Zustände des Planungsproblems Π *sind in der Abbildung 3 dargestellt. Die Pfeile symbolisieren das Ausführen der entsprechend genannten Operation.*

Das Ziel hat insgesamt $m = 6$ *Teilziele, wovon nicht alle in einem Zustand erreicht werden können. Das Planungsproblem* Π *ist daher inkonsistent. Es gibt jedoch zwei Zustände, die jeweils 4 der 6 Teilziele erreichen. Diese beiden Zustände sind günstige Zustände. Die Berechnung des Inkonsistenzwertes bezieht sich somit auf die maximale Anzahl an erreichten Teilzielen in einem Zustand von* $l = 4$*. Die Anzahl an unerreichten Teilzielen ergibt sich daraus mit:* $n = m - l = 6 - 4 = 2$*. Somit hat das Planungsproblem* Π *einen Inkonsistenzwert durch das Maß der Teilziele* $I_{Teilziele}$ *von ungefähr* $0,33$*.*

$$I_{Teilziele}(\Pi) = \frac{6-4}{6} = \frac{1}{3} \approx 0,33 \tag{7}$$

Sei ein Planungsproblem $\Pi' = \langle O, s'_I, g \rangle$ *gegeben, welches sich nur im Startzustand* s'_I *zum Planungsproblem* Π *unterscheidet, siehe Listing 26.*

```
1 init:
2    b, c, e, f, not i
3 goal:
4    a, b, not c, not e, f, not i
```

Listing 26: Startzustand und Ziel des Planungsproblems Π'

Die erreichbaren Zustände im Planungsproblem Π' *sind in der Abbildung 4 dargestellt. Auch hier entsprechen Pfeile einer ausgeführten Operation.*

Insgesamt kann das Planungsproblem Π' *nur zwei Zustände erreichen. Im Startzustand können 3 Teilziele erreicht werden, hingegen erreicht der zweite Zustand nur noch*

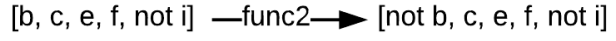

Abbildung 4: Diese Abbildung zeigt die möglichen Zustände des Planungsproblems Π'

2. Dadurch ist der Startzustand der günstigste Zustand und besitzt das Maximum an erreichbaren Teilzielen von $l = 3$. Die Anzahl an unerreichten Teilzielen ergibt sich also aus $n = m - l = 6 - 3 = 3$. Der Inkonsistenzwert für das zweite Planungsproblem Π' wächst damit an, da es weiter von einem konsistenten Planungsproblem entfernt ist als das Planungsproblem Π.

$$I_{Teilziele}(\Pi') = \frac{6-3}{6} = \frac{1}{2} = 0,5 \tag{8}$$

In Fällen, in denen das Ziel nur aus einem Teilziel besteht, also $m = 1$, kann dieses Maß nur auf 0 oder 1 abbilden.

4.2.2 Maß der fehlenden Atome

Wenn ein Teilziel durch mehrere Atome zu erreichen ist, dann sollte der Inkonsistenzwert kleiner sein, je weniger Atome zum Erreichen dieses Teilziels fehlen. Dabei werden so lange fehlende Atome hinzugefügt, bis das Planungsproblem konsistent wird. Die Anzahl an fehlenden Atomen, um das Planungsproblem konsistent zu machen, bestimmt dann den Inkonsistenzwert.

Von einem günstigen Zustand aus werden die Atome, die in den Operationen nicht erfüllt sind und nicht im Ziel gegeben sind, in den Startzustand hinzugefügt. Anschließend wird das Planungsproblem mit dem neuen Startzustand ausgeführt. Dies wiederholt sich so lange, bis eine Pseudo-Lösung für das ursprüngliche Planungsproblem gefunden wurde. Auf der Pseudo-Lösung wird die Anzahl an benötigten Atome v anhand der Anzahl an Atomen in den Vorbedingungen bestimmt.

Definition 38 (Anzahl benötigter Atome)**.** *Die Anzahl aller benötigten Atome v ist die Summe der Anzahl an Atomen in den Vorbedingungen der Operationen der Pseudo-Lösung.*

Durch die Pseudo-Lösung ist bekannt welche Operation als nächstes ausgeführt werden muss. Ist die Operation nicht ausführbar, dann wird die Anzahl der nicht erfüllten Atome der Vorbedingung aufaddiert.

Definition 39 (Anzahl fehlender Atome)**.** *Die Anzahl an fehlenden Atomen u berechnet sich aus der Anzahl an nicht erfüllten Atomen in den Vorbedingungen der Operationen der Pseudo-Lösung.*

Dadurch werden Atome, die für diese Pseudo-Lösung irrelevant sind, nicht mitberechnet. Abschließend berechnet sich das Maß der fehlenden Atome durch die Division der Anzahl an fehlenden Atomen mit der Anzahl aller benötigten Atome. Ist in einem Planungsproblem selbst durch die Hinzunahme von Atomen die Konsistenz nicht zu erreichen, dann ist die Anzahl an fehlenden Atomen und die der benötigten Atome undefiniert und wird auf eine 1 abgebildet. Dazu darf das Maß nur angewandt werden, wenn die Anzahl an benötigten Atome größer 0 ist, um eine Division durch 0 zu verhindern.

Definition 40 (Maß der fehlende Atome)**.** *Das Maß der fehlenden Atome $I_{FehlendeAtome}$ berechnet sich durch die Division der Anzahl an fehlenden Atome u mit der Anzahl aller benötigten Atome v.*

$$I_{FehlendeAtome} : \Pi \to \mathbb{R}^{\geq 0}$$

$$I_{FehlendeAtome}(\Pi) = \begin{cases} \frac{\textit{minimale Anzahl fehlender Atome}}{\textit{Anzahl aller benötigten Atome}} = \frac{u}{v}, & \textit{für } 0 \leq u \leq v \\ 1, & \textit{sonst} \end{cases}$$

Werden im gleichen Schritt zwei unterschiedliche Pseudo-Lösungen gefunden, dann wird das Planungsproblem auf den geringsten Inkonsistenzwert abgebildet.

Im konsistenten Fall bildet das Maß der fehlenden Atome $I_{FehlendeAtome}$ auf eine 0 ab, da bereits alle Atome zur Lösung des Planungsproblems vorhanden sind, also $u = 0$. Hingegen ist $u = v$, wenn kein einziges Atom des Startzustandes zur Lösung des Planungsproblems beiträgt. Das bedeutet, dass alle benötigten Atome erst schrittweise zum Startzustand hinzugefügt werden müssen.

$$\begin{aligned} I_{FehlendeAtome}(\Pi) &= \frac{u}{v} = \frac{0}{v} = 0, \text{ für } u = 0 \text{ und } v > 0 \\ I_{FehlendeAtome}(\Pi) &= \frac{u}{v} = \frac{v}{v} = 1, \text{ für } u = v \text{ und } v > 0 \end{aligned} \tag{9}$$

Beispiel 14. *Sei eine Firma gegeben, die verschiedene Produkte herstellt. Die Abbildung 5 stellt die Produktionsabhängigkeiten grafisch dar. Die Zahl an einem Pfeil steht für die erforderliche Anzahl des Ausgangsproduktes. Durchgehende Pfeile stehen für ein Muss zur Herstellung des Produktes. Sind die Pfeile gestrichelt, dann ist es möglich das Produkt durch diesen oder einen anderen Weg herzustellen. Somit kann das Produkt D entweder durch Produkt B oder durch Produkt K entstehen. Insgesamt gibt es drei Endprodukte, Produkt G, H und J. Die Operationen für dieses Planungsproblem $\Pi = \langle O, s_I, g \rangle$ sind in Listing 27 aufgeführt.*

```
1 produceB():
2     C: a>=5
3     E: a-=5 AND b+=1
4 produceC():
5     C: b>=2
```

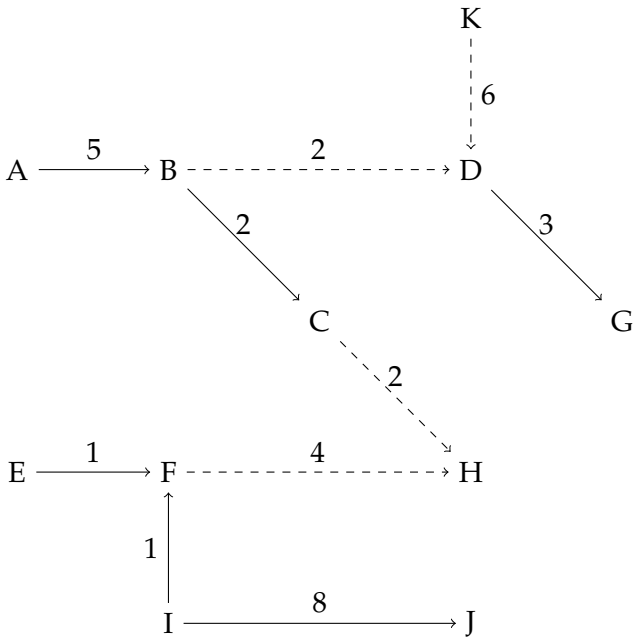

Abbildung 5: Diese Abbildung zeigt die Abhängigkeiten zwischen Produkten. Die Zahl an einem Pfeil steht jeweils für die erforderliche Anzahl des Ausgangsproduktes.

```
    E: b-=2 AND c+=1
produceD():
    C: b>=2 OR k>=6
    E: d+=1 AND (b-=2 OR k-=6)
produceF():
    C: e>=1 AND i>=1
    E: e-=1 AND f+=1 AND i-=1
produceG():
    C: d>=3
    E: d-=3 AND g+=1
produceH():
    C: c>=2 OR f>=4
    E: (c-=2 OR f-=4) AND h+=1
produceJ():
    C: i>=8
    E: i-=8 AND j+=1
```

Listing 27: Operationen des Planungsproblems Π

Weiterhin ist die Problemspezifikation des Planungsproblems $\Pi = \langle O, s_I, g \rangle$ *in Listing 28 gegeben.*

```
init:
    a=20
    e=3
    i=8
    k=6
goal:
    g==1
    h==1
    j==1
```

Listing 28: Problemspezifikation des Planungsproblems Π

Das Produkt H aus dem Ziel g kann nicht hergestellt werden. Dadurch ist das Planungsproblem Π *inkonsistent. Auf das Maß der Teilziele angewandt, wird dieses Beispiel auf einen Inkonsistenzwert von* $0,33$ *abgebildet.*

$$I_{Teilziele}(\Pi) = \frac{\textit{Anzahl unerreichter Teilziele}}{\textit{Anzahl aller Teilziele}} = \frac{1}{3} \approx 0,33 \tag{10}$$

Selbst wenn sich der Startzustand ändert und keine weiteren Teilziele erreicht werden, dann bleibt der Inkonsistenzwert bezüglich der Teilziele unverändert. Hingegen kann sich der Inkonsistenzwert durch das Maß der fehlenden Atome verringern oder anwachsen. Seien dazu die Problemspezifikationen aus Listing 29 mit dem Planungsproblem Π' *und Listing 30 mit dem Planungsproblem* Π'' *gegeben, die sich zum Planungsproblem* Π *nur im Startzustand unterscheiden. Alle drei Planungsprobleme bilden auf den gleichen Inkonsistenzwert ab, wenn das Maß der Teilziele verwendet wird.*

```
init:
    e=4
    i=10
    k=18
goal:
    g==1
    h==1
    j==1
```

Listing 29: Problemspezifikation des Planungsproblems Π'

```
init:
    i=8
    k=18
goal:
    g==1
    h==1
    j==1
```

Listing 30: Problemspezifikation des Planungsproblems Π''

Jedoch ist ersichtlich, dass im Startzustand des Planungsproblems Π' weniger Produkte fehlen um das Ziel zu erreichen, wie im Startzustand des Planungsproblems Π''. Somit berechnet das Maß der fehlende Atome einen kleineren Inkonsistenzwert für das Planungsproblem Π'.

$$
\begin{aligned}
I_{FehlendeAtome}(\Pi) &= \frac{u}{v} = \frac{5}{42} \approx 0,12 \\
I_{FehlendeAtome}(\Pi') &= \frac{u}{v} = \frac{2}{34} \approx 0,059 \\
I_{FehlendeAtome}(\Pi'') &= \frac{u}{v} = \frac{8}{34} \approx 0,235
\end{aligned}
\tag{11}
$$

Im Planungsproblem Π ist die geringste Anzahl an fehlenden Atomen $u = 5$, da für die Herstellung von Produkt H nur einmal Produkt E fehlt und 4 mal das Produkt I. Hingegen würde der Weg über die Produkte A->B->C->H insgesamt 20 mal das Produkt A benötigen. Insgesamt werden in dieser Pseudo-Lösung 42 Atome benötigt. Im Vergleich zum Planungsproblem Π' variiert die Anzahl an benötigten Atomen. Dies kommt daher, dass die Berechnung der Anzahl der benötigten Atome auf der Pseudo-Lösung erfolgt, der sich pro Problemspezifikation ändern kann. Für die beiden Planungsprobleme Π' und Π'' werden insgesamt nur $v = 34$ Atome benötigt. Für das Planungsproblem Π' fehlt das Produkt I zwei mal, also $u = 2$ und im Planungsproblem Π'' werden die Produkte E und I jeweils vier mal benötigt, was sich auf $u = 4 + 4 = 8$ addiert. Da im Planungsproblem Π' nur noch 2 Atome fehlen anstatt 8, erhält dieses einen kleineren Inkonsistenzwert als das Planungsproblem Π''. Damit ist das Planungsproblem Π' im Vergleich zu den Planungsproblemen Π und Π'' am wenigsten inkonsistent.

Theorem 2. *Besitzt ein günstiger Zustand s_{F1} mehr benötigte Atome, wie ein anderer günstiger Zustand s_{F2}, dann führt der günstige Zustand s_{F1} als erstes zu einer Pseudo-Lösung.*

Beweis. Das Planungsproblem wird schrittweise mit fehlenden Atomen erweitert. Da die Anzahl an fehlenden Atomen u' im Zustand s_{F1} geringer ist als die Anzahl an fehlenden Atomen u'' des Zustands s_{F2}, benötigt der Zustand s_{F1} weniger Schritte um das Ziel zu erreichen. Ist eine Pseudo-Lösung über den Zustand s_{F1} bereits gefunden, dann bricht der Algorithmus bereits ab, bevor der Zustand s_{F2} einen Zielzustand erreichen konnte. □

Beispiel 15. *Dies ist im Planungsproblem Π des Beispiels 14 zu sehen. Der Pseudo-Pfad um das Produkt H zu produzieren, kann mit dem einmaligen Hinzufügen des Produktes E und 4 mal dem Produkt I realisiert werden. Hingegen braucht der Pseudo-Pfad über das Produkt A 20 Durchläufe um das Produkt H zu realisieren. Da das Produkt H jedoch schon durch den Pseudo-Pfad mit Produkt E und Produkt I vorher erzeugt werden kann, bricht der Algorithmus bereits ab.*

Werden für die Anzahl an fehlenden Atomen u nicht relevante Atome der Pseudo-Lösung mit eingerechnet, dann kann der Inkonsistenzwert größer als 1 werden. Dadurch, dass die Berechnung der beiden Größen auf der gleichen Pseudo-Lösung geschieht, ist gewährleistet, dass $0 \leq u \leq v$ gilt.

4.2.3 Maß der Vorbedingungen

Anstatt die Anzahl an fehlenden Atomen zu berechnen, ist es möglich von einem günstigen Zustand aus alle Operationen auszuführen ohne die Vorbedingungen zu berücksichtigen. Dies wird sooft wiederholt, bis eine Pseudo-Lösung gefunden wurde.

Beginnend bei allen günstigen Zuständen werden die Operationen ausgeführt, deren Vorbedingung zuvor fehlgeschlagen sind. Die Vorbedingungen werden für diese Operationen ignoriert. Kann ein Zielzustand erreicht werden, dann bricht der Algorithmus ab. Ansonsten werden auf den erzeugten Zuständen ebenfalls alle Operationen ausgeführt. Dies wird so lange wiederholt, bis ein Zielzustand erreicht werden kann. Werden die Zustandsänderungen der Pseudo-Lösung auf den Startzustand angewandt, kann die Anzahl an ignorierten Vorbedingungen x bestimmt werden. Für jede Zustandsänderung, bei der die Vorbedingung fehlschlägt wird die Anzahl der ignorierten Vorbedingungen erhöht bis das Ziel erreicht ist.

Definition 41 (Anzahl ignorierter Vorbedingungen)**.** *Die Anzahl der ignorierten Vorbedingungen x auf einem Pseudo-Pfad p entspricht der Anzahl an Zustandsänderungen, bei denen die Vorbedingung einer Operation fehlschlägt.*

Um die Anzahl an ausgeführten Operationen y auf der Pseudo-Lösung zu berechnen, wird die Anzahl an Zustandsänderungen verwendet.

Definition 42 (Anzahl ausgeführter Operationen)**.** *Die Anzahl der ausgeführten Operationen y auf einem Pfad p entspricht der Anzahl an Zustandsänderungen des Pfades p.*

Das Maß der Vorbedingungen $I_{Vorbedingung}$ berechnet sich anschließend durch die Division der Anzahl ignorierter Vorbedingungen x mit der Anzahl aller ausgeführten Operationen y der Pseudo-Lösung. Kann das Planungsproblem jedoch nicht zur Konsistenz erlangen, selbst wenn alle Vorbedingungen ignoriert werden, dann sind sowohl x als auch y unbestimmt. In diesem Fall bildet das Maß auf eine 1 ab. Ebenso muss mindestens eine Operation ausgeführt werden, da sonst eine Division durch 0 folgt. Daher darf das Maß nicht angewandt werden, wenn der Startzustand bereits ein Zielzustand ist.

Definition 43 (Maß der Vorbedingungen)**.** *Das Maß der Vorbedingungen $I_{Vorbedingung}$ berechnet sich aus der Division der Anzahl an ignorierten Vorbedingungen x durch die*

Anzahl an ausgeführten Operationen y der Pseudo-Lösung.

$$I_{Vorbedingung} : \Pi \rightarrow \mathbb{R}^{\geq 0} :$$
$$I_{Vorbedingung}(\Pi) = \begin{cases} \frac{\textit{Anzahl ignorierter Vorbedingungen}}{\textit{Anzahl ausgeführter Operationen}} = \frac{x}{y} & \textit{für } 0 \leq x \leq y \\ 1, & \textit{sonst} \end{cases}$$

Werden im gleichen Schritt zwei unterschiedliche Pseudo-Lösungen gefunden, dann entspricht das Maß der Vorbedingung dem geringeren Inkonsistenzwert.

Ist ein Planungsproblem konsistent, dann konnte ein Zielzustand erreicht werden, ohne dass Vorbedingungen ignoriert werden mussten. Damit ist in diesem Fall $x = 0$. Hingegen ist $x = y$, wenn alle Vorbedingungen zum Erreichen eines Zielzustandes ignoriert werden mussten. Dann gilt das Planungsproblem als maximal inkonsistent für das Maß der Vorbedingungen.

$$\begin{aligned} I_{Vorbedingung}(\Pi) &= \frac{x}{y} = \frac{0}{y} = 0, \text{ für } x = 0 \text{ und } y > 0 \\ I_{Vorbedingung}(\Pi) &= \frac{x}{y} = \frac{y}{y} = 1, \text{ für } x = y \text{ und } y > 0 \end{aligned} \tag{12}$$

Beispiel 16. *Sei ein Straßennetz gegeben, welches Tankstellen beinhaltet. Ein Auto hat eine Tankfüllung, die für 200 Kilometer ausreicht. Sowohl in Städten als auch an Tankstellen kann getankt werden. Wenn ein Ort oder eine Tankstelle erreicht wird, füllt sich der Tank automatisch auf. Die Operation des Planungsproblems $\Pi = \langle O, s_I, g \rangle$ ist Listing 31 zu entnehmen. Listing 32 definiert die Problemspezifikation mit dem Startzustand und dem Ziel. Dabei soll mit einem Auto von Frankfurt nach Berlin gefahren werden. Zwischen dieser Strecke gibt es eine Tankstelle, die von Frankfurt 200 Kilometer entfernt liegt und zu Berlin eine Entfernung von 350 Kilometern hat. Die Abbildung 6 visualisiert das Straßennetz mit den jeweiligen Entfernungen zwischen zwei Orten. Sind zwei Orte miteinander verbunden, dann existieren dort befahrbare Straßen.*

```
1 driveTo(FROM,TO):
2    precondition: connection(FROM,TO)<=200
3    effect: at(vehicle,TO) AND not at(vehicle,FROM)
```

Listing 31: Operation *driveTo* des Straßennetz-Beispiels

```
1 objects:
2    car - vehicle
3    frankfurt berlin munich gasstation1 gasstation2 gasstation3 - location
4    connection(frankfurt,berlin)=550
5    connection(frankfurt,munich)=400
6    connection(frankfurt,gasstation1)=200
7    connection(frankfurt,gasstation2)=250
8
```

```
    connection(berlin,frankfurt)=550
    connection(berlin,munich)=600
    connection(berlin,gasstation1)=350
    connection(berlin,gasstation3)=300

    connection(munich,frankfurt)=400
    connection(munich,berlin)=600
    connection(munich,gasstation2)=150
    connection(munich,gasstation3)=300

    connection(gasstation1,berlin)=350
    connection(gasstation1,frankfurt)=200
    connection(gasstation2,frankfurt)=250
    connection(gasstation2,munich)=150
    connection(gasstation3,munich)=300
    connection(gasstation3,berlin)=300

init:
    at(car,frankfurt)
goal:
    at(car,berlin)
```

Listing 32: Problemspezifikation des Straßennetz-Beispiel in vereinfachter PDDL-Notation

Das Planungsproblem Π *ist nicht lösbar, da der Tank niemals bis zur nächsten Stadt reicht. Selbst das Maß der fehlenden Atome kann durch die Hinzunahme von Atomen nicht das Ziel erreichen. Das Ergebnis der Messung mit dem Maß der fehlenden Atome beträgt damit einer 1.*

$$I_{FehlendeAtome}(\Pi) = 1 \tag{13}$$

Die Operation driveTo von Frankfurt zur Tankstelle1 kann ausgeführt werden. Die zweite Wegstrecke ist größer als 200 Kilometer und daher ist die Ausführung der Operation driveTo nicht möglich. Wird nun die Vorbedingung ignoriert, um von der Tankstelle1 nach Berlin zu kommen, dann ist es möglich ein konsistentes Planungsproblem zu erhalten. Daher wird anschließend die Vorbedingung der Operation driveTo ignoriert und die Anzahl an ignorierten Vorbedingungen erhöht sich um 1. Nach dem zweiten Ausführen der Operation driveTo ist das Ziel erreicht. Somit ist die Anzahl an ausgeführten Operationen $y = 2$*. Insgesamt wurde davon die Vorbedingung von einer Operation ignoriert, also* $x = 1$*. Das Maß durch Vorbedingungen liefert daher für das Planungsproblem* Π *einen Inkonsistenzwert von* $0,5$*.*

$$I_{Vorbedingung}(\Pi) = \frac{x}{y} = \frac{1}{2} = 0,5 \tag{14}$$

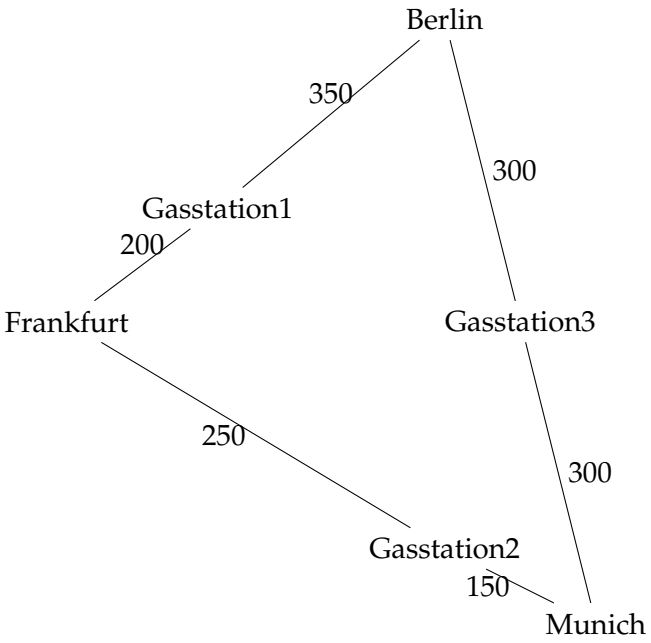

Abbildung 6: Straßennetz-Beispiel visualisiert

Wird der Startzustand geändert und das Auto startet in München, siehe Listing 33, dann ändert sich das Planungsproblem zu $\Pi' = \langle O, s_{I'}, g \rangle$ ab. Dann ist das Ausführen der Operation driveTo, um von München zur Tankstelle 3 und von dort nach Berlin zu gelangen, in beiden Fällen nicht möglich.

```
1 init:
2    at(car,frankfurt)
3 goal:
4    at(car,berlin)
```

Listing 33: Startzustand und Ziel des Planungsproblem Π' für das Straßennetz-Beispiel

Um den Weg München -> Tankstelle 3 -> Berlin befahren zu können, muss zwei Mal die Vorbedingung der Operation driveTo ignoriert werden. Hingegen könnte der Weg München -> Tankstelle2 -> Frankfurt -> Tankstelle 1 -> Berlin durch 4 Operationen gelöst werden, wovon nur in zwei Operationsausführungen die Vorbedingung ignoriert werden muss. Die zweite Variante würde zwar einen kleineren Inkonsistenzwert liefern, jedoch bricht der Algorithmus bereits ab bevor die zweite Variante die Stadt Berlin erreichen würde. Dies kommt daher, dass im zweiten Schritt der Suche nach einer Pseudo-Lösung Berlin bereits erreicht wird und der Weg über Frankfurt nicht weiter betrachtet wird. Der Inkonsistenzwert für das Planungsproblem Π' beläuft sich damit auf eine 1.

$$I_{Vorbedingung}(\Pi') = \frac{x}{y} = \frac{2}{2} = 1 \tag{15}$$

4.2.4 Maß der kritischen Atome

Im Beispiel der Produktionsfirma aus Kapitel 4.2.2 hat das Produkt B eine höhere Wertigkeit, da es zur Herstellung von den Produkten D und C genutzt werden kann. Das Gleiche gilt für das Produkt I, welches für F und J benötigt wird. Diese Produkte sollten bei der Berechnung des Inkonsistenzwertes stärker ins Gewicht fallen, da sie kritischer sind als die anderen Atome.

Dafür muss zunächst die Wertigkeit pro Atom bestimmt werden. Sind diese gegeben, dann wird der Inkonsistenzwert ähnlich dem Maß der Vorbedingungen berechnet. Von allen günstigen Zuständen aus werden alle Operationen ausgeführt, auch wenn die Vorbedingung nicht erfüllt ist. Dabei wird eine Summe über die Wertigkeiten der Atome in der Vorbedingung pro Operation gebildet. Ist ein Atom nicht erfüllt, dann wird dessen Wertigkeit addiert. Anschließend wird durch die Anzahl an Atomen in der Vorbedingung dividiert. Somit wird pro Operation die Ausführbarkeit auf einen Zustand bestimmt. Eine ausführbare Operation wird dadurch mit einer 0 dargestellt und eine nicht ausführbare Operation kann maximal einen Wert von 1 erreichen.

Definition 44 (Wertigkeit einer Operation)**.** *Die Funktion f zur Bestimmung der Wertigkeit einer Operation berechnet sich aus der Summe der Wertigkeiten der nicht erfüllten Atome α der Vorbedingung c auf einen Zustand und dividiert diese Summe durch die Anzahl an vorhandenen Atomen t in der Vorbedingung.*

$$f : O \to \mathbb{R}^{\geq 0}$$
$$f(o) = \frac{\sum^{\alpha \in c} \omega(\alpha)}{t}$$

Die Berechnung des Maßes der kritischen Atome erfolgt abschließend auf einer Pseudo-Lösung. Der Inkonsistenzwert ist die Summe der Wertigkeiten der Operationen der Pseudo-Lösung dividiert mit der Anzahl ausgeführter Operationen. Um eine Division durch 0 zu vermeiden, muss das Planungsproblem mindestens eine Operation ausführen. Das bedeutet, dass der Startzustand nicht das Ziel erfüllen darf.

Definition 45 (Maß durch kritische Atome)**.** *Das Maß der kritischen Atome $I_{KritischeAtome}$ berechnet sich aus der Division der Summe der Wertigkeit einer Operation $f(o)$ durch die Anzahl ausgeführter Operationen y einer Pseudo-Lösung.*

$$I_{KritischeAtome} : \Pi \to \mathbb{R}^{\geq 0}$$
$$I_{KritischeAtome}(\Pi) = \frac{\sum^{o \in O} f(o)}{\textit{Anzahl ausgeführter Operationen}} = \frac{s}{y},$$
$$\textit{für } 0 \leq s \leq y$$

Tritt ein Atom in mehreren Vorbedingungen auf, dann ist der Grad der Inkonsistenz bezüglich dieses Atoms potentiell größer. Dazu hat das Atom eine Wertigkeit

a	b	c	d	e	f
$\frac{1}{1}$	$\frac{2}{3}$	$\frac{3}{3}$	$\frac{2}{3}$	$\frac{1}{2}$	$\frac{0}{2}$

Abbildung 7: Wertigkeiten der Atome aus den Operationen

von 1, wenn es in Vorbedingungen vorkommt, jedoch in keinem Effekt. Dadurch kann es nicht durch einen Effekt erzeugt werden und muss im Startzustand spezifiziert werden, um in einer Vorbedingung erfüllt zu sein. Eine Operation hat eine Wertigkeit von 1, wenn alle Atome eine Wertigkeit von einer 1 besitzen. Damit wird keines der Atome in einem Effekt erzeugt.

Findet der Algorithmus mehrere Pseudo-Lösungen im gleichen Schritt, dann bildet das Maß auf den geringsten berechneten Wert ab.

Das Maß bildet auf eine 0 ab, wenn das Planungsproblem konsistent ist. Im Maß wird somit die Summe über die Wertigkeiten aller Operationen gebildet, welche alle 0 sind. Dadurch ist $s = 0$. Für den maximal inkonsistenten Fall ist die Summe der Wertigkeiten einer Operation gleich der Anzahl an ausgeführten Operation: $s = y$. Dies ist nur der Fall, wenn jede ausgeführte Operation eine Wertigkeit von 1 hat.

$$\begin{aligned} I_{KritischeAtome}(\Pi) &= \frac{s}{y} = \frac{0}{y} = 0, \text{ für } s = 0 \text{ und } y > 0 \\ I_{KritischeAtome}(\Pi) &= \frac{s}{y} = \frac{y}{y} = 1, \text{ für } s = y \text{ und } y > 0 \end{aligned} \tag{16}$$

Beispiel 17. *Sei dazu ein Planungsproblem $\Pi = \langle O, s_I, g \rangle$ mit 3 Operationen aus Listing 34 gegeben. Der dazugehörige Startzustand und das Ziel ist in Listing 35 definiert.*

```
1 func1():
2     C: a, not c
3     E: not b, e+=2
4 func2():
5     C: b, c, d
6     E: f+=1
7 func3():
8     C: b, e>=4, c, d
9     E: d, f+=1
```

Listing 34: Operationen des Planungsproblems Π

Zunächst wird die Wertigkeit pro Atom in den Vorbedingungen berechnet, siehe Abbildung 7. Normalerweise werden die Wertigkeiten für jeden Zustand neu berechnet, doch für dieses Beispiel bleiben die Wertigkeiten der Atome bei jedem Zustand gleich.

```
1 init:
2     a, c, d, e=0, f=0
3 goal:
```

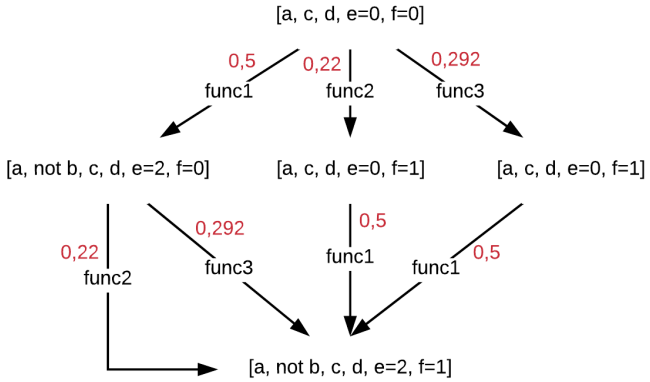

Abbildung 8: Diese Abbildung zeigt die 4 Pseudo-Lösungen im Planungsprobelm Π

```
4    a, c, not b, d, e=2, f=1
```

Listing 35: Startzustand und Ziel des Planungsproblems Π

Insgesamt können vier Pseudo-Lösungen in einem Schritt gefunden werden. Diese vier Pseudo-Lösungen sind in der Abbildung 8 dargestellt. Da das Ziel in allen vier Pseudo-Lösungen in zwei Schritten erreicht werden kann, ist die Anzahl an ausgeführten Operation $y = 2$.

Die rot dargestellten Zahlen aus der Abbildung 8 zeigen die Wertigkeiten der Operation auf den jeweiligen Zustand an. Die Berechnungen der Pseudo-Pfade sind dabei wie folgt:

$$\begin{aligned}
&\textit{1. Pseudo-Pfad: } \frac{\frac{1}{2}+\frac{2}{9}}{2} \approx 0,361\\
&\textit{2. Pseudo-Pfad: } \frac{\frac{1}{2}+\frac{7}{24}}{2} \approx 0,396\\
&\textit{3. Pseudo-Pfad: } \frac{\frac{2}{9}+\frac{1}{2}}{2} \approx 0,361\\
&\textit{4. Pseudo-Pfad: } \frac{\frac{7}{24}+\frac{1}{2}}{2} \approx 0,396
\end{aligned} \tag{17}$$

In der vierten Pseudo-Lösung wird zunächst die Operation func3 ausgeführt. Da die Operation sowohl für das Atom b als auch für das Atom e fehlschlägt, werden deren Wertigkeiten addiert. Um anschließend die Wertigkeit der Operation zu erhalten, wird durch

die Anzahl an Atomen in der Vorbedingung der Operation func3, also mit 4, dividiert. Damit hat die Operationfunc3 eine Wertigkeit von ungefähr $0,292$*. Die nächste Operation ist func1, bei der nur das Atom not c nicht erfüllt ist. Da das Atom c eine Wertigkeit von 1 hat und die Anzahl an Atomen in der Vorbedingung der Operation func1 2 ist, ist die Wertigkeit für diese Operation* $0,5$*. Da anschießend ein Zielzustand erreicht wurde, wird der Inkonsistenzwert durch die Summe der Wertigkeiten der beiden Operationen func3 und func1 durch die Anzahl an ausgeführten Operationen, also* $y = 2$ *dividiert.*

Dabei entspricht das Maß dem kleinsten berechneten Wert. Das Planungsproblem Π *wird dadurch auf einen Inkonsistenzwert von ungefähr* $0,361$ *durch das Maß der kritischen Atome* $I_{KritischeAtome}$ *abgebildet. Würde das Maß der Vorbedingungen* $I_{Vorbedingung}$ *auf das Planungsproblem* Π *angewandt, dann wäre der Inkonsistenzwert eine 1.*

$$I_{Vorbedingung}(\Pi) = \frac{x}{y} = \frac{2}{2} = 1 \tag{18}$$

Da x *die Anzahl an ignorierten Vorbedingungen ist und alle Vorbedingungen fehlschlagen, ist die Anzahl an ignorierten Vorbedingungen gleich der Anzahl an ausgeführten Operationen:* $x = y$*.*

Dieses Maß kann das gleiche Ergebnis wie das Maß durch Vorbedingungen erzielen. Dies ist der Fall, wenn alle Atome eine Wertigkeit von 1 haben und jede Operation entweder auf eine 0 oder 1 abgebildet wird.

5 Eigenschaften der Inkonsistenzmaße

Die Inkonsistenzmaße aus Kapitel 4.2 erfüllen nicht alle Eigenschaften aus dem Kapitel 4.1. Nachfolgend werden für jedes Maß die Eigenschaften überprüft.

5.1 Maß der Teilzielen

Das Maß an unerreichten Teilziele $I_{Teilziele}$ für ein Planungsproblem Π ist definiert durch:

$$\begin{aligned} &I_{Teilziele} : \Pi \to \mathbb{R}^{\geq 0} \\ &I_{Teilziele}(\Pi) = \frac{\text{Anzahl unerreichter Teilziele}}{\text{Anzahl aller Teilziele}} = \frac{n}{m}, \\ &\text{für } 0 \leq n \leq m \end{aligned}$$

Es erfüllt die Eigenschaften der Normalisierung, der Monotonie im Startzustand und des Ziels, der Monotonie in einer Vorbedingung einer Operation, der Unabhängigkeit eines Atoms im Startzustand, der Unabhängigkeit eines Atoms in einer Vorbedingung einer Operation und der Dominanz.

Das Maß $I_{Teilziele}$ erfüllt die Eigenschaft der Unabhängigkeit im Ziel nicht.

5.1.1 Normalisierung

Theorem 3. *Sei ein Planungsproblem* $\Pi = \langle O, s_I, g \rangle$ *gegeben. Laut der Normalisierung gilt:*

$$0 \leq I_{Teilziele}(\Pi) \leq 1$$

Beweis. Ist das Planungsproblem Π konsistent, dann ist die Anzahl der unerreichten Teilziele $n = 0$. Dann berechnet das Maß $I_{Teilziele}$ einen Wert von 0.

$$I_{Teilziele}(\Pi) = \frac{n}{m} = \frac{0}{m} = 0, \text{ für } n = 0 \text{ und } m > 0 \tag{19}$$

Ist hingegen kein einziges Teilziel erreichbar, dann ist die Anzahl der unerreichten Teilziele gleich der Anzahl aller Teilziele, also $n = m$. Dieser Fall wird im Maß durch Teilziele $I_{Teilziele}$ immer auf eine 1 abgebildet.

$$I_{Teilziele}(\Pi) = \frac{n}{m} = \frac{m}{m} = 1, \text{ für } n = m \text{ und } m > 0 \tag{20}$$

In allen weiteren Fälle liegt n laut Definition immer zwischen einem Wert von 0 und m. Somit gilt:

$$0 < I_{Teilziele}(\Pi) = \frac{n}{m} < 1, \text{ für } 0 < n < m \tag{21}$$

Somit gilt die Eigenschaft der Normalisierung für das Maß der Teilziele $I_{Teilziele}$. □

5.1.2 Monotonie

Theorem 4. *Sei ein Atom* α *und ein Planungsproblem* $\Pi = \langle O, s_I, g \rangle$ *gegeben. Laut der Monotonie-Eigenschaft für den Startzustand gilt folgendes:*

$$I_{Teilziele}(\Pi) \geq I_{Teilziele}(\Pi'), \textit{ wobei } \Pi' = \langle O, s_I \cup \{\alpha\}, g \rangle$$

Beweis. Ist das Planungsproblem Π konsistent, dann bildet das Maß $I_{Teilziele}$ auf eine 0 ab. Durch das zusätzliche Atom α im Startzustand ändert sich jedoch nichts an der Konsistenz des Planungsproblems, da die Atome, die das Planungsproblem Π konsistent gemacht haben, immer noch vorhanden sind. Damit bleibt die Konsistenz bestehen. Es gilt $n' = n = 0$ als auch $m' = m$.

$$\begin{aligned} I_{Teilziele}(\Pi) &= \frac{n}{m} = 0, \text{ für } n = 0 \text{ und } m > 0 \\ I_{Teilziele}(\Pi') &= \frac{n'}{m'} = \frac{0}{m} = 0, \text{ für } n' = 0 \text{ und } m' = m \end{aligned} \tag{22}$$

Können keine Teilziele erreicht werden, dann ist das Planungsproblem Π maximal inkonsistent bezüglich des Maßes $I_{Teilziele}$. Dadurch wird das Planungsproblem

Π auf eine 1 abgebildet. Wird hier ein Atom α in den Startzustand hinzugefügt, dann kann sich die Inkonsistenz verringern, da die Möglichkeit besteht, dass ein Teilziel gelöst werden kann. Daher ist die Anzahl an unerreichten Teilzielen entweder gleich geblieben $n' = m$ oder um eins verringert: $n' = m - 1$.

$$\begin{aligned} I_{Teilziele}(\Pi) = \frac{n}{m} = \frac{m}{m} &= 1, \text{ für } n = m \text{ und } m > 0 \\ I_{Teilziele}(\Pi') = \frac{n'}{m'} = \frac{m}{m} &= 1, \text{ für } n' = m \text{ und } m' = m \\ I_{Teilziele}(\Pi') = \frac{n'}{m'} = \frac{m-1}{m} &< 1, \text{ für } n' = m - 1 \text{ und } m' = m \end{aligned} \tag{23}$$

In den Fällen, in denen die Anzahl an unerreichten Teilzielen des Planungsproblems Π zwischen 0 und m liegt, ist der Inkonsistenzwert zwischen 0 und 1. Im Planungsproblem Π' könnte durch das neue Atom α im Startzustand ein Teilziel mehr als zum Planungsproblem Π gelöst werden. Dadurch ist die Anzahl an unerreichten Teilzielen des Planungsproblems Π' entweder gleich des Planungsproblems Π, also $n' = n$, oder um eins verringert: $n' = n - 1$.

$$\begin{aligned} 0 < I_{Teilziele}(\Pi) &< 1, \text{ für } 0 < n < m \text{ und } m > 0 \\ 0 < I_{Teilziele}(\Pi') = I_{Teilziele}(\Pi) &< 1, \text{ für } n' = n \text{ und } m' = m \\ 0 < I_{Teilziele}(\Pi') < I_{Teilziele}(\Pi) &< 1, \text{ für } n' = n - 1 \text{ und } m' = m \end{aligned} \tag{24}$$

Somit verringert sich die Inkonsistenz oder bleibt gleich, wenn ein Atom α zum Startzustand hinzugefügt wird. Die Eigenschaft der Monotonie im Startzustand ist für das Maß der Teilziele $I_{Teilziele}$ damit erfüllt. □

Theorem 5. *Sei ein Atom α und ein Planungsproblem $\Pi = \langle O, s_I, g \rangle$ gegeben. Ist das Atom α nicht vom Startzustand aus erreichbar und wird dem Ziel hinzugefügt, dann gilt:*

$$I_{Teilziele}(\Pi) \geq I_{Teilziele}(\Pi')$$
$$\textit{wobei } \Pi' = \langle O, s_I, g \cup \{\alpha\} \rangle \textit{ und } \alpha \textit{ ist nicht erreichbar}$$

Beweis. Das Planungsproblem hat m Teilziele, wovon n nicht erreichbar sind. Durch das Maß ist definiert, dass das Ziel nicht leer sein darf $m > 0$ und die Anzahl an unerreichten Teilzielen kann nur zwischen $0 \leq n \leq m$ liegen. Da das zweite Planungsproblem Π' ein Atom mehr im Zielzustand hat, ist $m' = m + 1$. Weiterhin ist das neue Teilziel α nicht erreichbar und somit wächst auch die Anzahl an unerreichten Teilzielen um 1: $n' = n + 1$.

Seien alle Teilziele im Planungsproblem Π erreichbar, dann ist $n = 0$. Wird nun das neue unerreichbare Teilziel α hinzugefügt, dann wächst der Inkonsistenzwert, da n' zu $n' = n + 1 = 1$ wird.

$$
\begin{aligned}
I_{Teilziele}(\Pi) &= \frac{n}{m} = \frac{0}{m} = 0, \text{ für } n = 0 \text{ und } m > 0 \\
I_{Teilziele}(\Pi') &= \frac{n'}{m'} = \frac{1}{m+1} > 0, \text{ für } n' = 1 \text{ und } m' = m+1
\end{aligned}
\tag{25}
$$

Wird nun der maximal inkonsistente Fall betrachtet, also wenn $n = m$ ist, dann liefert das Maß $I_{Teilziele}$ für das Planungsproblem Π eine 1. Auf das Planungsproblem Π' angewandt, bildet das Maß $I_{Teilziele}$ ebenfalls auf eine 1 ab, da das Teilziel unerreichbar ist. Der maximal inkonsistente Fall bleibt damit bestehen, da sowohl n als auch m um eins anwachsen. Da $n = m$ im Planungsproblem Π gilt, wird im Planungsproblem Π' daraus $n' = n + 1 = m + 1 = m'$.

$$
\begin{aligned}
I_{Teilziele}(\Pi) &= \frac{n}{m} = \frac{m}{m} = 1, \text{ für } n = m \text{ und } m > 0 \\
I_{Teilziele}(\Pi') &= \frac{n'}{m'} = \frac{m+1}{m+1} = 1, \text{ für } n' = m+1 \text{ und } m' = m+1
\end{aligned}
\tag{26}
$$

Wird die Inkonsistenz für einen Fall berechnet, in dem $0 < n < m$ gilt, dann liegt der Wert für das Planungsproblem Π zwischen 0 und 1. Die Anzahl unerreichter Teilziele n' aus dem Planungsproblem Π' ist um eins erhöht, da das hinzugefügte Teilziel α unerreichbar ist. Gleichzeitig wächst auch die Anzahl aller Teilziele um 1. Dadurch wächst auch der Inkonsistenzwert für das Planungsproblem Π'.

$$
\begin{aligned}
0 < I_{Teilziele}(\Pi) = \frac{n}{m} < I_{Teilziele}(\Pi') = \frac{n'}{m'} = \frac{n+1}{m+1} < 1, \\
\text{für } 0 < n < m \text{ und } n' = n+1, m' = m+1
\end{aligned}
\tag{27}
$$

Die Eigenschaft der Monotonie für das Ziel gilt somit für das Maß der Teilziele $I_{Teilziele}$, da es für alle Fälle, in denen ein Teilziel α nicht erreichbar ist, bewiesen werden kann. □

Theorem 6. *Sei ein Atom α und ein Planungsproblem $\Pi = \langle O, s_I, g \rangle$ gegeben. Wird das Atom α in eine Vorbedingung einer Operation aufgenommen, dann gilt laut der Monotonie-Eigenschaft einer Vorbedingung folgendes:*

$$
\begin{aligned}
&I_{Teilziele}(\Pi) \geq I_{Teilziele}(\Pi'), \textit{ wenn } o' \textit{ ausführbar ist} \\
&I_{Teilziele}(\Pi) \leq I_{Teilziele}(\Pi'), \textit{ wenn } o' \textit{ nicht ausführbar ist} \\
&\quad \textit{wobei } \Pi' = \langle O', s_I, g \rangle \textit{ mit } o' = \langle c \cup \{\alpha\}, e \rangle \in O'
\end{aligned}
$$

Beweis. Ist das Planungsproblem Π konsistent, dann bildet das Maß $I_{Teilziele}$ auf eine 0 ab. Kann die Operation o' durch die erweiterte Vorbedingung weiterhin ausgeführt werden, dann werden weiterhin alle Teilziele erreicht und somit ist $n' = n = 0$.

Das führt dazu, dass der Inkonsistenzwert gleich 0 ist. Hingegen kann die Inkonsistenz anwachsen, wenn die Operation o' mit der erweiterten Vorbedingung das Ausführen einer Operation verhindert. Ist die Operation für das Erreichen von mindestens einem Teilziel nötig, dann steigt die Anzahl an unerreichten Teilzielen um mindestens 1 und maximal bis m an. Wird die Operation für das Erreichen von keinem Teilziel benötigt, dann bleibt die Anzahl an unerreichten Teilzielen gleich und der Inkonsistenzwert bleibt 0. In dem Fall, dass die Operation o' nicht mehr ausführbar ist, ist die Konsistenz im Planungsproblem Π' nicht mehr gewährleistet und der Inkonsistenzwert kann ansteigen.

$$\begin{aligned} I_{Teilziele}(\Pi) &= \frac{n}{m} = \frac{0}{m} = 0, \text{ für } n = 0 \text{ und } m > 0 \\ I_{Teilziele}(\Pi') &= \frac{n'}{m'} = \frac{0}{m} = 0, \text{ für } n' = 0 \text{ und } m' = m \\ I_{Teilziele}(\Pi') &= \frac{n'}{m'} = \frac{n'}{m} \geq 0, \text{ für } 0 \leq n' \leq m \text{ und } m' = m \end{aligned} \tag{28}$$

Bildet das Maß $I_{Teilziele}$ auf eine 1 ab, dann ist das Planungsproblem Π maximal inkonsistent. Damit sind alle Teilziele unerreichbar: $n = m$. Wird nun das Atom α in eine Vorbedingung einer ausführbaren Operation hinzugefügt, dann ist das Planungsproblem Π' weiterhin maximal inkonsistent. Dies liegt daran, dass die Operationen, die Teilziele erreichen können, immer noch nicht ausführbar sind. Somit ist $n' = m$.

$$\begin{aligned} I_{Teilziele}(\Pi) &= \frac{n}{m} = \frac{m}{m} = 1, \text{ für } n = m \text{ und } m > 0 \\ I_{Teilziele}(\Pi') &= \frac{n'}{m'} = \frac{m}{m} = 1, \text{ für } n' = m \text{ und } m' = m \end{aligned} \tag{29}$$

Für alle weiteren Fälle, in denen $0 < n < m$ gilt, liegt der Inkonsistenzwert für das Planungsproblem Π zwischen 0 und 1. Wird das Atom α in die Vorbedingung einer ausführbaren Operation hinzugenommen und die Operation ist danach nicht mehr ausführbar, dann könnten weniger Teilziele erreicht werden. Somit kann die Anzahl an unerreichten Teilzielen anwachsen: $n + 1 \leq n' \leq m$. Ist die nicht mehr ausführbare Operation jedoch nicht für die erreichten Teilziele in Π' relevant, dann bleibt die Anzahl an unerreichten Teilzielen gleich: $n' = n$. Ist die Operation o' weiterhin ausführbar, dann ändert sich an der Anzahl an unerreichbaren Teilzielen nichts und bleibt gleich: $n' = n$.

$$\begin{aligned} 0 < I_{Teilziele}(\Pi) = \frac{n}{m} = \frac{n'}{m'} = I_{Teilziele}(\Pi') < 1 \\ \text{für } 0 < n < m \text{ und } n' = n, m' = m \\ 0 < I_{Teilziele}(\Pi) = \frac{n}{m} < \frac{n'}{m'} = I_{Teilziele}(\Pi') \leq 1 \\ \text{für } 0 < n < m \text{ und } n + 1 \leq n' \leq m' = m \end{aligned} \tag{30}$$

Somit gilt die Eigenschaft der Monotonie in einer Vorbedingung einer Operation für das Maß der Teilziele $I_{Teilziele}$. □

5.1.3 Unabhängigkeit eines Atoms

Theorem 7. *Sei ein unabhängiges Atom α im Startzustand s_I eines Planungsproblems $\Pi = \langle O, s_I, g\rangle$ gegeben. Dann lässt sich das Planungsproblem zu $\Pi' = \langle O, s_I \setminus \{\alpha\}, g\rangle$ vereinfachen und es gilt:*

$$I_{Teilziele}(\Pi) = \frac{n}{m} = \frac{n'}{m'} = I_{Teilziele}(\Pi')$$

Beweis. Da das Atom α unabhängig ist, ist es entweder in einem Effekt einer ausführbaren Operation vorhanden und wird somit bei der Exploration des Planungsproblems erzeugt oder existiert in keiner Vorbedingung einer Operation. Dadurch variieren die Pfade der Planungsprobleme Π und Π' nur minimal. Ist das Atom in keiner Vorbedingung einer Operation enthalten, dann unterscheiden sich die Zustände lediglich im Atom α. Wird das Atom α hingegen erzeugt, dann können alle Zustände des Planungsproblems Π auch im Planungsproblem Π' erzeugt werden. Somit wird das Atom α nicht im Startzustand des Planungsproblems benötigt, da durch dieses Atom keine weiteren Zustände möglich sind und daher können auch keine weiteren Teilziele erreicht werden. Da im Planungsproblem Π' die gleichen Zustände wie im Planungsproblem Π möglich sind, verringert sich die Anzahl der erreichten Teilziele ebenfalls nicht. Abschließend ändert sich auch die Anzahl an unerreichten Teilzielen nicht: $n' = n$. Der Inkonsistenzwert ist also gleich:

$$I_{Teilziele}(\Pi) = \frac{n}{m} = \frac{n'}{m'} = I_{Teilziele}(\Pi'), \text{ für } 0 \leq n' = n \leq m' = m \tag{31}$$

Die Eigenschaft der Unabhängigkeit eines Atoms im Startzustand des Planungsproblems Π gilt damit für das Maß der Teilziele $I_{Teilziele}$. □

Theorem 8. *Sei ein unabhängiges Atom α im Ziel g eines Planungsproblems $\Pi = \langle O, s_I, g\rangle$ gegeben. Dann lässt sich das Planungsproblem zu $\Pi' = \langle O, s_I, g \setminus \{\alpha\}\rangle$ vereinfachen und es gilt:*

$$I_{Teilziele}(\Pi) = \frac{n}{m} = \frac{n'}{m'} = I_{Teilziele}(\Pi')$$

Beweis. Seien die Operationen des Planungsproblems $\Pi = \langle O, s_I, g\rangle$ aus Listing 36 gegeben. Der Startzustand und das Ziel beschreibt Listing 37.

```
1 func1():
2     C: a, b
3     E: c
4 func2(): c, d
5     E: e
```

```
6 func2(): d, h
7     E: f
```

Listing 36: Operationen des Planungsproblems Π

```
1 init:
2     a, b
3 goal:
4     c, d, f
```

Listing 37: Startzustand und Ziel des Planungsproblems Π

Das einzige Teilziel, welches erreicht werden kann ist das Atom c. Damit ergibt sich ein Inkonsistenzwert von ungefähr $0,66$.

$$I_{Teilziele}(\Pi) = \frac{2}{3} \approx 0,66 \tag{32}$$

Da das Teilziel c für das Teilziel d benötigt wird, wäre es möglich das Teilziel c als unabhängig zu klassifizieren. Wird das Teilziel c anschließend entfernt, verändert sich jedoch der Inkonsistenzwert zu 1.

$$I_{Teilziele}(\Pi) = \frac{2}{2} = 1 \tag{33}$$

Dadurch, dass beim Maß der Teilziele $I_{Teilziele}$ jedes Teilziel einzeln betrachtet wird, ist die Eigenschaft der Unabhängigkeit eines Atoms im Ziel nicht erfüllt. Somit gilt die Eigenschaft der Unabhängigkeit eines Atoms im Ziel für das Maß der Teilziele $I_{Teilziele}$ nicht. □

Theorem 9. *Sei ein unabhängiges Atom α in einer Vorbedingung einer Operation eines Planungsproblems $\Pi = \langle O, s_I, g\rangle$ gegeben. Dann lässt sich das Planungsproblem zu $\Pi' = \langle O', s_I, g\rangle$ vereinfachen mit $o' = \langle c \setminus \{\alpha\}, e\rangle$ und es gilt:*

$$I_{Teilziele}(\Pi) = \frac{n}{m} = \frac{n'}{m'} = I_{Teilziele}(\Pi')$$

Beweis. Da das Atom α unabhängig in der Vorbedingung ist, ist es immer gültig. Wird das Atom in keinem Effekt verändert und ist von Anfang an im Startzustand gegeben, dann ist das Atom in Vorbedingungen immer erfüllt. Sind Operationen nicht ausführbar, dann kann es also nicht am Atom α liegen. Daher erreicht das Planungsproblem Π' genau die gleichen Zustände wie das Planungsproblem Π und die Anzahl an unerreichten Teilzielen bleibt unverändert. Da das Entfernen von α keine Operationen ausführbar macht, kann die Anzahl an unerreichten Teilzielen nicht wachsen. Weiterhin sind die zuvor ausführbaren Operationen immer noch ausführbar, nachdem α entfernt wurde. Daher verringert sich die Anzahl an unerreichten Teilzielen nicht, also $n' = n$. Dazu ist die Anzahl aller Teilziele in beiden Planungsproblemen gleich: $m' = m$. Somit gilt:

$$I_{Teilziele}(\Pi) = \frac{n}{m} = \frac{n'}{m'} = I_{Teilziele}(\Pi') \tag{34}$$

Damit gilt die Eigenschaft der Unabhängigkeit eines Atoms in einer Vorbedingung einer Operation für das Maß der Teilziele $I_{Teilziele}$. □

5.1.4 Dominanz

Theorem 10. *Seien zwei Atome α und β und drei Planungsprobleme $\Pi = \langle O, s_I, g\rangle$, $\Pi' = \langle O', s_I, g\rangle$ und $\Pi'' = \langle O'', s_I, g\rangle$ mit $o' = \langle c \cup \{\alpha\}, e\rangle \in O'$ und $o'' = \langle c, e \cup \{\beta\}\rangle \in O''$ gegeben. Dazu dominiert das Atom α über das Atom β. Dann gilt:*

$$I_{Teilziele}(\Pi') \geq I_{Teilziele}(\Pi'')$$

Beweis. Sei die Vorbedingung c erfüllbar. Verhindert das Atom α die Ausführbarkeit der Operation o', dann könnte die Anzahl an unerreichten Teilzielen zum Planungsproblem Π anwachsen: $n' \geq n$. Damit würde der Inkonsistenzwert steigen. Ist das Atom α jedoch erfüllbar, dann ändert sich die Anzahl an unerreichten Teilzielen zum Planungsproblem Π nicht: $n' = n$ und der Inkonsistenzwert bleibt gleich. Hingegen kann das Atom β aus dem Effekt der Operation o'' dazu führen, dass Operationen aus O'' ausführbar werden und somit mehr Teilziele im Vergleich zum Planungsproblem Π erreicht werden können. Dazu könnte β auch ein Teilziel sein und so sinkt die Anzahl an unerreichten Teilzielen ebenfalls im Vergleich zum Planungsproblem Π: $n'' \leq n$. Der Inkonsistenztwert kann sich dadurch verringern. Da kein Teilziel zum Zielzustand hinzugefügt wurde, gilt: $m = m' = m''$. Dadurch ergibt sich:

$$\begin{aligned} I_{Teilziele}(\Pi') = \frac{n'}{m'} \geq \frac{n}{m} = I_{Teilziele}(\Pi), \text{ für } n' \geq n \text{ und } m' = m \\ I_{Teilziele}(\Pi) = \frac{n}{m} \geq \frac{n''}{m''} = I_{Teilziele}(\Pi''), \text{ für } n'' \leq n \text{ und } m'' = m \end{aligned} \tag{35}$$

Ist die Vorbedingung c bereits nicht erfüllt, dann wird das Atom β aus dem Planungsproblem Π'' nicht erzeugt. Dadurch können keine zusätzlich Operationen zum Planungsproblem Π angewandt werden und deren Inkonsistenzwert ist gleich, da die Anzahl an unerreichten Teilzielen gleich ist: $n'' = n$. Im Planungsproblem Π' bleibt die Anzahl an unerreichten Teilzielen gleich der des Planungsproblems Π: $n' = n$. Dies kommt daher, dass die Operation mit der unerfüllbaren Vorbedingung c bereits nicht für das Planungsproblem Π angewandt werden kann. Daher sind die Zustände und möglichen Pfade in beiden Planungsproblemen gleich. Es gilt also:

$$\begin{gathered} I_{Teilziele}(\Pi') = \frac{n'}{m'} = I_{Teilziele}(\Pi) = \frac{n}{m} = I_{Teilziele}(\Pi'') = \frac{n''}{m''} \\ \text{für } n = n' = n'' \text{ und } m = m' = m'' \end{gathered} \tag{36}$$

Somit gilt die Eigenschaft der Dominanz für das Maß der Teilziele $I_{Teilziele}$. □

5.2 Maß der fehlenden Atomen

Das Maß der fehlenden Atome $I_{fehlendeAtome}$ für ein Planungsproblem Π ist definiert durch

$$I_{FehlendeAtome} : \Pi \rightarrow \mathbb{R}^{\geq 0}$$
$$I_{FehlendeAtome}(\Pi) = \begin{cases} \frac{\text{minimale Anzahl fehlender Atome}}{\text{Anzahl aller benötigten Atome}} = \frac{u}{v}, & \text{für } 0 \leq u \leq v \\ 1, & \text{sonst} \end{cases}$$

Es erfüllt die Eigenschaften der Normalisierung, der Monotonie im Startzustand, der Monotonie der Vorbedingung einer Operation, die Unabhängigkeit eines Atoms im Startzustand und im Ziel. Weiterhin erfüllt es die Eigenschaft der Unabhängigkeit eines Atoms in einer Vorbedingung einer Operation.

Jedoch erfüllt das Maß die Eigenschaft der Monotonie des Ziels und die Eigenschaft der Dominanz nicht.

5.2.1 Normalisierung

Theorem 11. *Sei ein Planungsproblem $\Pi = \langle O, s_I, g \rangle$ gegeben. Laut der Normalisierung gilt, dass*

$$0 \leq I_{FehlendeAtome}(\Pi) \leq 1$$

Beweis. Zum Einen kann das Planungsproblem Π konsistent sein. Dann fehlen keine Atome um das Planungsproblem zu lösen und somit ist $u = 0$. Das Inkonsistenzmaß $I_{FehlendeAtome}$ berechnet dann eine 0.

$$I_{FehlendeAtome}(\Pi) = \frac{u}{v} = \frac{0}{v} = 0, \text{ für } u = 0 \text{ und } v > 0 \tag{37}$$

Ist kein Atom vorhanden, welches zur Lösung des Problems beiträgt, dann ist die Anzahl der benötigten Atome v gleich der Anzahl der fehlenden Atome u. In diesem Fall berechnet das Maß $I_{FehlendeAtome}$ eine 1. Wenn das Planungsproblem selbst durch die Hinzunahme neuer Atome nicht konsistent werden kann und u sowie v damit unbestimmt sind, dann bildet das Maß $I_{FehlendeAtome}$ auf eine 1 ab.

$$I_{FehlendeAtome}(\Pi) = \frac{u}{v} = \frac{v}{v} = 1, \quad \text{für } u = v \text{ und } v > 0 \tag{38}$$

Außer im Fall der Unbestimmtheit kann die Anzahl der fehlenden Atome u niemals unter 0 fallen, also $u \geq 0$. Dazu kann die minimale Anzahl an fehlenden Atomen nur maximal so groß werden, wie die Anzahl aller benötigten Atome v. Dies kommt daher, da beide Werte auf der gleichen Pseudo-Lösung berechnet werden. Es

gilt also: $0 \leq u \leq v$. Dadurch liegt der Inkonsistenzwert für das Maß $I_{FehlendeAtome}$ in allen weiteren Fällen immer zwischen 0 und 1.

$$0 < I_{FehlendeAtome}(\Pi) = \frac{u}{v} < 1, \text{ für } 0 < u < v \text{ und } v > 0 \tag{39}$$

Die Eigenschaft der Normalisierung ist somit für das Maß der fehlenden Atomen $I_{FehlendeAtome}$ erfüllt. □

5.2.2 Monotonie

Theorem 12. *Sei ein Atom α und ein Planungsproblem $\Pi = \langle O, s_I, g\rangle$ gegeben. Laut der Monotonie-Eigenschaft für den Startzustand gilt:*

$$I_{FehlendeAtome}(\Pi) \geq I_{FehlendeAtome}(\Pi'), \textit{ wobei } \Pi' = \langle O, s_I \cup \{\alpha\}, g\rangle$$

Beweis. Ist das Planungsproblem Π konsistent, dann bildet das Maß $I_{FehlendeAtome}$ auf eine 0 ab. Da bereits alle nötigen Atome im Startzustand sind, führt das Hinzufügen eines neuen Atoms α zu keiner Änderung des Inkonsistenzwertes. Die Anzahl an fehlenden Atomen bleibt gleich: $u' = u = 0$, da das Planungsproblem Π' ebenfalls konsistent ist. Auch die Anzahl an benötigten Atomen ist gleich: $v' = v$.

$$\begin{aligned} I_{FehlendeAtome}(\Pi) = \frac{u}{v} = \frac{0}{v} &= 0, \text{ für } u = 0 \text{ und } v > 0 \\ I_{FehlendeAtome}(\Pi') = \frac{u'}{v'} = \frac{0}{v} &= 0, \text{ für } u' = 0 \text{ und } v' = v \end{aligned} \tag{40}$$

In einem maximal inkonsistenten Planungsproblem Π wird auf eine 1 abgebildet. Wird hier ein Atom α in den Startzustand hinzugefügt, dann kann sich die Inkonsistenz verringern. Es besteht die Möglichkeit, dass das neue Atom α ein fehlendes Atom im Startzustand ist. Dadurch würde sich die Anzahl an fehlenden Atomen verringern: $u' = v - 1$. Ist das Atom α jedoch nicht für die Lösung des Planungsproblem nötig, dann bleibt die Anzahl an fehlenden Atomen gleich: $u' = u = v$.

$$\begin{aligned} I_{FehlendeAtome}(\Pi) = \frac{u}{v} = \frac{v}{v} &= 1, \text{ für } u = v \text{ und } v > 0 \\ I_{FehlendeAtome}(\Pi') = \frac{u'}{v'} = \frac{v}{v} &= 1, \text{ für } u' = v \text{ und } v' = v \\ I_{FehlendeAtome}(\Pi') = \frac{u'}{v'} = \frac{v-1}{v} &< 1, \text{ für } u' = v - 1 \text{ und } v' = v \end{aligned} \tag{41}$$

In den Fällen, in denen das Planungsproblem Π auf einen Wert zwischen 0 und 1 abgebildet wird, gilt $0 < u < v$. Wird in dieses Planungsproblem ein Atom α zum Startzustand hinzugefügt, dann ist es möglich, dass sich die Inkonsistenz verringert. Dies ist dann der Fall, wenn das Atom zur Lösung des Planungsproblems Π benötigt

wird; $u' = u - 1$. Ist das Atom α jedoch nicht an der Lösung des Planungsproblems Π beteiligt, dann bleibt der Inkonsistenzwert gleich: $u' = u$.

$$\begin{gathered} 0 < I_{FehlendeAtome}(\Pi') = \frac{u'}{v'} = \frac{u}{v} = I_{FehlendeAtome}(\Pi) < 1, \\ \text{für } 0 < u < v \text{ und } u' = u \text{ und } v' = v \\ 0 < I_{FehlendeAtome}(\Pi') = \frac{u'}{v'} = \frac{u-1}{v} < \frac{u}{v} = I_{FehlendeAtome}(\Pi) < 1, \\ \text{für } 0 < u < v \text{ und } u' = u - 1 \text{ und } v' = v \end{gathered} \tag{42}$$

Somit verringert sich die Inkonsistenz des Planungsproblems Π' oder bleibt gleich im Vergleich zum Planungsproblem Π, wenn ein Atom α zum Startzustand hinzugefügt wird. Die Eigenschaft der Monotonie im Startzustand ist damit für das Maß der fehlenden Atome $I_{FehlendeAtome}$ erfüllt. □

Theorem 13. *Sei ein Atom α und ein Planungsproblem $\Pi = \langle O, s_I, g \rangle$ gegeben. Ist das Atom α nicht vom Startzustand aus erreichbar und wird dem Ziel hinzugefügt, dann gilt:*

$$\begin{gathered} I_{FehlendeAtome}(\Pi) \geq I_{FehlendeAtome}(\Pi') \\ \textit{wobei } \Pi' = \langle O, s_I, g \cup \{\alpha\} \rangle \textit{ und } \alpha \textit{ ist nicht erreichbar} \end{gathered}$$

Beweis. Sei folgendes Planungsproblem $\Pi = \langle O, s_I, g \rangle$ gegeben. Listing 38 spezifiert zwei Operationen des Planungsproblems Π. Der Startzustand sowie das Ziel des Planungsproblems Π wird dabei in Listing 39 definiert.

```
1 func1:
2    C: a, b, c
3    E: d
4 func2:
5    C: f, h
6    E: g
```

Listing 38: Operationen des Planungsproblems Π

```
1 init: f
2 goal: d
```

Listing 39: Startzustand und Ziel des Planungsproblems Π

Um das Planungsproblem Π zu lösen, müssen die Atome a, b und c schrittweise zum Startzustand hinzugefügt werden. Die Anzahl an fehlenden Atomen ist somit 3. Insgesamt werden 3 Atome benötigt, um das Planungsproblem konsistent zu machen. Das Maß der fehlenden Atome $I_{FehlendeAtome}$ bildet daher auf eine 1 ab und damit tritt der maximal inkonsistente Fall ein.

$$I_{FehlendeAtome}(\Pi) = \frac{u}{v} = \frac{3}{3} = 1 \tag{43}$$

Nun wird ein weiteres Atom dem Ziel hinzugefügt. Das Planungsproblem Π' besitzt dabei die gleichen Operationen wie aus Listing 38. Der Startzustand und das neue Ziel sind dabei aus Listing 40 zu entnehmen.

```
1 init: f
2 goal: d, g
```

Listing 40: Startzustand und Ziel des Planungsproblems Π'

Das neue Teilziel g ist nicht erreichbar. Um es zu erhalten, muss dem Startzustand das Atom h hinzugefügt werden. Da für die zweite Operation das Atom f jedoch schon vorhanden ist, ist die Anzahl an fehlenden Atomen nur 4: $u' = 4$. Insgesamt werden allerdings 5 Atome benötigt um das neue Planungsproblem Π' zu lösen. Die Inkonsistenz verringert sich.

$$I_{FehlendeAtome}(\Pi') = \frac{u'}{v'} = \frac{4}{5} = 0,8 < I_{FehlendeAtome}(\Pi) \tag{44}$$

Daher gilt die Eigenschaft der Monotonie des Ziels nicht für das Maß der fehlenden Atome $I_{FehlendeAtome}$, da es den Inkonsistenzwert verringern kann, auch wenn das neue Teilziel α nicht vom Startzustand aus erreichbar ist. □

Theorem 14. *Sei ein Atom α und ein Planungsproblem $\Pi = \langle O, s_I, g\rangle$ gegeben. Wird das Atom α in eine Vorbedingung einer Operation aufgenommen, dann gilt laut der Monotonie-Eigenschaft einer Vorbedingung folgendes:*

$$\begin{gathered} I_{FehlendeAtome}(\Pi) \geq I_{FehlendeAtome}(\Pi'),\ \textit{wenn } o' \textit{ ausführbar ist} \\ I_{FehlendeAtome}(\Pi) \leq I_{FehlendeAtome}(\Pi'),\ \textit{wenn } o' \textit{ nicht ausführbar ist} \\ \textit{wobei } \Pi' = \langle O', s_I, g\rangle \textit{ mit } o' = \langle c \cup \{\alpha\}, e\rangle \in O' \end{gathered}$$

Beweis. Wird das Atom α in eine Vorbedingung einer Operation o hinzugefügt, die für die Lösung des Planungsproblems nicht relevant ist, dann bleibt die Anzahl an fehlenden Atomen und der benötigten Atome gleich: $u' = u$ und $v' = v$. Das liegt daran, dass sich die Pseudo-Lösung nicht geändert hat. Dies gilt sowohl für den Fall, dass die Operation o' ausführbar ist oder nicht.

$$\begin{gathered} I_{FehlendeAtome}(\Pi) = \frac{u}{v} = \frac{u'}{v'} = I_{FehlendeAtome}(\Pi') \\ \text{für } 0 \leq u' = u \leq v = v' \end{gathered} \tag{45}$$

Ist die Operation weiterhin ausführbar und für die Lösung des Planungsproblems Π relevant, dann erhöht sich die Anzahl an benötigten Atomen um 1: $v' = v + 1$. Hingegen bleibt die Anzahl an fehlenden Atomen gleich: $u' = u$. Dadurch,

dass die Operation immer noch ausführbar ist, muss das Atom, welches hinzugefügt wurde, bereits vorhanden sein. Somit werden beide Planungsprobleme auf eine 0 abgebildet, wenn das Planungsproblem Π konsistent ist. Ansonsten verringert sich der Inkonsistenzwert.

$$\begin{aligned}
I_{FehlendeAtome}(\Pi) = \frac{u}{v} \geq \frac{u'}{v'} = \frac{u}{v+1} = I_{FehlendeAtome}(\Pi') \\
\text{für } 0 \leq u' = u \leq v < v+1 = v' \\
0 = I_{FehlendeAtome}(\Pi) = \frac{u}{v} = \frac{0}{v} = \frac{0}{v+1} = \frac{u'}{v'} = I_{FehlendeAtome}(\Pi') \\
1 = I_{FehlendeAtome}(\Pi) = \frac{u}{v} = \frac{v}{v} > \frac{v}{v+1} = \frac{u'}{v'} = I_{FehlendeAtome}(\Pi') \\
1 > I_{FehlendeAtome}(\Pi) = \frac{u}{v} > \frac{u}{v+1} = \frac{u'}{v'} = I_{FehlendeAtome}(\Pi') > 0
\end{aligned} \tag{46}$$

Scheitert die Ausführung der Operation nach dem Hinzufügen des Atoms α in dessen Vorbedingung und ist für die Lösung des Planungsproblems Π relevant, dann steigt die Inkonsistenz an. Sowohl die Anzahl der fehlenden Atome u' als auch die Anzahl der benötigten Atome v' wächst um eins. Damit ist der Inkonsistenzwert zum Planungsproblem Π gewachsen.

$$\begin{aligned}
I_{FehlendeAtome}(\Pi) = \frac{u}{v} \leq \frac{u+1}{v+1} = \frac{u'}{v'} = I_{FehlendeAtome}(\Pi') \\
\text{für } 0 \leq u' = u+1 \leq v' \text{ und } v' = v+1 \\
0 = I_{FehlendeAtome}(\Pi) = \frac{u}{v} = \frac{0}{v} < \frac{1}{v+1} = \frac{u'}{v'} = I_{FehlendeAtome}(\Pi') < 1 \\
1 = I_{FehlendeAtome}(\Pi) = \frac{u}{v} = \frac{v}{v} = \frac{v+1}{v+1} = \frac{u'}{v'} = I_{FehlendeAtome}(\Pi') \\
0 < I_{FehlendeAtome}(\Pi) = \frac{u}{v} < \frac{u+1}{v+1} = \frac{u'}{v'} = I_{FehlendeAtome}(\Pi') < 1
\end{aligned} \tag{47}$$

Somit gilt die Eigenschaft der Monotonie in einer Vorbedingung einer Operation für das Maß der fehlenden Atome $I_{FehlendeAtome}$. □

5.2.3 Unabhängigkeit eines Atoms

Theorem 15. *Sei ein unabhängiges Atom α im Startzustand s_I eines Planungsproblems $\Pi = \langle O, s_I, g\rangle$ gegeben. Dann lässt sich das Planungsproblem zu $\Pi' = \langle O, s_I \setminus \{\alpha\}, g\rangle$ vereinfachen und es gilt:*

$$I_{FehlendeAtome}(\Pi) = \frac{u}{v} = \frac{u'}{v'} = I_{FehlendeAtome}(\Pi')$$

Beweis. Da das Atom α unabhängig ist, trägt es nicht zur Lösung des Planungsproblems bei. Dadurch ist es nicht in den Variablen der fehlenden Atome oder der benötigten Atome mit ein berechnet. Dadurch existiert es in keiner Vorbedingung einer benötigten Operation, die für die Lösung des Planungsproblems gebraucht wird. Wird anschließend das Atome α aus dem Startzustand entfernt, dann bleibt die Anzahl der fehlenden Atome $u' = u$ und die der benötigten Atome $v' = v$ gleich. Es gilt:

$$I_{FehlendeAtome}(\Pi) = \frac{u}{v} = \frac{u'}{v'} = I_{FehlendeAtome}(\Pi') \tag{48}$$

Somit ist die Eigenschaft für ein unabhängiges Atom α im Startzustand für das Maß der fehlenden Atome $I_{FehlendeAtome}$ erfüllt. □

Theorem 16. *Sei ein unabhängiges Atom α im Ziel g eines Planungsproblems $\Pi = \langle O, s_I, g\rangle$ gegeben. Dann lässt sich das Planungsproblem zu $\Pi' = \langle O, s_I, g \setminus \{\alpha\}\rangle$ vereinfachen und es gilt:*

$$I_{FehlendeAtome}(\Pi) = \frac{u}{v} = \frac{u'}{v'} = I_{FehlendeAtome}(\Pi')$$

Beweis. Ein Teilziel ist unabhängig vom Ziel, wenn es für das Erreichen eines anderen Teilziels benötigt wird. Dies ist zum Beispiel der Fall, wenn das Teilziel in der Vorbedingung einer Operation ist, dessen Effekt ein anderes Teilziel löst. Somit ist das erste Teilziel notwendig um das zweite Teilziel zu erreichen. Auf einem Pfad kann das zweite Teilziel also nicht ohne das erste Teilziel erreicht werden. Wird schließlich das erste Teilziel entfernt, dann erfolgt die Berechnung immer noch auf dem gleichen Pfad. Somit ist die Anzahl der fehlenden und der benötigten Atome sowohl für das Planungsproblem Π als auch Π' gleich.

$$I_{FehlendeAtome}(\Pi) = \frac{u}{v} = \frac{u'}{v'} = I_{FehlendeAtome}(\Pi') \tag{49}$$

Daraus folgt, dass die Unabhängigkeit eines Atoms im Ziel für das Maß der fehlenden Atome $I_{FehlendeAtome}$ gilt. □

Theorem 17. *Sei ein unabhängiges Atom α in einer Vorbedingung einer Operation eines Planungsproblems $\Pi = \langle O, s_I, g\rangle$ gegeben. Dann lässt sich das Planungsproblem zu $\Pi' = \langle O', s_I, g\rangle$ vereinfachen mit $o' = \langle c \setminus \{\alpha\}, e\rangle$ und es gilt:*

$$I_{FehlendeAtome}(\Pi) = \frac{u}{v} = \frac{u'}{v'} = I_{FehlendeAtome}(\Pi')$$

Beweis. Das Atom α ist unabhängig, wenn es im Planungsproblem immer gilt. Dadurch kann die Operation, in der das Atom vorkommt, nur aufgrund der anderen Atome fehlschlagen. Wird dann das Atom α aus der Vorbedingung einer Operation entfernt, dann ist die Operation o' ausführbar, wenn die Operation o es war. Sonst ist die Operation o' nicht ausführbar, falls die Operation o vorher auch nicht

ausführbar ist. Dadurch erreichen beide Planungsprobleme Π und Π' die gleichen Zustände. Damit werden zur Berechnung der Größen der fehlenden und benötigten Atome wieder die gleichen Pseudo-Lösungen genommen, falls es diese gibt. Also ist die Anzahl der fehlenden und der benötigten Atome in beiden Planungsproblemen Π und Π' gleich: $u' = u$ und $v' = v$.

$$I_{FehlendeAtome}(\Pi) = \frac{u}{v} = \frac{u'}{v'} = I_{FehlendeAtome}(\Pi') \tag{50}$$

Somit gilt die Eigenschaft der Unabhängigkeit eines Atoms in einer Vorbedingung einer Operation für das Maß der fehlenden Atome $I_{FehlendeAtome}$. □

5.2.4 Dominanz

Theorem 18. *Seien zwei Atome α und β und drei Planungsprobleme $\Pi = \langle O, s_I, g\rangle$, $\Pi' = \langle O', s_I, g\rangle$ und $\Pi'' = \langle O'', s_I, g\rangle$ mit $o' = \langle c \cup \{\alpha\}, e\rangle \in O'$ und $o'' = \langle c, e \cup \{\beta\}\rangle \in O''$ gegeben. Dazu dominiert das Atom α über das Atom β. Dann gilt:*

$$I_{FehlendeAtome}(\Pi') \geq I_{FehlendeAtome}(\Pi'')$$

Beweis. Sei dazu folgendes Planungsproblem $\Pi = \langle O, s_I, g\rangle$ gegeben, das den Flugverkehr darstellt. Die Operationen aus Listing 41 beschreiben zum Einen die Möglichkeit von einem Flughafen zum Nächsten zu fliegen und das Auftanken eines Flugzeuges an einem Flughafen.

```
1 fly(FROM, TO, PLANE):
2    C: flightRoute(FROM, TO) and at(FROM, PLANE)
3    E: at(TO, PLANE) and not at(FROM, PLANE)
4 refuel(PLACE, PLANE):
5    C: at(PLACE, PLANE), fuelLevel(empty, PLANE)
6    E: fuelLevel(full, PLANE) and not fuelLevel(empty, PLANE)
```

Listing 41: Operationen des Planungsproblems Π

Sei der Startzustand und das Ziel wie in Listing 42 gegeben. Das Ziel ist es mit dem Flugzeug *plane1* den Flughafen Berlin zu erreichen.

```
1 init:
2    at(stn, plane1), at(cdg, plane2), fuelLevel(full, plane1),
3    flightRoute(stn,cdg), flightRoute(ber,muc)
4 goal:
5    at(ber, plane1)
```

Listing 42: Startzustand und Ziel des Planungsproblems Π

Da es jedoch keine Flugroute zwischen dem Flughafen in London und Berlin gibt, ist das Planungsproblem inkonsistent. Damit das Planungsproblem Π konsistent wird, ist die Flugroute zwischen London und Berlin nötig, also *flightRoute(stn,ber)*. Da das zweite Atom aus der Vorbedingung bereits erfüllt ist, fehlt also

nur 1 Atom. Insgesamt werden allerdings 2 Atome benötigt um das Ziel zu erreichen. Daher entspricht der Inkonsistenzwert einer $0,5$.

$$I_{FehlendeAtome}(\Pi) = \frac{x}{y} = \frac{1}{2} = 0,5 \tag{51}$$

Nun wird das Planungsproblem in Π' umgewandelt, in dem ein Atom α einer Vorbedingung hinzugefügt wird. Dazu soll für das Fliegen auch der Füllstand des Treibstoffes auf *full* stehen, siehe Listing 43.

```
1 fly(FROM, TO, PLANE):
2    C: flightRoute(FROM, TO) and at(FROM, PLANE) and fuelLevel(full, PLANE)
3    E: at(TO, PLANE) and not at(FROM, PLANE)
4 refuel(PLACE, PLANE):
5    C: at(PLACE, PLANE), fuelLevel(empty, PLANE)
6    E: fuelLevel(full, PLANE) and not fuelLevel(empty, PLANE)
```

Listing 43: Operationen des Planungsproblems Π'

Da für das Flugzeug *plane1* der Füllstand des Treibstoffes schon auf *full* steht, wird dieses Atom benötigt, jedoch fehlt es nicht. Dadurch verringert sich Inkonsistenz im Gegensatz zum Planungsproblem Π. Die Anzahl an fehlenden Atom x' ist gleich, jedoch wird ein Atom mehr benötigt als im Planungsproblem Π.

$$I_{FehlendeAtome}(\Pi') = \frac{x'}{y'} = \frac{1}{3} \approx 0,3 \tag{52}$$

Wird hingegen ein Atom β in den Effekt der gleiche Operation hinzugefügt, dann muss sich die Inkonsistenz zum Planungsproblem Π nicht ändern. Listing 44 zeigt die veränderte Operation im Planungsproblem Π''. Ist ein Flugzeug gelandet, dann ist der Füllstand des Treibstoffes auf *empty*.

```
1 fly(FROM, TO, PLANE):
2    C: flightRoute(FROM, TO) and at(FROM, PLANE)
3    E: at(TO, PLANE) and not at(FROM, PLANE) and
4       fuelLevel(empty, PLANE)
5 refuel(PLACE, PLANE):
6    C: at(PLACE, PLANE), fuelLevel(empty, PLANE)
7    E: fuelLevel(full, PLANE) and not fuelLevel(empty, PLANE)
```

Listing 44: Operationen des Planungsproblems Π''

Die Berechnung der Inkonsistenz unterscheidet sich dabei nicht zu der des Planungsproblems Π. Daher ist die Anzahl an fehlenden Atomen x'' und die Anzahl an benötigten Atome y'' gleich. Der Inkonsistenzwert ist also $0,5$.

$$I_{FehlendeAtome}(\Pi'') = \frac{x''}{y''} = \frac{1}{2} = 0,5 \tag{53}$$

Es soll gelten, dass die Inkonsistenz für das Planungsproblem Π' größer oder gleich des Planungsproblems Π'' ist. Jedoch ist an diesem Beispiel zu erkennen, dass sich die Inkonsistenz auch verringern kann:

$$I_{FehlendeAtome}(\Pi') = \frac{1}{3} < \frac{1}{2} = I_{FehlendeAtome}(\Pi) = I_{FehlendeAtome}(\Pi'') \tag{54}$$

Somit gilt die Eigenschaft der Dominanz nicht für das Maß der fehlenden Atome $I_{FehlendeAtome}$. □

5.3 Maß der Vorbedingungen

Das Maß zur Berechnung der Inkonsistenz durch das Ignorieren von Vorbedingungen ist definiert durch:

$$I_{Vorbedingung} : \Pi \to \mathbb{R}^{\geq 0} :$$
$$I_{Vorbedingung}(\Pi) = \begin{cases} \dfrac{\text{Anzahl ignorierter Vorbedingungen}}{\text{Anzahl ausgeführter Operationen}} = \dfrac{x}{y} & \text{für } 0 \leq x \leq y \\ 1, & \text{sonst} \end{cases}$$

Es erfüllt die Eigenschaften der Normalisierung, der Eigenschaft der Monotonie im Startzustand und in einer Vorbedingung einer Operation, der Unabhängigkeit eines Atoms im Startzustand, im Ziel und in einer Operationen. Dazu gilt ebenfalls die Eigenschaft der Dominanz.

Hingegen ist die Eigenschaft der Monotonie im Ziel nicht erfüllt.

5.3.1 Normalisierung

Theorem 19. *Sei ein Planungsproblem $\Pi = \langle O, s_I, g \rangle$ gegeben. Laut der Normalisierung muss gelten, dass*

$$0 \leq I_{Vorbedingung}(\Pi) \leq 1$$

Beweis. Ist das Planungsproblem Π konsistent, dann gibt es einen Pfad zwischen dem Startzustand und einem Zielzustand. Dadurch konnten alle Operationen ausgeführt werden, ohne eine Vorbedingung zu ignorieren. Daher ist $x = 0$. In diesem Fall bildet das Maß $I_{Vorbedingung}$ auf eine 0 ab.

$$I_{Vorbedingung}(\Pi) = \frac{x}{y} = \frac{0}{y} = 0, \text{ für } x = 0 \text{ und } y > 0 \tag{55}$$

Wenn keine Operation ausführbar ist, dann ist das Planungsproblem für das Maß $I_{Vorbedingung}$ maximal inkonsistent. Die Anzahl an ignorierten Vorbedingungen wächst damit auf die Anzahl benötigter Operationen an: $x = y$. Das Maß $I_{Vorbedingung}$ bildet dann auf eine 1 ab. Ist es nicht möglich eine Pseudo-Lösung zu finden, dann

sind sowohl x als auch y unbestimmt. Ebenfalls bildet das Maß der Vorbedingung $I_{Vorbedingung}$ dann auf eine 1 ab.

$$I_{Vorbedingung}(\Pi) = \frac{x}{y} = \frac{y}{y} = 1, \text{ für } x = y \text{ und } y > 0 \tag{56}$$

Da es nicht möglich ist eine negative Anzahl an ignorierten Vorbedingungen zu erreichen, ist $x \geq 0$. Weiterhin können nur so viele Vorbedingungen ignoriert werden, wie Operationen ausgeführt werden, da jede Operation nur eine Vorbedingung hat. Dadurch liegt x immer zwischen 0 und y: $0 \geq x \geq y$. Da die Fälle $x = 0$ und $x = y$ bereits gesondert gezeigt werden konnten, gelten für alle weiteren Fällen:

$$0 < I_{Vorbedingung}(\Pi) = \frac{x}{y} < 1, \text{ für } 0 < x < y \tag{57}$$

Somit gilt die Eigenschaft der Normalisierung für das Maß der Vorbedingungen $I_{Vorbedingung}$. □

5.3.2 Monotonie

Theorem 20. *Sei ein Atom α und ein Planungsproblem $\Pi = \langle O, s_I, g\rangle$ gegeben. Laut der Monotonie-Eigenschaft für den Startzustand gilt:*

$$I_{Vorbedingung}(\Pi) \geq I_{Vorbedingung}(\Pi'), \textit{ wobei } \Pi' = \langle O, s_I \cup \{\alpha\}, g\rangle$$

Beweis. Seien zwei Pseudo-Lösungen p_1, p_2 gegeben. Sei der Inkonsistenzwert der Pseudo-Lösung p_1 kleiner der zweiten Pseudo-Lösung p_2. Damit entspricht die Inkonsistenz des Planungsproblems Π dem Inkonsistenzwert der ersten Pseudo-Lösung p_1. Wird nun ein Atom α dem Startzustand hinzugefügt, dann sind alle Pfade des Planungsproblems Π auch im Planungsproblem Π' möglich. Dazu könnten durch das Atom α neue Pfade entstehen. Die zwei Pseudo-Lösungen p_1 und p_2 werden daher bei der Berechnung des Maßes der Vorbedingung wieder gefunden. Dabei bleibt der Inkonsistenzwert gleich, wenn für die Berechnung wieder die Pseudo-Lösung p_1 gewählt wird, da dieser einen kleineren Inkonsistenzwert ergibt als die Pseudo-Lösung p_2. Verändert sich hingegen die Inkonsistenz, dann müssen auf der Pseudo-Lösung p_1 weniger Vorbedingungen ignoriert werden und damit ist $x' < x$ oder eine andere Pseudo-Lösung führt zu einem geringeren Inkonsistenzwert als der der Pseudo-Lösung p_1. Da bei mehreren gefundenen Pseudo-Lösungen, die Inkonsistenz dem geringsten Inkonsistenzwert entspricht, wird wieder der Inkonsistenzwert der Pseudo-Lösung p_1 zurück gegeben oder es wurde eine Pseudo-Lösung mit einem geringeren Inkonsistenzwert gefunden. Somit kann sich die Inkonsistenz nur verringern oder gleich bleiben, wenn ein Atom α dem Startzustand hinzugefügt wird. Für das Maß der Vorbedingung $I_{Vorbedingung}$ gilt daher die Eigenschaft der Monotonie im Startzustand. □

Theorem 21. *Sei ein Atom α und ein Planungsproblem $\Pi = \langle O, s_I, g\rangle$ gegeben. Ist das Atom α nicht vom Startzustand aus erreichbar und wird dem Ziel hinzugefügt, dann gilt:*

$$I_{Vorbedingung}(\Pi) \geq I_{Vorbedingung}(\Pi')$$
wobei $\Pi' = \langle O, s_I, g \cup \{\alpha\} \rangle$ *und* α *ist nicht erreichbar*

Beweis. Sei dazu das Planungsproblem aus Listing 45 gegeben. Dabei geht es um ein Transportsystem, in dem Flugzeuge und LKWs Pakete transportieren können.

```
1  fly(FROM, TO):
2     C: at(PLANE, FROM) and flightLine(FROM, TO)
3     E: at(PLANE, TO) and not at(PLANE, FROM)
4  drive(FROM, TO):
5     C: at(TRUCK, FROM) and truckLine(FROM, TO)
6     E: at(TRUCK, TO) and not at(TRUCK, FROM)
7  charge(PACKAGE, VEHICLE):
8     C: at(PACKAGE, X) and at(VEHICLE, X)
9     E: in(VEHICLE, PACKAGE)
10 unload(PACKAGE, vehicle):
11    C: in(VEHICLE, PACKAGE)
12    E: when at(VEHICLE, X) at(PACKAGE, X)
```

Listing 45: Operationen des Planungsproblems Π

Listing 46 beschreibt hingegen den Startzustand und das Ziel.

```
1  init:
2     in(plane,p1), at(plane,ber), at(p2,lhr), at(truck, lhr)
3     flightLine(cdg, lhr), truckLine(cdg, ber)
4  goal:
5     at(p1,lhr)
```

Listing 46: Startzustand und Ziel des Planungsproblems Π

Das Planungsproblem ist inkonsistent, da das Ziel nicht erreicht werden kann. Das Ziel kann jedoch auf einem Pseudo-Pfad erreicht werden, in dem die Operation *fly(ber,lhr)*, ohne die Vorbedingung zu betrachten, ausgeführt wird. In der Vorbedingung ist das Atom *flightLine(ber,lhr)* nicht erfüllt, da es nicht im Startzustand gegeben ist. Wurde das Paket *p1* zum Flughafen *lhr* geflogen, dann kann es mit der Operation *unload(p1,plane)* entladen werden. Damit ist das Ziel erreicht. Insgesamt wurden zwei Operationen benötigt, also ist $y = 2$. Da nur die erste Operation die Vorbedingung ignorieren muss, ist die Anzahl an ignorierten Vorbedingungen gleich 1. Der Inkonsistenzwert ist damit 0,5.

Nun wird das Atom *at(p2,cdg)*, welches unerreichbar ist, zum Ziel hinzugefügt. Der Pseudo-Pfad, der zu einem Zielzustand führt, ist aus Listing 47 zu entnehmen.

```
1  fly(ber,lhr)
2  unload(p1, plane)
3  charge(p2, plane2)
```

```
4 drive(lhr, cdg)
5 unload(p2, cdg)
```

Listing 47: Pseudo-Pfad um beide Teilziele zu erreichen

Insgesamt wurden 5 Operationen gebraucht, also $y = 5$. Die Anzahl an ignorierten Vorbedingung beläuft sich auf 2, da sowohl die Operation *fly* als auch *drive* nicht ausgeführt werden können. Damit ergibt sich für das zweite Planungsproblem Π' ein Inkonsistenzwert von 0,4.

$$\begin{aligned} I_{Vorbedingung}(\Pi) &= \frac{x}{y} = \frac{1}{2} = 0,5 \\ I_{Vorbedingung}(\Pi') &= \frac{x'}{y'} = \frac{2}{5} = 0,4 \end{aligned} \tag{58}$$

Damit hat sich die Inkonsistenz durch die Hinzunahme eines Atoms in das Ziel verringert. Die Eigenschaft der Monotonie im Ziel gilt daher für das Maß der Vorbedingung $I_{Vorbedingung}$ nicht. □

Theorem 22. *Sei ein Atom α und ein Planungsproblem $\Pi = \langle O, s_I, g\rangle$ gegeben. Wird das Atom α in eine Vorbedingung einer Operation aufgenommen, dann gilt laut der Monotonie-Eigenschaft einer Vorbedingung folgendes:*

$$\begin{aligned} I_{Vorbedingung}(\Pi) &\geq I_{Vorbedingung}(\Pi'),\ \textit{wenn } o' \textit{ ausführbar ist} \\ I_{Vorbedingung}(\Pi) &\leq I_{Vorbedingung}(\Pi'),\ \textit{wenn } o' \textit{ nicht ausführbar ist} \\ &\textit{wobei } \Pi' = \langle O', s_I, g\rangle \textit{ mit } o' = \langle c \cup \{\alpha\}, e\rangle \in O' \end{aligned}$$

Beweis. Seien dazu zwei Pseudo-Lösungen p_1 und p_2 gegeben. Weiterhin sei der Inkonsistenzwert der Pseudo-Lösung p_1 kleiner als der der Pseudo-Lösung p_2. Ist das Atom α immer erfüllt, dann können die gleichen Pseudo-Lösungen p_1 und p_2 gefunden werden, wie im Planungsproblem Π. Da die Pseudo-Lösung p_1 den kleinsten Inkonsistenzwert liefert, ist der Inkonsistenzwert zum Planungsproblem Π gleich geblieben.

Kann nun das Atom α nicht erfüllt werden, dann treten weniger Pfade im Planungsproblem Π' auf als im Planungsproblem Π. Bei der Suche nach Pseudo-Lösungen werden die Pseudo-Lösungen p_1 und p_2 ebenfalls wieder gefunden. Verändere sich der Inkonsistenzwert auf der Pseudo-Lösung p_2 nicht, da das Atom α in ausführbaren Operationen erfüllt ist oder nicht auftaucht. Damit ist der Inkonsistenzwert auf der Pseudo-Lösung p_2 kleiner. Selbst, wenn sich der Inkonsistenzwert für die Pseudo-Lösung p_1 verändert, steigt der Inkonsistenzwert des Planungsproblems Π' an oder bleibt gleich. Das liegt daran, dass auf der Pseudo-Lösung p_2 zu Beginn ein größerer Wert berechnet wurde. Somit ist der kleinste Wert über die Pseudo-Lösung p_2 zu erreichen. Da p_2 jedoch einen gleichen oder größeren Inkonsistenzwert erzielt als die Pseudo-Lösung p_1 des Planungsproblems Π, erhöht sich die Inkonsistenz oder bleibt gleich im Planungsproblem Π'. Damit gilt:

$$\begin{aligned} &I_{Vorbedingung}(\Pi) \geq I_{Vorbedingung}(\Pi'), \text{ wenn } o' \text{ ausführbar ist} \\ &I_{Vorbedingung}(\Pi) \leq I_{Vorbedingung}(\Pi'), \text{ wenn } o' \text{ nicht ausführbar ist} \end{aligned} \tag{59}$$

Somit ist die Eigenschaft der Monotonie in einer Vorbedingung für das Maß der Vorbedingungen $I_{Vorbedingung}$ erfüllt. □

5.3.3 Unabhängigkeit eines Atoms

Theorem 23. *Sei ein unabhängiges Atom α im Startzustand s_I eines Planungsproblems $\Pi = \langle O, s_I, g\rangle$ gegeben. Dann lässt sich das Planungsproblem zu $\Pi' = \langle O, s_I \setminus \{\alpha\}, g\rangle$ vereinfachen und es gilt:*

$$I_{Vorbedingung}(\Pi) = \frac{u}{v} = \frac{u'}{v'} = I_{Vorbedingung}(\Pi')$$

Beweis. Da das Atom α unabhängig des Planungsproblems Π ist, wird es für die Lösung des Planungsproblems Π nicht benötigt. Damit ist es nicht in einer Vorbedingung einer Operation enthalten, nicht Teil des Ziels oder wird durch eine Operation im Pfad erzeugt. Damit ist es im Startzustand nicht notwendig und kann aus dem Startzustand entfernt werden, ohne dass sich der Inkonsistenzwert verändert. Dies liegt daran, dass die Operationen auf dem Pfad zum Zielzustand weiterhin ausführbar sind, auch wenn das Atom α im Startzustand nicht vorhanden ist. Somit werden in den Planungsproblemen Π und Π' die selben Pfade zur Berechnung der Inkonsistenz gewählt. Da diese die gleiche Anzahl an ignorierten Vorbedingungen und an benötigten Operationen aufweisen wie zuvor, gilt folgendes:

$$I_{Vorbedingung}(\Pi) = \frac{x}{y} = \frac{x'}{y'} = I_{Vorbedingung}(\Pi') \tag{60}$$

Somit gilt die Eigenschaft der Unabhängigkeit eines Atoms im Startzustand für das Maß der Vorbedingungen $I_{Vorbedingung}$. □

Theorem 24. *Sei ein unabhängiges Atom α im Ziel g eines Planungsproblems $\Pi = \langle O, s_I, g\rangle$ gegeben. Dann lässt sich das Planungsproblem zu $\Pi' = \langle O, s_I, g \setminus \{\alpha\}\rangle$ vereinfachen und es gilt:*

$$I_{Vorbedingung}(\Pi) = \frac{u}{v} = \frac{u'}{v'} = I_{Vorbedingung}(\Pi')$$

Beweis. Sei das Atom α ein Teilziel, welches für ein weiteres Teilziel benötigt wird. Ohne das Atom α ist es daher nicht möglich das zweite Teilziel zu erreichen. Da bei der Berechnung des Inkonsistenzwertes jeweils der gleiche Pfad verwendet wird, muss das Atom α in einer Vorbedingung einer Operation des Pfades enthalten sein. Ist dies der Fall, dann ist die Anzahl an benötigten Operationen gleich, also $x' = x$, da der Pfad immer noch die gleiche Anzahl an Operationen aufweist. Da sich der

Startzustand nicht geändert hat, ändert sich die Anzahl an ignorierten Vorbedingungen ebenfalls nicht und ist somit $y' = y$. Somit gilt:

$$I_{Vorbedingung}(\Pi) = \frac{x}{y} = \frac{x'}{y'} = I_{Vorbedingung}(\Pi') \tag{61}$$

Es gilt die Eigenschaft der Unabhängigkeit eines Atoms im Ziel für das Maß der Vorbedingungen $I_{Vorbedingung}$. □

Theorem 25. *Sei ein unabhängiges Atom α in einer Vorbedingung einer Operation eines Planungsproblems $\Pi = \langle O, s_I, g\rangle$ gegeben. Dann lässt sich das Planungsproblem zu $\Pi' = \langle O', s_I, g\rangle$ vereinfachen mit $o' = \langle c \setminus \{\alpha\}, e\rangle$ und es gilt:*

$$I_{Vorbedingung}(\Pi) = \frac{u}{v} = \frac{u'}{v'} = I_{Vorbedingung}(\Pi')$$

Beweis. Da das Atom α unabhängig vom Planungsproblem ist, kann es in keiner Operation eines Pseudo-Pfades vorkommen, welches zu einem Zielzustand führt. Damit sind die nicht ausführbaren Operationen weiterhin nicht ausführbar auf diesem Pseudo-Pfad. Die Anzahl an ignorierten Vorbedingungen bleibt damit gleich: $x' = x$. Da der gleichen Pfad verwendet wird, ist die Anzahl an benötigten Operationen ebenfalls gleich, also $y' = y$.

Tritt die Operation dennoch im Pseudo-Pfad, der zur Berechnung der Inkonsistenz verwendet wird, auf, dann ist das Atom nur unabhängig, wenn es im Startzustand enthalten ist und von den Effekten der Operationen nicht beeinflusst wird. Dann ist das Atom immer erfüllt und kann aus der Vorbedingung einer Operation entfernt werden ohne, dass sich die Inkonsistenz ändert. Daher gilt folgendes:

$$I_{Vorbedingung}(\Pi) = \frac{x}{y} = \frac{x'}{y'} = I_{Vorbedingung}(\Pi') \tag{62}$$

Somit gilt die Unabhängigkeit eines Atoms in einer Vorbedingung für das Maß der Vorbedingungen $I_{Vorbedingung}$. □

5.3.4 Dominanz

Theorem 26. *Seien zwei Atome α und β und drei Planungsprobleme $\Pi = \langle O, s_I, g\rangle$, $\Pi' = \langle O', s_I, g\rangle$ und $\Pi'' = \langle O'', s_I, g\rangle$ mit $o' = \langle c \cup \{\alpha\}, e\rangle \in O'$ und $o'' = \langle c, e \cup \{\beta\}\rangle \in O''$ gegeben. Dazu dominiert das Atom α über das Atom β. Dann gilt:*

$$I_{Vorbedingung}(\Pi') \geq I_{Vorbedingung}(\Pi'')$$

Beweis. Ist die Vorbedingung c nicht erfüllt, dann kann sowohl die Operation o' als auch o'' nicht ausgeführt werden. Hingegen kann die Operation o'' und o ausgeführt werden, wenn die Vorbedingung c erfüllbar ist. Anders sieht es bei der Operation o' aus, die durch das Atom α nicht mehr ausführbar sein kann. Ist dies der Fall, dann können weniger Pfade im Planungsproblem Π' erreicht werden, wie im Vergleich

zum Planungsproblem Π. Das Maß der Vorbedingung berechnet die Inkonsistenz dann auf Pseudo-Lösungen, die eine höhere Inkonsistenz ergeben. Diese Pseudo-Lösungen sind auch im Planungsproblem Π möglich, jedoch wird der niedrigste Inkonsistenzwert gewählt. Damit kann sich der Inkonsistenzwert im Planungsproblem Π' nur vergrößern.

Hingegen können durch die Operation o'' neue Pfade im Vergleich zum Planungsproblem Π gebildet werden. Dadurch kann eine Pseudo-Lösung gefunden werden, die einen geringeren Inkonsistenzwert ergibt wie der vom Planungsproblem Π. Hier kann sich die Inkonsistenz daher verringern. Es gilt:

$$I_{Vorbedingung}(\Pi') \geq I_{Vorbedingung}(\Pi) \geq I_{Vorbedingung}(\Pi'') \tag{63}$$

Somit ist die Eigenschaft der Dominanz für das Maß der Vorbedingungen $I_{Vorbedingung}$ erfüllt. □

5.4 Maß der Kritische Atome

Das Maß der kritischen Atome $I_{KritischeAtome}$für ein Planungsproblem Π ist definiert durch

$$I_{KritischeAtome} : \Pi \to \mathbb{R}^{\geq 0}$$
$$I_{KritischeAtome}(\Pi) = \frac{\sum^{o \in O} f(o)}{\text{Anzahl ausgeführter Operationen}} = \frac{s}{y},$$
$$\text{für } 0 \leq s \leq y$$

Es erfüllt die Eigenschaften der Normalisierung, der Monotonie im Startzustand, der Monotonie in einer Vorbedingung, der Unabhängigkeit eines Atoms im Startzustand und des Ziels, der Unabhängigkeit eines Atoms in einer Vorbedingung und der Dominanz.

Das Maß erfüllt die Eigenschaft der Monotonie im Ziel nicht.

5.4.1 Normalisierung

Theorem 27. *Sei ein Planungsproblem $\Pi = \langle O, s_I, g \rangle$ gegeben. Laut der Normalisierung gilt:*

$$0 \leq I_{KritischeAtome}(\Pi) \leq 1$$

Beweis. Ist das Planungsproblem Π konsistent, dann gibt es einen Pfad zwischen dem Startzustand und einem Zielzustand. Da alle Atome der Vorbedingungen der Operationen erfüllt sind, haben alle Operationen eine Wertigkeit von 0. Dadurch ist die Summe der Wertigkeiten der ausgeführten Operationen ebenfalls $s = 0$ und das Maß bildet auf eine 0 ab.

$$I_{KritischeAtome}(\Pi) = \frac{s}{y} = \frac{0}{y} = 0, \text{ für } s = 0 \text{ und } y > 0 \tag{64}$$

Haben hingegen alle ausgeführten Operationen eine Wertigkeit von 1, dann ist deren Summe gleich der Anzahl an ausgeführten Operationen: $s = y$. In diesem Fall ist das Planungsproblem Π für das Maß der kritischen Atome $I_{KritischeAtome}$ maximal inkonsistent und bildet auf eine 1 ab.

$$I_{KritischeAtome}(\Pi) = \frac{s}{y} = \frac{y}{y} = 1, \text{ für } s = y \text{ und } y > 0 \tag{65}$$

In allen weiteren Fälle liegt s immer zwischen einem Wert von $0 < s < y$, da die Wertigkeit einer Operation $f(o)$ immer zwischen $0 \leq f(o) \leq 1$ liegt. Dadurch kann die Summe aus der Wertigkeit aller ausgeführten Operationen niemals negativ werden und auch die Anzahl an ausgeführten Operationen y nicht übersteigen. Somit gilt:

$$0 < I_{KritischeAtome}(\Pi) = \frac{s}{y} < 1, \text{ für } 0 < s < y \text{ und } y > 0 \tag{66}$$

Damit gilt die Eigenschaft der Normalisierung für das Maß der kritischen Atome $I_{KritischeAtome}$. □

5.4.2 Monotonie

Theorem 28. *Sei ein Atom α und ein Planungsproblem $\Pi = \langle O, s_I, g\rangle$ gegeben. Laut der Monotonie-Eigenschaft für den Startzustand gilt:*

$$I_{KritischeAtome}(\Pi) \geq I_{KritischeAtome}(\Pi'), \textit{ wobei } \Pi' = \langle O, s_I \cup \{\alpha\}, g\rangle$$

Beweis. Ist das Planungsproblem Π konsistent, dann ist auch Π' konsistent und der Inkonsistenzwert ist gleich. Sei nun das Planungsproblem Π inkonsistent und der Inkonsistenzwert sei über die Pseudo-Lösung p bestimmt. Das Planungsproblem Π' hat mindestens die gleichen Pseudo-Pfade wie Π. Besitzt eine Operation das Atom α in der Vorbedingung, dann erfolgt die Berechnung der Wertigkeit dieser Operation ohne die Wertigkeit des Atoms α, sofern das Atom noch im betrachteten Zustand vorhanden ist. Wird zur Berechnung des Inkonsistenzwertes von Π' nun die gleiche Pseudo-Lösung p gewählt und α war ein Atom, welches mindestens eine Vorbedingung zum fehlschlagen gebracht hat, dann verringert sich die Inkonsistenz im Vergleich zum Planungsproblem Π. Hat das Atom α hingegen keinen Einfluss auf die Pseudo-Lösung p, dann ist der Inkonsistenzwert von Π' gleich dem Inkonsistenzwert des Planungsproblems Π.

Erfolgt die Berechnung des Inkonsistenzwertes auf einer anderen Pseudo-Lösung wie auf p, dann ist dieser Wert ebenfalls kleiner als der Inkonsistenzwert des Planungsproblems Π. Sei dazu die Berechnung auf der Pseudo-Lösung h ausgeführt. Erfolgt die Berechnung nun auf der Pseudo-Lösung h, dann muss dieser Wert noch

kleiner sein als der der Pseudo-Lösung p. Dies liegt daran, dass das Maß der kritischen Atome dem kleinstmöglichen Inkonsistenzwert entspricht. Damit kann der berechnete Wert der Pseudo-Lösung h nur kleiner sein als der Wert der Pseudo-Lösung p. Dazu ist der Wert der Pseudo-Lösung p aus Π' kleiner gleich dem Wert der Pseudo-Lösung p des Planungsproblems Π. Daher gilt:

$$I_{KritischeAtome}(\Pi) \geq I_{KritischeAtome}(\Pi') \tag{67}$$

Somit ist die Eigenschaft der Monotonie im Startzustand für das Maß der kritischen Atome $I_{KritischeAtome}$ erfüllt. □

Theorem 29. *Sei ein Atom α und ein Planungsproblem $\Pi = \langle O, s_I, g\rangle$ gegeben. Ist das Atom α nicht vom Startzustand aus erreichbar und wird dem Ziel hinzugefügt, dann gilt:*

$$I_{KritischeAtome}(\Pi) \geq I_{KritischeAtome}(\Pi')$$
$$\textit{wobei } \Pi' = \langle O, s_I, g \cup \{\alpha\}\rangle \textit{ und } \alpha \textit{ ist nicht erreichbar}$$

Beweis. Sei dazu das Planungsproblem aus Listing 45 des Kapitels 5.3.2 gegeben. Um Pakete zwischen zwei Standorten zu transportieren gibt es Flugzeuge und LKWs, die jedoch Flug- und Fahrtrouten unterliegen. Listing 48 beschreibt den Startzustand und das Ziel.

```
1 init:
2    in(plane,p1), at(plane,ber), at(p2,lhr), at(truck, lhr)
3    flightLine(cdg, lhr), truckLine(cdg, ber)
4 goal:
5    at(p1,lhr)
```

Listing 48: Startzustand und Ziel des Planungsproblems Π

Das Planungsproblem ist inkonsistent, da das Ziel nicht erreicht werden kann. Das Ziel kann jedoch auf einem Pseudo-Pfad erreicht werden, in dem die Operation *fly(ber,lhr)*, ohne die Vorbedingung zu betrachten, ausgeführt wird. In der Vorbedingung ist das Atom *flightLine(ber,lhr)* nicht erfüllt, da es nicht im Startzustand gegeben ist. Wurde das Paket *p1* zum Flughafen *lhr* geflogen, dann kann es mit der Operation *unload(p1,plane)* entladen werden. Damit wäre das Ziel erreicht. Insgesamt wurden zwei Operationen gebraucht, also ist $y = 2$. In der ersten Vorbedingung musste das Atom *flightLine(ber,lhr)* ignoriert werden. Dieses Atom hat eine Wertigkeit von 1. Da das andere Atom der Vorbedingung jedoch erfüllt ist, hat die Operation insgesamt eine Wertigkeit von 0,5. Die zweite Operation der Pseudo-Lösung hat eine Wertigkeit von 0. Insgesamt ergibt sich daraus ein Inkonsistenzwert von 0,25.

Nun wird das Atom *at(p2,cdg)*, welches unerreichbar ist, zum Ziel hinzugefügt. Der Pseudo-Pfad, der zu einem Zielzustand führt, ist aus Listing 49 zu entnehmen.

```
1 fly(ber,lhr)
2 unload(p1, plane)
```

```
3 charge(p2, plane2)
4 drive(lhr, cdg)
5 unload(p2, cdg)
```

Listing 49: Pseudo-Pfad um beide Teilziele zu erreichen

Insgesamt wurden 5 Operationen gebraucht, also $y' = 5$. Die Wertigkeiten der Operationen *fly* und *drive* sind $0,5$. Alle anderen Operationen haben eine Wertigkeit von 0. Daraus ergibt sich ein Inkonsistenzwert von $0,2$.

$$\begin{aligned} I_{KritischeAtome}(\Pi) &= \frac{0,5+0}{2} = 0,25 \\ I_{KritischeAtome}(\Pi') &= \frac{0,5*2+0*3}{5} = 0,2 \end{aligned} \tag{68}$$

Damit hat sich die Inkonsistenz durch die Hinzunahme eines Atoms in das Ziel verringert. Die Eigenschaft der Monotonie im Ziel gilt daher für das Maß der kritischen Atome $I_{KritischeAtome}$ nicht. □

Theorem 30. *Sei ein Atom α und ein Planungsproblem $\Pi = \langle O, s_I, g\rangle$ gegeben. Wird das Atom α in eine Vorbedingung einer Operation aufgenommen, dann gilt laut der Monotonie-Eigenschaft einer Vorbedingung folgendes:*

$$\begin{aligned} I_{KritischeAtome}(\Pi) &\geq I_{KritischeAtome}(\Pi'),\ \textit{wenn } o' \textit{ ausführbar ist} \\ I_{KritischeAtome}(\Pi) &\leq I_{KritischeAtome}(\Pi'),\ \textit{wenn } o' \textit{ nicht ausführbar ist} \\ \textit{wobei } \Pi' &= \langle O', s_I, g\rangle \textit{ mit } o' = \langle c \cup \{\alpha\}, e\rangle \in O' \end{aligned}$$

Beweis. Seien dazu zwei Pseudo-Lösungen p_1 und p_2 gegeben. Weiterhin sei der Inkonsistenzwert der Pseudo-Lösung p_1 kleiner als der der Pseudo-Lösung p_2. Ist das Atom α immer erfüllt, dann können die gleichen Pseudo-Lösungen p_1 und p_2 gefunden werden, wie im Planungsproblem Π. Da die Pseudo-Lösung p_1 den kleinsten Inkonsistenzwert liefert, ist der Inkonsistenzwert zum Planungsproblem Π gleich geblieben.

Kann nun das Atom α nicht erfüllt werden, dann treten weniger Pfade im Planungsproblem Π' auf als im Planungsproblem Π. Bei der Suche nach Pseudo-Lösungen werden die Pseudo-Lösungen p_1 und p_2 ebenfalls wieder auftreten. Verändere sich der Inkonsistenzwert auf der Pseudo-Lösung p_2 nicht, da das Atom α in ausführbaren Operationen erfüllt ist oder nicht auftaucht. Sei damit der Inkonsistenzwert auf der Pseudo-Lösung p_2 kleiner als im Planungsproblem Π. Selbst, wenn sich der Inkonsistenzwert für die Pseudo-Lösung p_1 verändert, steigt der Inkonsistenzwert des Planungsproblems Π' an oder bleibt gleich. Das liegt daran, dass auf der Pseudo-Lösung p_2 zu Beginn ein größerer Wert berechnet wurde. Somit ist der kleinste Wert über die Pseudo-Lösung p_2 zu erreichen. Da p_2 jedoch einen gleichen oder größeren Inkonsistenzwert erzielt als die Pseudo-Lösung p_1 des Planungsproblems Π, erhöht sich die Inkonsistenz oder bleibt gleich im Planungsproblem Π'. Damit gilt:

$$\begin{aligned} &I_{KritischeAtome}(\Pi) \geq I_{KritischeAtome}(\Pi'), \text{ wenn } o' \text{ ausführbar ist} \\ &I_{KritischeAtome}(\Pi) \leq I_{KritischeAtome}(\Pi'), \text{ wenn } o' \text{ nicht ausführbar ist} \end{aligned} \tag{69}$$

Somit ist die Eigenschaft der Monotonie in einer Vorbedingung für das Maß der kritischen Atome $I_{KritischeAtome}$ erfüllt. □

5.4.3 Unabhängigkeit eines Atoms

Theorem 31. *Sei ein unabhängiges Atom α im Startzustand s_I eines Planungsproblems $\Pi = \langle O, s_I, g\rangle$ gegeben. Dann lässt sich das Planungsproblem zu $\Pi' = \langle O, s_I \setminus \{\alpha\}, g\rangle$ vereinfachen und es gilt:*

$$I_{KritischeAtome}(\Pi) = \frac{u}{v} = \frac{u'}{v'} = I_{KritischeAtome}(\Pi')$$

Beweis. Da das Atom α unabhängig ist, wird es auch nicht zur Berechnung des Inkonsistenzwertes benötigt. Es ist damit entweder nicht auf der Pseudo-Lösung enthalten oder wird auf diesem in einem Effekt erzeugt. Die Berechnung der Inkonsistenz wird dazu weiterhin auf der gleichen Pseudo-Lösung ausgeführt. Damit bleibt die Anzahl an benötigten Operationen gleich: $y' = y$. Kommt das Atom α in keiner Vorbedingung des Pfades vor, dann ist die Summe der Wertigkeiten der Operationen ebenfalls gleich: $s' = s$. Das gleiche ist der Fall, wenn das Atom α durch ein Effekt erzeugt wird. Dann ist das Atom α verfügbar und ändert an der Summer der Wertigkeiten der Operationen nichts, es gilt also $s' = s$:

$$I_{KritischeAtome}(\Pi) = \frac{s}{y} = \frac{s'}{y'} = I_{KritischeAtome}(\Pi') \tag{70}$$

Somit gilt die Eigenschaft der Unabhängigkeit eines Atoms im Startzustand für das Maß der kritischen Atome $I_{KritischeAtome}$. □

Theorem 32. *Sei ein unabhängiges Atom α im Ziel g eines Planungsproblems $\Pi = \langle O, s_I, g\rangle$ gegeben. Dann lässt sich das Planungsproblem zu $\Pi' = \langle O, s_I, g \setminus \{\alpha\}\rangle$ vereinfachen und es gilt:*

$$I_{KritischeAtome}(\Pi) = \frac{u}{v} = \frac{u'}{v'} = I_{KritischeAtome}(\Pi')$$

Beweis. Ein Atom kann im Ziel nur unabhängig sein, wenn es für die Ausführung von Operationen für ein anderes Teilziel benötigt wird. Sei das Teilziel β also nur erreichbar, wenn vorher das Teilziel α erreicht wurde. Wird nun α aus dem Ziel entfernt, dann erfolgt die Berechnung der Inkonsistenz immer noch auf der gleichen Pseudo-Lösung p wie im Planungsproblem Π. Dadurch, dass der gleiche Pfad p verwendet wird, ist die Anzahl an benötigten Operationen gleich: $y' = y$. Da sich sonst

nichts am Planungsproblem geändert hat, ändern sich die Wertigkeiten der Operationen ebenfalls nicht. Dadurch ist die Summe der Wertigkeiten der Operationen ebenfalls gleich mit $s' = s$. Es gilt also:

$$I_{KritischeAtome}(\Pi) = \frac{s}{y} = \frac{s'}{y'} = I_{KritischeAtome}(\Pi') \tag{71}$$

Somit gilt die Eigenschaft der Unabhängigkeit eines Atoms im Ziel für das Maß der kritischen Atome $I_{KritischeAtome}$. □

Theorem 33. *Sei ein unabhängiges Atom α in einer Vorbedingung einer Operation eines Planungsproblems $\Pi = \langle O, s_I, g\rangle$ gegeben. Dann lässt sich das Planungsproblem zu $\Pi' = \langle O', s_I, g\rangle$ vereinfachen mit $o' = \langle c \setminus \{\alpha\}, e\rangle$ und es gilt:*

$$I_{KritischeAtome}(\Pi) = \frac{u}{v} = \frac{u'}{v'} = I_{KritischeAtome}(\Pi')$$

Beweis. Da das Atom α unabhängig vom Planungsproblem ist, ist es im Pfad zur Berechnung der Inkonsistenz nicht gegeben, wird in einem Effekt erzeugt oder ist bereits im Startzustand enthalten. Da wieder die gleiche Pseudo-Lösung zur Berechnung des Inkonsistenzwertes genommen wird, gibt es keine Operation, in der das Atom α fehlschlägt. Das liegt daran, dass es ansonsten nicht unabhängig vom Planungsproblem ist. Da die gleiche Pseudo-Lösung zur Berechnung verwendet wird, ist die Anzahl an benötigten Operationen gleich, also: $y' = y$. Ebenso muss das Atom α auf der Pseudo-Lösung immer ausführbar sein. Daher ändert es an der Summer der Wertigkeiten der Operationen nichts: $s' = s$. Es gilt:

$$I_{KritischeAtome}(\Pi) = \frac{s}{y} = \frac{s'}{y'} = I_{KritischeAtome}(\Pi') \tag{72}$$

Somit gilt die Unabhängigkeit eines Atoms in einer Vorbedingung für das Maß der kritischen Atome $I_{KritischeAtome}$. □

5.4.4 Dominanz

Theorem 34. *Seien zwei Atome α und β und drei Planungsprobleme $\Pi = \langle O, s_I, g\rangle$, $\Pi' = \langle O', s_I, g\rangle$ und $\Pi'' = \langle O'', s_I, g\rangle$ mit $o' = \langle c \cup \{\alpha\}, e\rangle \in O'$ und $o'' = \langle c, e \cup \{\beta\}\rangle \in O''$ gegeben. Dazu dominiert das Atom α über das Atom β. Dann gilt:*

$$I_{KritischeAtome}(\Pi') \geq I_{KritischeAtome}(\Pi'')$$

Beweis. Wenn alle drei Planungsprobleme konsistent sind, dann liegt deren Inkonsistenzwert bei 0 und ist für alle gleich. Die Gleichung gilt daher, wenn alle konsistent sind. Sei nun das Planungsproblem Π inkonsistent und dessen Inkonsistenzwert über die Pseudo-Lösung p berechnet. Dazu sei die Wertigkeit der Operation o mit $f(o) = r$ gegeben. Die Wertigkeit der Operation o'' ist gleich der Operation o, da die Wertigkeit durch die Atome in den Vorbedingungen bestimmt wird. Da die

Vorbedingungen bei beiden Operationen gleich ist, gilt $f(o) = f(o'') = r$. Für die Operation o' gilt jedoch, dass sich die Wertigkeit zur Operation o vergrößern kann, wenn das Atom α nicht erfüllt ist. Die Wertigkeit der Operation o' ist dann die Wertigkeit der Operation o plus der Wertigkeit des Atoms α: $f(o) < f(o') = f(o) + \omega(\alpha)$. Ist das Atom α hingegen erfüllt, dann ist die Wertigkeit der Operation o' gleich der Wertigkeit der Operation o. Erfolgt die Berechnung des Inkonsistenzwertes auf der gleichen Pseudo-Lösung p, dann kann die Inkonsistenz für das Planungsproblem Π' im Gegensatz zu Π und Π'' ansteigen. Dies ist der Fall, wenn α nicht erfüllt ist und somit die Wertigkeit der Operation steigt. Das Planungsproblem Π'' kann sogar einen geringeren Inkonsistenzwert erzielen, da sich die Wertigkeit einer Operation verringern kann, falls das Atom β in dessen Vorbedingung vorkommt. Weiterhin tritt es nur im Effekt auf und dadurch erhöht es bei der Berechnung der Wertigkeit des Atoms nur den Nenner. Das führt zu einer kleineren Wertigkeit als zuvor.

Werden in den Planungsproblemen Π' und Π'' andere Pseudo-Lösungen zur Berechnung des Inkonsistenzwertes genommen, dann ist der Inkonsistenzwert von Π'' zum Planungsproblem Π kleiner und der Inkonsistenzwert von Π' zu Π größer. Im Planungsproblem Π'' sind die gleichen Pfade wie in Π möglich und es können durch das Atom β sogar weitere entstanden sein. Dadurch muss die Pseudo-Lösung p, die für die Berechnung des Inkonsistenzwertes von Π verwendet wird, auch im Planungsproblem Π'' enthalten sein. Verändert sich für das Planungsproblem Π'' jedoch der Inkonsistenzwert, dann muss es dort eine Pseudo-Lösung geben, die einen geringeren Wert als die Pseudo-Lösung p hat. Ansonsten würde der Wert über die Pseudo-Lösung p berechnet werden. Damit kann sich der Inkonsistenzwert für das Planungsproblem Π'' im Vergleich zu Π nur verringern.

Im Gegensatz zum Planungsproblem Π' können eventuell nicht alle Pfade wie im Planungsproblem Π erreicht werden. Bei der Suche nach Pseudo-Lösungen, wird auch die Pseudo-Lösung p gefunden. Dadurch, dass das Atom α den Inkonsistenzwert anheben kann, gibt es die Möglichkeit, dass eine Pseudo-Lösung h gefunden wird, die einen geringeren Inkonsistenzwert für das Planungsproblem Π' zurück gibt, als die Pseudo-Lösung p. Dennoch ist der Inkonsistenzwert für die Pseudo-Lösung h immer noch höher als der Inkonsistenzwert der Pseudo-Lösung p des Planungsproblems Π. Die Pseudo-Lösung h im Planungsproblem Π muss einen größeren Inkonsistenzwert ergeben als die Pseudo-Lösung p, da das Maß der kritischen Atome immer den geringsten Inkonsistenzwert der Pseudo-Lösungen zurück gibt. Dadurch ist der geringste Wert für das Planungsproblem Π' der Inkonsistenzwert der Pseudo-Lösung p des Planungsproblems Π. Damit gilt folgendes:

$$I_{KritischeAtome}(\Pi') \geq I_{KritischeAtome}(\Pi) \geq I_{KritischeAtome}(\Pi'') \tag{73}$$

Somit gilt die Eigenschaft der Dominanz für das Maß der kritischen Atome $I_{KritischeAtome}$. □

6 Programmierung

Die Inkonsistenzmaße sind in Java mit der Version 8 programmiert. Dazu wurde die Bibliothek pddl4j [13] [15] verwendet, die Java um die Sprache PDDL erweitert. Die Suche nach einem Zielzustand erfolgt durch das Ausführen aller Operationen auf die Zustände, falls deren Vorbedingung erfüllt ist. Kann kein Zielzustand gefunden werden, dann wurden bei der Suche alle möglichen Zustände des Planungsproblems erzeugt. Treten keine neuen Zustände mehr auf, dann kann die Suche abgebrochen werden und das Planungsproblem ist inkonsistent.

Um die Bibliothek verwenden zu können, ist die Datei *pddl4j-VERSION.jar* nötig. Für diese Arbeit wurde die Version 3.7.2 verwendet und ist unter [14] zu finden. Abschließend muss diese Datei als externe Bibliothek dem Projektordner hinzugefügt werden.

Bevor im Planungsproblem gesucht werden kann, wird zunächst die Datei der Domäne und der Problemspezifikation mit einer Konsoleneingabe übergeben. Unter der Programmierumgebung *eclipse* lässt sich die Konsoleneingabe bei der Ausführung des Projektes simulieren. Dafür müssen die zwei Dateien im root-Verzeichnis des Projektes liegen. Die Konsoleneingabe unter *eclipse* ist dem Listing 50 zu entnehmen.

```
1 -o domaene.pddl -f problem.pddl
```

Listing 50: Argumente der Konsoleneingabe in *eclipse*

Die Domäne wird durch das Argument *-o* dargestellt und die Problemspezifikation durch das Argument *-f*. Anschließend werden die Dateien geladen und auf die Korrektheit der Syntax überprüft. Listing 51 zeigt diesen Vorgang. Zeile 3 und 4 laden jeweils die Domäne und das Problem, welche in Zeile 7 durch die ProblemFactory auf die Syntax überprüft werden. Tritt bei der Überprüfung der Syntax ein Fehler auf, dann wird dies in Zeile 13 ausgegeben und das Programm terminiert. Ansonsten wird in Zeile 17 das Problem als CodedProblem kodiert.

```
1  final ProblemFactory factory = ProblemFactory.getInstance();
2
3  File domain = (File) arguments.get(Planner.DOMAIN);
4  File problem = (File) arguments.get(Planner.PROBLEM);
5  ErrorManager errorManager = null;
6  try {
7     errorManager = factory.parse(domain, problem);
8  } catch (IOException e) {
9     System.exit(0);
10 }
11
12 if (!errorManager.isEmpty()) {
13    errorManager.printAll();
14    System.exit(0);
```

```
15 }
16
17 final CodedProblem pb = factory.encode();
18 BitState init = new BitState(pb.getInit());
19 BitVector goal= new BitState(pb.getGoal());
20 List<BitOp> operations = pb.getOperators();
```

Listing 51: Kodieren der Domäne und der Problemspezifikation in pddl4j

Innerhalb des CodedProblems lässt sich der Startzustand wie in Zeile 18, das Ziel wie in Zeile 19 und die Operationen wie in Zeile 20 extrahieren. Anschließend kann die Suche wie in Kapitel 6.1.1 nach einem Zielzustand beginnen. Wird kein Zielzustand erreicht, dann beginnt die Berechnung der vier vorgestellten Maße aus Kapitel 4.2. Die genaue Berechnung der Maße ist unter den Kapiteln 6.2 bis 6.5 zu finden.

Für die Suche im Planungsproblem wird zusätzlich eine Klasse *Node* implementiert, die für jeden Zustand weitere Informationen bereitstellt. Zum Einen werden Informationen zum vorherigen Zustand *parent* bereit gestellt. Somit kann ein Pfad von jedem Zustand zum Startzustand zurückverfolgt werden. Um ebenfalls die Operationen des Pfades leicht auslesen zu können, wird die Operation abgespeichert, mit der der vorherige Zustand zum aktuellen Zustand transformiert wurde. Für die Suche werden ebenfalls Kosten *costs* abgespeichert. Diese beschreiben den Kostenaufwand um vom Startzustand zu diesem Zustand zu gelangen. Entspricht das Ausführen einer Operation den Kosten von 1, dann sagen die Kosten des Zustandes aus, wie viele Operationen bereits ausgeführt wurden, um diesen Zustand zu erhalten.

Weiterhin wird die Anzahl an erreichten Teilzielen *numberReachedPG* für jeden Zustand bestimmt, um die Berechnung der günstigen Zustände zu erleichtern. Dafür wird die Vereinigung von jedem Zustand mit dem Ziel gebildet und anschließend die Kardinalität des vereinigten Zustandes in der Variable *numberReachedPG* gespeichert.

Für die Berechnung von Pseudo-Lösungen wird ebenfalls ein Zeitstempel *timeStemp* benötigt, der für jeden weiteren Schritt in der Suche erhöht wird. Somit können mehrere Pseudo-Lösungen in einem gleichen Schritt gefunden werden. Ein Zeitschritt beschreibt dabei das Ausführen aller möglichen Operationen auf jeden Zustand des selben Zeitschrittes. Sei dazu ein Zustand zum Zeitschritt 0 gegeben. Alle Zustände, die aus diesem Zustand durch Operationen erzeugt werden können, bekommen damit den Zeitschritt 1. Der Zeitschritt 2 gilt dann für alle möglichen erzeugbaren Zustände aus dem Zeitschritt 1.

6.1 Hilfsmethoden

6.1.1 search

Die Methode *search* bekommt ein Planungsproblem *problem* und eine PriorityQueue *open*. Dabei sucht die Methode einen Zielzustand, falls es so einen Zustand gibt.

```
1  public Node search(CodedProblem problem, PriorityQueue<Node> open){
2     ...
3     List<Node> closed = new ArrayList<Node>();
4
5     while (!open.isEmpty()) {
6        final Node current = open.poll();
7        closed.add(current);
8
9        if (current.satisfy(problem.getGoal())) {
10          return current;
11       } else {
12          for (int i = 0; i < problem.getOperators().size(); i++) {
13             BitOp a = problem.getOperators().get(i);
14             if (a.isApplicable(current)) {
15                Node next = new Node(current);
16                final List<CondBitExp> effects = a.getCondEffects();
17                for (CondBitExp ce : effects) {
18                   if (current.satisfy(ce.getCondition()))
19                      next.apply(ce.getEffects());
20                }
21                if (!closed.contains(next)) {
22                   next.setParent(current);
23                   next.setOperator(i);
24                   next.setTimeStemp(current.getTimeStemp()+1);
25                   next.setCosts(current.getCost()+1);
26                   open.add(next);
27                }
28             }
29          }
30       }
31    }
32 return null;
33 }
```

Listing 52: Methode *search*

Listing 52 zeigt die Suche. Zunächst wird eine Liste *closed* von Zuständen erstellt, die später alle abgearbeiteten Zustände enthält. Anschließend beginnt die Suche ab Zeile 5. Dafür wird wie in Zeile 6 und 7 der vorderste Zustand aus der

PriorityQueue *open* entfernt und in die Liste *closed* eingefügt. Anschließend wird der Zustand *current* in den Zeilen 9 bis 24 abgearbeitet. Kann der Zustand *current* das Ziel erreichen, dann wird dieser Zustand zurück gegeben und das Planungsproblem ist konsistent, siehe Zeile 9 und 10. Ansonsten wird für jede Operation des Planungsproblems getestet, ob diese auf den aktuell betrachtenden Zustand *current* anwendbar ist, siehe Zeile 14. Ist eine Operation anwendbar, dann wird in den Zeilen 15 bis 20 ein neuer Zustand *next* aus dem aktuell betrachtenden Zustand *current* erstellt um anschließend alle Effekte aus der Operation auf den Zustand *next* anzuwenden. Abschließend wird in Zeile 21 bis 25 der Elternzustand auf den aktuell betrachtenden Zustand *current* gesetzt, die aktuell ausgeführte Operation gespeichert und dann in die PriorityQueue *open* hinzugefügt. Jedoch werden die letzten drei Operationen nur ausgeführt, wenn die List *closed* den Zustand *next* nicht schon enthält. Damit wird verhindert, dass bereits abgearbeitete Zustände nochmals besucht werden.

Kann im Planungsproblem kein Zustand gefunden werden, der das Ziel erfüllt, dann wird null zurück gegeben und damit ist das Planungsproblem inkonsistent.

6.1.2 extractPlan

Kann ein Zielzustand erreicht werden, dann wird der Pfad, der zu diesem Zustand führt, benötigt. Dazu gibt die Methode *extractPlan* einen Pfad zurück.

Die Methode *extractPlan* bekommt einen Zustand *node* und ein Planungsproblem *problem* übergeben. Da pro Zustand eine Operation *operator* und ein vorheriger Zustand *parent* gespeichert wird, lässt sich anhand dessen der Pfad extrahieren. In Listing 53 ist die Methode *extractPlan* aufgeführt.

```
1  private Plan extractPlan(Node node, CodedProblem problem) {
2     Node n = node;
3     final Plan plan = new SequentialPlan();
4     while (n!=null && n.getOperator() != -1) {
5        final BitOp op = problem.getOperators().get(n.getOperator());
6        plan.add(0, op);
7        n = n.getParent();
8     }
9     return plan;
10 }
```

Listing 53: Methode *extractPlan*

Zeile 3 erstellt einen sequentiellen Pfad, der die gleiche Operation mehrmals speichern kann und die Operationen nach der Zeit ordnet. Anschließend betrachtet die while-Schleife von Zeile 4 bis 8 den Operator, der im aktuell betrachteten Zustand vermerkt ist. Ist ein Operator eingespeichert, dann wird diese Operation aus dem Planungsproblem extrahiert und an die 0. Stelle im Pfad gespeichert, siehe Zeile 6. Dadurch, dass hier jede Operation an die 0. Stelle hinzugefügt wird, ist der

Pfad anschließend in der richtigen Reihenfolge und beginnt am Startzustand. Zeile 7 setzt dann den Elternzustand auf den aktuell betrachtenden Zustand *n*. Die while-Schleife bricht ab, wenn keine Operation abgespeichert wurde oder der aktuelle Zustand null ist. Dies ist der Fall, wenn der Startzustand erreicht ist. Abschließend wird der Pfad in Zeile 9 zurückgegeben.

6.1.3 getFavorableNodes

Während der Suche wird pro Zustand die Anzahl an erreichten Teilzielen berechnet. Dieser Wert wird mit einer zuvor definierten Variable *reachedPG* verglichen und beinhaltet immer den höheren Wert von beiden. Somit ist jeweils die höchste Anzahl an erreichten Teilzielen bekannt.

Die Methode *getFavorableNodes* aus Listing 54 bekommt eine Menge an Zuständen, die vom Startzustand aus erreichbar sind. Da durch die Suche die Anzahl an maximal erreichbaren Teilzielen, Variable *reachedPG*, bekannt ist, kann nun, wie in Ziele 3, über die Menge der erreichbaren Zustände iteriert werden. Anschließend werden all die Zustände in einer Array-Liste gespeichert, die die maximale Anzahl an erreichbaren Teilzielen haben, siehe Zeile 5. In Zeile 7 wird abschließend die Array-Liste mit allen günstigen Zuständen zurück gegeben.

```
1 public ArrayList<Node> getFavorableStates(Set<Node> nodes, int reachedPG){
2    ArrayList<Node> favorableNodes= new ArrayList<>();
3    for (Node n : nodes) {
4       if(n.getNumberOfReachedPartialGoals() == reachedPG)
5          favorableNodes.add(n);
6    }
7    return favorableNodes;
8 }
```

Listing 54: Methode *getFavorableNodes*

Da die Anzahl an maximal erreichten Teilzielen auf den Zuständen basiert, besitzt die Array-List mindestens einen Knoten. Ebenso enthält die Array-List nur unterschiedliche Zustände, da die Zustände aus einer Menge betrachtet wurden.

6.1.4 searchPseudoSolutions

Die Methode *searchPseudoSolutions* aus Listing 55 ist ähnlich wie die Methode *search* aufgebaut. Allerdings werden nun alle Lösungen in einem Zeitschritt gesucht anstatt nach der ersten gefundenen Lösung aufzuhören. Die Zustände aus der übergebenen PriorityQueue *open* sollten dabei alle den gleichen Zeitschritt aufweisen. Dazu sollte die PriorityQueue *open* nach dem Zeitstempel sortieren. Erfüllt ein Zustand das Ziel in einem Zeitschritt *n*, Zeile 13-16, dann wird zunächst dieser Zustand der Liste *pseudoSolutions* hinzugefügt. Anschließend wird die Suche für die weiteren Zustände aus der PriorityQueue *open* fortgeführt. Ist bereits mindestens eine Lösung gefunden, dann ist die Variable *foundSolution* auf true. Wird nun ein Zustand

aus der PriorityQueue *open* entfernt, welches einen größeren Zeitstempel hat als der vorherige Zustand, dann bricht die Suche ab. Damit wurden alle Pseudo-Lösungen im gleichen Zeitschritt gefunden. Zum Schluss gibt die Zeile 21 alle gefundenen Pseudo-Lösungen des Zeitschrittes n zurück.

```
1  List<Node> pseudoSolutions=new ArrayList<Node>();
2  boolean foundSolution=false;
3  int timeStep=0;
4
5  while (!open.isEmpty()) {
6     Node current = open.poll();
7     if(timeStep<current.getTimeStemp()&&foundSolution)
8        break;
9     if(timeStep<current.getTimeStemp())
10       timeStep++;
11    closed.add(current);
12
13    if (current.satisfy(problem.getGoal())) {
14       foundSolution=true;
15       pseudoSolutions.add(current);
16    }
17    else {
18       ...
19    }
20 }
21 return pseudoSolutions;
```

Listing 55: Methode *searchPseudoSolutions*

6.2 Maß der Teilziele

Das Maß der Teilziele, ist definiert als:

$$
\begin{aligned}
&I_{Teilziele} : \Pi \rightarrow \mathbb{R}^{\geq 0} \\
&I_{Teilziele}(\Pi) = \frac{\text{Anzahl unerreichter Teilziele}}{\text{Anzahl aller Teilziele}} = \frac{n}{m}, \\
&\text{für } 0 \leq n \leq m
\end{aligned}
$$

Zu Beginn wird die Anzahl an Teilzielen m wie in Zeile 1 des Listing 56 berechnet, in dem die Kardinalität auf das Ziel ausgeführt wird und in die Variable *numberPG* gespeichert wird. Anschließend wird die Methode *search* aufgerufen und eine Lösung für das Planungsproblem wird gesucht.

Wird bei der Suche eine Lösung gefunden, dann ist das Planungsproblem konsistent und das Maß der Teilziele berechnet eine 0. Kann kein Zielzustand gefunden

werden, dann erfolgt die Berechnung wie in Listing 56. In Zeile 2-3 wird die Methode *getFavorableNodes* aufgerufen, die alle günstigen Zustände zurück gibt. Da alle günstigen Zustände die gleiche Anzahl an erreichten Teilzielen l hat, genügt es den ersten Knoten der zurückgegebenen Liste zu betrachten, siehe Zeile 4. Auf diesem Knoten kann anschließend die Berechnung des Maßes ausgeführt werden, in dem die Anzahl an unerreichten Teilzielen $n = m - l$, wie in Zeile 13, berechnet wird. Für die Normalisierung wird in Zeile 14 der Wert der unerreichten Teilziele n abschließend durch die Anzahl an Teilzielen m dividiert.

```
1  int numberPG=goal.cardinality();
2  ArrayList<Node> favorableNodes=getFavorableStates(
3        close,maxReachedPG);
4  double incPG=calcIncPG(favorableNodes.get(0),numberPG);
5  System.out.println("Inkonsistenz durch das
6        Mass der Teilziele: "+incPG);
7
8  public double calcIncPG(Node favorableNode, int numberPG){
9     if(numberPG <=0){
10       return 1.0;
11    }
12    int maxReachedPG=favorableNode.getNumberOfReachedPG();
13    int unreachedPG=numberPG-maxReachedPG;
14    return (double)unreachedPG/(double)numberPG;
15 }
```

Listing 56: Berechnung der Inkonsistenz mit dem Maß der Teilziele

6.3 Maß der fehlenden Atome

Das Maß der fehlenden Atome ist definiert als:

$$I_{FehlendeAtome} : \Pi \to \mathbb{R}^{\geq 0}$$

$$I_{FehlendeAtome}(\Pi) = \begin{cases} \dfrac{\text{minimale Anzahl fehlender Atome}}{\text{Anzahl aller benötigten Atome}} = \dfrac{u}{v}, & \text{für } 0 \leq u \leq v \\ 1, & \text{sonst} \end{cases}$$

Für das Maß der fehlenden Atome beginnt die Suche nach einer Pseudo-Lösung für jeden günstigen Zustand extra, siehe Listing 57. Dabei wird pro Operation die Vorbedingung mit dem günstigen Zustand verglichen. Die Atome, die nicht erfüllt sind, werden dem Zustand *newInit* hinzugefügt, der zu Beginn der Startzustand ist. Dies geschieht in Zeile 15. Um zu verhindern, dass der Startzustand zu einem Zielzustand wird, werden die Atome, die im Ziel beschrieben sind, nicht hinzugefügt, siehe Zeile 14. Anschließend wird in Zeile 22 der neue Startzustand der Priority-Queue *open* hinzugefügt und die Suche nach Pseudo-Lösungen beginnt in Zeile 25.

Wenn mindestens eine Pseudo-Lösung für das Planungsproblem gefunden wurde, dann bricht die Suche für den aktuell betrachteten günstigen Zustand ab. Diese Suche wird mit allen günstigen Zuständen wiederholt.

```
1  boolean foundSolution=false;
2  for(Node n: favorableNodes){
3     foundSolution=false;
4     BitExp newInit=new BitExp(init);
5     while(!foundSolution){
6        for(BitOp op: operations){
7           BitVector posPre=op.getPreconditions().getPositive();
8           BitVector negPre=op.getPreconditions().getNegative();
9           bitValue=0;
10          for(int i=0; i<posPre.cardinality();i++){
11             int tmp=posPre.nextSetBit(bitValue);
12             bitValue=tmp+1;
13             if(!n.get(tmp))
14                if(!goal.getPositive().get(tmp))
15                   newInit.getPositive().set(tmp);
16          }
17          ...
18       }
19
20       open.clear();
21       Node beginning=new Node(new BitState(newInit));
22       open.add(beginning);
23       closed.clear();
24       int oldSize=pseudoSolutions.size();
25       pseudoSolutions.addAll(searchPseudoSolutions(problem,open,closed));
26       if(oldSize<pseudoSolutions.size())
27          foundSolution=true;
28    }
29 }
```

Listing 57: Berechnung der Pseudo-Lösungen im Maß der fehlenden Atome

Sind alle Pseudo-Lösungen gefunden, dann wird das Maß für jede Pseudo-Lösung wie in Listing 58 berechnet. Zunächst wird in Zeile 5 der Pfad mit der Methode *extractPlan* extrahiert. Anschließend wird für jede Operation des Pfades die Anzahl an benötigten Atomen und die Anzahl an nicht verfügbaren Atomen ermittelt. Dafür werden sowohl die positiven als auch die negativen Atome, Zeile 8 und 9, aus der Vorbedingung geholt. Anschließend wird überprüft, ob jedes Atom im aktuellen Zustand *next* vorhanden ist, siehe Zeile 14. Ist das Atom nicht erfüllt, dann wurde es für die Pseudo-Lösung dem Startzustand hinzugefügt und die Variable *missedAtoms* wird um eins erhöht. Zusätzlich wird die Variable *usedAtoms* in Zeile

16 für jedes vorkommende Atom um eins erhöht. Zeile 19 bis 22 führen dann die Effekte auf den Zustand *next* aus, der zu Beginn der Suche dem ursprünglichen Startzustand entspricht. Abschließend berechnen die Zeilen 23 bis 26 den Inkonsistenzwert und speichern diesen in eine Liste *incValues* ab. Der Inkonsistenzwert des Planungsproblems entspricht dann dem Minimum der Liste *incValues*.

```
1  for(Node n: pseudoSolutions){
2     Node next=new Node(new BitState(init));
3     int missedAtoms=0;
4     int usedAtoms=0;
5     plan=extractPlan(n,problem);
6     List<BitOp> operators=plan.actions();
7     for(BitOp op: operators){
8        BitVector posPre=op.getPreconditions().getPositive();
9        BitVector negPre=op.getPreconditions().getNegative();
10       bitValue=0;
11       for(int b=0; b<posPre.cardinality();b++){
12          int tmp=posPre.nextSetBit(bitValue);
13          bitValue=tmp+1;
14          if(!next.get(tmp))
15             missedAtoms++;
16          usedAtoms++;
17       }
18       ...
19       List<CondBitExp> effects = op.getCondEffects();
20       for (CondBitExp ce : effects)
21          next.apply(ce.getEffects());
22    }
23    double inc=1.0;
24    if(usedAtoms>0){
25       inc=(double)missedAtoms/(double)usedAtoms;
26    incValues.add(inc);
27 }
```

Listing 58: Berechnung der Inkonsistenz mit dem Maß der fehlenden Atome

6.4 Maß der Vorbedingungen

Das Maß der Vorbedingungen ist definiert als:

$$I_{Vorbedingung} : \Pi \rightarrow \mathbb{R}^{\geq 0} :$$

$$I_{Vorbedingung}(\Pi) = \begin{cases} \dfrac{\text{Anzahl ignorierter Vorbedingungen}}{\text{Anzahl ausgeführter Operationen}} = \dfrac{x}{y} & \text{für } 0 \leq x \leq y \\ 1, & \text{sonst} \end{cases}$$

Zunächst wird im gegebenen Planungsproblem nach einem Zielzustand gesucht. Kann kein Zielzustand gefunden werden, dann wird das Planungsproblem wie in Listing 59 geändert. Dazu werden wie in Zeile 1 bis 6 die Vorbedingungen aus allen Operationen entfernt werden. Zeile 8 bis 12 erstellt eine neue PriorityQueue *open*, die die Zustände nach dem Zeitstempel sortiert. Zusätzlich bekommen allen günstigen Zustände den Zeitstempel 0 und werden in die PriorityQueue *open* eingefügt. Anschließend beginnt eine erneute Suche von allen günstigen Zuständen, die jedoch das abgeänderte Planungsproblem übergeben bekommt, siehe Zeile 20. Die Suche wird über die Methode *searchPseudoSolutions* realisiert um alle Pseudo-Lösungen, die in einem Zeitschritt n auftauchen, zu finden. Dies ist nötig um den jeweils kleinsten Inkonsistenzwert unter den verschiedenen Pseudo-Lösungen berechnen zu können.

```
1  CodedProblem pbWithoutPre= new CodedProblem(problem);
2  List<BitOp> operators = pbWithoutPre.getOperators();
3  BitExp noPre= new BitExp();
4  for(BitOp op: operators){
5     op.setPreconditions(noPre);
6  }
7
8  PriorityQueue<Node> open = new PriorityQueue<>(100, new Comparator<Node>()
9     public int compare(Node n1, Node n2) {
10       return Double.compare(n1.getTimeStemp(), n2.getTimeStemp());
11    }
12 });
13
14 for(Node n: favorableNodes){
15    n.setTimeStemp(0);
16    open.add(n);
17 }
18 Set<Node> closed=new HashSet<Node>();
19 List<Node> pseudoSolutions = new ArrayList<Node>();
20 pseudoSolutions=searchPseudoSolutions(pbWithoutPre,open,closed);
```

Listing 59: Berechnung der Pseudo-Lösungen

Anschließend wird pro Pseudo-Lösung die Berechnung wie in Listing 60 ausgeführt. Die Anzahl an benötigten Operationen y entspricht der Anzahl an Operationen der Pseudo-Lösung, siehe Zeile 5. Beginnend beim Startzustand, Zeile 9, wird die entsprechende Operation aus der Pseudo-Lösung mit dessen Effekten auf den Zustand *next* ausgeführt, siehe Zeile 20-22. Dazu wird in Zeile 13-15 zunächst die Operation aus dem ursprünglichen Planungsproblem geholt. Anschließend wird in Zeile 17 getestet, ob die Vorbedingung der Operation im Zustand erfüllt ist. Ist dies nicht der Fall, dann wird die Variable *usedOpWithoutPre*, die die Anzahl der ignorierten Vorbedingungen x darstellt, um eins erhöht.

```
1  List<Double> incValues = new ArrayList<Double>();
2  for(Node n: pseudoSolutions){
3     int usedOpWithoutPre=0;
4     Plan plan=extractPlan(n,pbWithoutPre);
5     int usedOp=plan.actions().size();
6
7     List<BitOp> actions=plan.actions();
8     List<BitOp> actionsWithPre=problem.getOperators();
9     BitState init =new BitState(problem.getInit());
10    Node next= new Node(init);
11    for(BitOp action: actions){
12       BitOp actionWithPre= new BitOp(action);
13       for(BitOp action2: actionsWithPre){
14          if(action2.getName()==action.getName())
15             actionWithPre=new BitOp(action2);
16       }
17       if(!actionWithPre.isApplicable(next)){
18          usedOpWithoutPre++;
19       }
20       List<CondBitExp> effects = actionWithPre.getCondEffects();
21       for (CondBitExp ce : effects) {
22          next.apply(ce.getEffects());
23       }
24    }
25    double inc=1.0;
26    if(usedOp>0){
27       inc=(double)usedOpWithoutPre/(double)usedOp;
28    }
29    incValues.add(inc);
30 }
```

Listing 60: Berechnung der Inkonsistenz für die Pseudo-Lösungen

Besitzt die Pseudo-Lösung keine weiteren Operationen mehr, dann kann die Inkonsistenz durch das Maß der Vorbedingung in Zeile 27 berechnet werden. Da die verschiedenen Pseudo-Lösungen unterschiedliche Ergebnisse liefern können, werden die Inkonsistenzwerte der einzelnen Pseudo-Lösungen in Zeile 29 in eine Liste *incValues* gespeichert. Abschließend entspricht der Inkonsistenzwert dem Minimum der Inkonsistenzwerte aus der Liste *incValues*.

Kann selbst durch das Ignorieren der Vorbedingungen in den Operationen der Zielzustand nicht erreicht werden, dann ist das Planungsproblem maximal inkonsistent und die Funktion des Maßes $I_{Vorbedingung}$ gibt eine 1 zurück.

6.5 Maß der kritischen Atome

Das Maß der kritischen Atome ist definiert als:

$$I_{KritischeAtome} : \Pi \rightarrow \mathbb{R}^{\geq 0}$$
$$I_{KritischeAtome}(\Pi) = \frac{\sum^{o \in O} f(o)}{\text{Anzahl ausgeführter Operationen}} = \frac{s}{y},$$
$$\text{für } 0 \leq s \leq y$$

Das Maß der kritischen Atome bedient sich der gleichen Methode zur Berechnung der Pseudo-Lösungen wie das Maß der Vorbedingungen. Somit entspricht die Berechnung der Pseudo-Lösungen dem Listing 59 aus dem Kapitel 6.4.

Um die Inkonsistenz zu berechnen, muss zunächst die Wertigkeit pro Atom bestimmt werden. Dafür wird die Methode *calcAtomWeight* aufgerufen, die in Listing 61 als Pseudocode angegeben ist. Dabei werden die Atome aus allen Operationen betrachtet. Die Variable *counterPre* berechnet die Anzahl pro Atom in den Vorbedingungen, hingegen summiert die Variable *counter* die Anzahl der Atome in den Vorbedingungen und Effekten auf. Somit werden wie in Zeile 4 alle Operationen durchgegangen. Zeile 5 bis 10 erhöht für die entsprechenden Atome die Variablen *counterPre* und *counter* um eins. Anschließend wird die Anzahl der Atome in den Effekten durch die Zeilen 11 bis 15 berechnet. Abschließend berechnet die Schleife von Zeile 16 bis 19 die Wertigkeit pro Atom und speichert diese in der Variable *weight*.

```
1  calcAtomWeight(problem){
2     var<atom, integer> counterPre, counter;
3     var<atom, double> weight;
4     for(var op: problem.getOperators()){
5        for(var a: op.precondition()){
6           var tmp=counterPre.get(a);
7           counterPre.replace(a,tmp++);
8           var tmp2=counter.get(a);
9           counter.replace(a,tmp2++);
10       }
11       for(var e: op.effects()){
12          var tmp=counter.get(a);
13          counter.replace(a,tmp++);
14       }
15    }
16    for(var i=0; i<counter.size();i++){
17       var value=counterPre.get(i)/counter.get(i);
18       weight.put(i,value);
19    }
20 return weight;
```

```
21 }
```

Listing 61: Methode *calcAtomWeight*

Sind die Pseudo-Lösungen gegeben, erfolgt die Berechnung der Inkonsistenz nach Listing 62. Pro Pseudo-Lösung wird zunächst der Pfad aus dem Planungsproblem extrahiert, siehe Zeile 6. Dazu wird in Zeile 11 ein Zustand *next* angelegt, der zu Beginn dem Startzustand entspricht. Anschließend beginnt in Zeile 14 die Berechnung der Wertigkeit pro Operation. Dafür wird zunächst in den Zeilen 18 bis 22 die Operation aus dem Planungsproblem mit den Vorbedingungen extrahiert. Ist dies geschehen, dann wird jedes positive Atom, Zeile 26, und jedes negative Atom aus der Vorbedingung der Operation betrachtet. Ist das aktuell betrachtete Atom aus der Vorbedingung nicht im Zustand *next* vorhanden, siehe Zeile 30 und 31, dann wird die Variable *usedNoneAppliAtomInPre* um die Wertigkeit des Atoms erhöht. Dies bedeutet, dass dieses Atom aus der Vorbedingung nicht erfüllt ist und die Ausführung der Operation dadurch fehlschlägt. Weiterhin wird wie in Zeile 33 die Variable *usedAtomsInPre* um eins erhöht. Diese Variable zählt die Anzahl an Atomen in der Vorbedingung der Operation.

```
1  HashMap<Integer,Double> atomWeight=calcAtomWeight(problem);
2  List<Double> incValues = new ArrayList<Double>();
3  for(Node n: pseudoSolutions){
4     double usedNoneApplicableAtoms=0.0;
5     Plan plan=extractPlan(n,pbWithoutPre);
6     int usedOp=plan.actions().size();
7     List<BitOp> actions=plan.actions();
8     List<BitOp> actionsWithPre=problem.getOperators();
9     BitState init =new BitState(problem.getInit());
10    Node next= new Node(init);
11    int usedAtomsInPre=0;
12    double usedNoneAppliAtomInPre=0;
13
14    for(BitOp action: actions){
15       BitOp actionWithPre= new BitOp(action);
16       usedNoneAppliAtomInPre=0;
17       usedAtomsInPre=0;
18       for(BitOp action2: actionsWithPre){
19          if(action2.getName()==action.getName()){
20             actionWithPre=new BitOp(action2);
21          }
22       }
23       BitVector posPre=actionWithPre.getPreconditions().getPositive();
24       BitVector negPre=actionWithPre.getPreconditions().getNegative();
25       int bit=0;
26       for(int i=0; i<posPre.cardinality();i++){
```

```
27            int bitValue=posPre.nextSetBit(bit);
28            bit=bitValue+1;
29
30            if(!next.get(bitValue)){
31               usedNoneAppliAtomInPre+=atomWeight.get(bitValue);
32            }
33            usedAtomsInPre++;
34         }
35         ...
36         List<CondBitExp> effects = actionWithPre.getCondEffects();
37         for (CondBitExp ce : effects) {
38            next.apply(ce.getEffects());
39         }
40         if(usedAtomsInPre>0){
41            usedNoneApplicableAtoms+=((double)usedNoneAppliAtomInPre/(double)u
42         }
43      }
44      double inc=1.0;
45      if(usedOp>0){
46         inc=(double)usedNoneApplicableAtoms/(double)usedOp;
47      }
48      incValues.add(inc);
49   }
```

Listing 62: Berechnung der Inkonsistenz mit dem Maß der kritischen Atome

Die Zeilen 36 bis 39 wenden anschließend die Operation auf den Zustand *next* an. Anschließend kann die Wertigkeit der Operation berechnet werden, falls die Anzahl an Atomen in der Vorbedingung größer 0 ist. Dazu wird die Summe der Wertigkeiten der nicht erfüllbaren Atome durch die Anzahl aller Atome geteilt und anschließend auf die Variable *usedNoneApplicableAtoms* addiert.

Der Inkonsistenzwert bildet sich dann aus der Division der Variable *usedNoneApplicableAtomes*, die die Summer der Wertigkeiten aller Operationen aufweist, durch die Anzahl aller ausgeführten Operationen *usedOP*, siehe Zeile 46. Da es mehrere Pseudo-Lösungen geben kann, wird der Inkonsistenzwert in der Liste *incValues* gespeichert. Abschließend entspricht der Inkonsistenzwert dem kleinsten Wert aus der Liste *incValues*.

6.6 Ergebnisse

Da es bereits eine Vielzahl an Planungsproblemen in PDDL gibt, wurden für die Ergebnisse Domänen und Problemspezifikationen aus dem Internet verwendet. Bei konsistenten Planungsproblemen, wurde der Startzustand soweit abgeändert, dass eine Inkonsistenz erzeugt wurde.

Unter [14] lässt sich ein Planungsproblem aus der Logistik finden. Die Datei *logistics.pddl* beschreibt Operationen um Pakete in Flugzeuge oder LKWs zu laden und diese an einen anderen Ort zu transportieren. Die dazugehörige Problemspezifikation aus der Datei *problem.pddl* wurde so geändert, dass das Planungsproblem inkonsistent wurde. Der neue Startzustand und das Ziel des Planungsproblems sind in Listing 63 wiedergegeben.

```
(:init (in-city cdg paris)
   (in-city lhr london)
   (in-city north paris)
   (in-city south paris)
   (at plane lhr)
   (at plane cdg)
   (at truck cdg)
   (at p1 lhr)
)
(:goal (and (at p1 north) (at p2 south)))
```

Listing 63: Startzustand und das Ziel für das Planungsproblem der Logistik

Die Webseite [1] stellt eine PDDL-Datei *travel.pddl* zur Verfügung, die es einer Person ermöglicht sich mit einem Fahrzeug von einem Standort zu einem Anderen zu bewegen. Ebenso ist unter [1] eine Problemspezifikation *travel1.pddl* zu finden. Diese wurde wie in Listing 64 abgeändert.

```
(:init (at jack a) (at bulldozer e)
   (vehicle bulldozer)
   (person jack)
   (road a b) (road b a)
   (road d g) (road g d))
(:goal (and (at bulldozer g) (at jack a)))
```

Listing 64: Startzustand und das Ziel für das Planungsproblem Travel

Unter [2] ist die Datei *wumpus-adl.pddl* zu finden. Diese Datei spezifiziert die Wumpus Welt in PDDL und benutzt bedingte Effekte sowie Objekttypen. Die Problemspezifikation wurde der Dabei *wumpus-adl-1.pddl* entnommen. Dabei wurde der Startzustand so verändert, dass das Planungsproblem inkonsistent wurde. Das Ziel und der neue Startzustand sind dem Listing 65 zu entnehmen.

```
(:init (adj s-1-1 s-1-2) (adj s-1-2 s-1-1)
    (adj s-1-2 s-1-3) (adj s-1-3 s-1-2)
    (adj s-1-2 s-2-2) (adj s-2-2 s-1-2)
    (adj s-1-3 s-2-3) (adj s-2-3 s-1-3)
    (pit s-1-2)
    (at gold-1 s-1-3)
    (at agent-1 s-1-1)
    (alive agent-1)
```

	logistic	travel	wumpus
Teilziele	0,5	0,5	0,33
fehlende Atome	0,429	0,75	0,4
Vorbedingung	0,286	1.0	1.0
kritische Atome	0,714	0,442	0,167

Abbildung 9: Inkonsistenzwerte für die jeweiligen Planungsprobleme

Inkonsistenzmaß	**Beispiel aus 4.2.1**	**Beispiel aus 4.2.2**	**Beispiel aus 4.2.3**	**Beispiel aus 4.2.4**
Teilziele	$0,33$	$0,33$	1	$0,5$
fehlende Atome	1	$0,12$	1	$0,4$
Vorbedingung	1	$0,286$	$0,5$	1
kritische Atome	1	$0,089$	$0,5$	$0,361$

Abbildung 10: Inkonsistenzwerte der Maße aus Kapitel 4.2

```
9       (have agent-1 arrow-1)
10      (at wumpus-1 s-2-3)
11      (alive wumpus-1)
12  )
13  (:goal (and (have agent-1 gold-1) (at agent-1 s-1-1) (alive agent-1)))
```

Listing 65: Startzustand und das Ziel für das Planungsproblem Wumpus

Die Inkonsistenzwerte der Inkonsistenzmaße für die jeweiligen Planungsprobleme sind in der Abbildung 9 aufgeführt.

7 Diskussion

Seien die Inkonsistenzmaße und die Beispiele aus dem Kapitel 4.2 gegeben. Die Abbildung 10 führt die Inkonsistenzwerte der Maße für jedes Beispiel auf.

Das Planungsproblem aus dem Kapitel 4.2.1 ist selbst durch das Ignorieren aller Vorbedingungen oder durch die Hinzunahme von Atomen nicht lösbar. Daher bilden alle Maße, bis auf das Maß der Teilziele, auf eine 1 ab. Da jedoch einige Teilziele erreicht werden können, errechnet das Maß der Teilziele einen Inkonsistenzwert von $0,33$.

Die Berechnungen für das Planungsproblem aus dem Kapitel 4.2.2 für die einzelnen Maße ergeben sich wie folgt:

$$
\begin{aligned}
I_{Teilziele}(\Pi) &= \frac{n}{m} = \frac{1}{3} \approx 0,33 \\
I_{FehlendeAtome}(\Pi) &= \frac{u}{v} = \frac{5}{42} \approx 0,12 \\
I_{Vorbedingung}(\Pi) &= \frac{x}{y} = \frac{4}{14} \approx 0,286 \\
I_{KritischeAtome}(\Pi) &= \frac{s}{y} = \frac{3 \cdot \frac{1}{4} + \frac{1}{2}}{14} \approx 0,089
\end{aligned}
\tag{74}
$$

Das Teilziel H kann als einziges nicht erreicht werden, also $x = 1$. Da es drei Teilziele gibt, berechnet das Maß der Teilziele einen Inkonsistenzwert von $0,33$. Das Inkonsistenzmaß der Vorbedingung berechnet einen Wert von ungefähr $0,286$. Insgesamt werden $y = 14$ Operationen benötigt. Davon kann die Operation *produceF* vier mal nicht ausgeführt werden. Die Operation *produceH* ist jedoch anschließend ausführbar. Damit ergibt sich die Anzahl an ignorierten Vorbedingungen mit $x = 4$. Das Maß der kritischen Atome benötigt ebenfalls $y = 14$ Operationen. Die Operation *produceF* hat zwei Atome in der Vorbedingung, wobei das Atom e drei mal im Startzustand enthalten ist. Daher ist die Wertigkeit der Operation *produceF* drei mal $0,25$, da nur das Atom i fehlt. Beide Atome haben eine Wertigkeit von $0,5$. Da während der vierten Ausführung der Operation *produceF* beide Atome nicht erfüllt sind, steigt die Wertigkeit der Operation auf $0,5$ an. Damit ergibt sich insgesamt ein Inkonsistenzwert von $0,089$.

Die Berechnungen für das Planungsproblem aus dem Kapitel 4.2.3 ergeben sich wie folgt:

$$
\begin{aligned}
I_{Teilziele}(\Pi) &= \frac{n}{m} = \frac{1}{1} = 1 \\
I_{FehlendeAtome}(\Pi) &= \frac{u}{v} = 1 \\
I_{Vorbedingung}(\Pi) &= \frac{x}{y} = \frac{1}{2} = 0,5 \\
I_{KritischeAtome}(\Pi) &= \frac{s}{y} = \frac{1}{2} = 0,5
\end{aligned}
\tag{75}
$$

Insgesamt gibt es nur 1 Teilziel, weshalb $m = 1$ ist. Da dieses Teilziel jedoch nicht erreichbar ist, also $n = 1$, bildet das Maß der Teilziele auf eine 1 ab. Ebenso bildet das Maß der fehlenden Atome auf eine 1 ab, da die Hinzunahme von Atomen das Planungsproblem nicht lösen kann. Dies kommt daher, dass die Vorbedingung der Operation *driveTo* von der Entfernung zweier Standort abhängt und diese nicht durch weitere Atome erfüllbar ist. Somit kann die Vorbedingung nicht erfüllt werden. Im Maß der Vorbedingung wird dies jedoch ignoriert und es kann eine Pseudo-Lösung gefunden werden. Dafür wurden insgesamt $y = 2$ Operationen benötigt, von denen einmal die Vorbedingung ignoriert werden musste: $x = 1$.

Somit berechnet das Maß der Vorbedingungen einen Inkonsistenzwert von $0,5$. Da das Maß der kritischen Atome ebenfalls die Vorbedingungen der Operationen ignoriert, kann hier auch eine Pseudo-Lösung gefunden werden. Die Anzahl an benötigten Operationen ist auch hier $y = 2$. Da die Vorbedingung der ersten Operation erfüllbar ist, ist die Wertigkeit für das Ausführen dieser Operation 0. Für das zweite Ausführen der Operation *driveTo* ist das Atom *connection(from, to)<=200* nicht erfüllt. Dieses Atom hat eine Wertigkeit von 1. Da es das einzige Atom der Vorbedingung ist, ist die Wertigkeit dieser Operation ebenfalls eine 1. Dadurch ergibt sich aus der Summe der Wertigkeiten der Operationen: $s = 1$. Damit bildet das Maß der kritischen Atome auf eine $0,5$ ab.

Die Berechnungen für das Planungsproblem aus dem Kapitel 4.2.4 ergeben sich wie folgt:

$$\begin{aligned} I_{Teilziele}(\Pi) &= \frac{n}{m} = \frac{3}{6} = 0,5 \\ I_{FehlendeAtome}(\Pi) &= \frac{u}{v} = \frac{2}{5} = 0,4 \\ I_{Vorbedingung}(\Pi) &= \frac{x}{y} = \frac{2}{2} = 1 \\ I_{KritischeAtome}(\Pi) &= \frac{s}{y} = \frac{\frac{2}{9} + \frac{1}{2}}{2} \approx 0,361 \end{aligned} \tag{76}$$

Die Anzahl an Teilzielen ist für dieses Planungsproblem $m = 6$. Da keine der drei Operationen auf den Startzustand ausgeführt werden kann, ergibt sich die Anzahl an unerreichten Teilzielen durch den Startzustand. Da 3 bereits erfüllt sind, ist die Anzahl an unerreichten Teilzielen $x = 3$. Somit bildet das Maß der Teilziele auf eine $0,5$ ab. Das Maß der fehlenden Atome berechnet die Anzahl an fehlenden Atome mit $u = 2$. Für die Operation *func2* fehlt das Atom b und die Operation *func1* kann ausgeführt werden, wenn das Atom not c hinzugefügt wird. Insgesamt sind $v = 5$ Atome nötig, damit das Planungsproblem konsistent wird. Somit berechnet das Maß der fehlenden Atome eine Inkonsistenz von $0,4$. Das Maß der Vorbedingung bildet auf eine 1 ab, da zwei Operationen nötig sind um das Planungsproblem zu lösen: $y = 2$. Von den beiden Operationen muss jedoch die Vorbedingung ignoriert werden, also $x = 2$. Der Inkonsistenzwert aus dem Maß der kritischen Atome ist ungefähr $0,361$. Die Summe der Wertigkeiten der Operationen setzt sich aus den Wertigkeiten der Operationen *func1* und *func2* zusammen.

7.1 Gemeinsamkeiten und Unterschiede der Maße

Das Maß der Teilziele unterscheidet sich zu den anderen Maßen in der Berechnung der Inkonsistenz. Während die Maße der fehlenden Atome, der Vorbedingung und der kritischen Atome die Inkonsistenz für die Berechnung schrittweise auflösen, falls dies möglich ist, erfolgt die Berechnung des Inkonsistenzwertes im Maß der

Teilziele auf den erreichbaren Zustände des Planungsproblems. Es ist das einzige Maß, welches das Planungsproblem zur Berechnung nicht abändert. Da die Berechnung nicht auf Pseudo-Lösungen angewiesen ist, kann dieses Maß einen Inkonsistenzwert kleiner 1 liefern, auch wenn keine Pseudo-Lösungen gefunden werden können. Dies ist im Beispiel des Kapitels 4.2.1 zu sehen. Der Inkonsistenzwert gibt somit Aufschluss auf die maximale Anzahl an gleichzeitig erreichbaren Teilzielen im Planungsproblem.

Weiterhin ist das Maß der kritischen Atome auf dem Maß der Vorbedingung aufgebaut. Sind auf einer Pseudo-Lösung alle Vorbedingungen der Operationen nicht erfüllt, dann gibt das Maß der Vorbedingung einen Inkonsistenzwert von 1 aus, siehe das Beispiel das Kapitels 4.2.4. Dabei ist nicht ersichtlich, wie viele Atome der Vorbedingung nicht erfüllt sind und wie kritisch sie gegenüber anderen Atomen sind. Somit gibt der Inkonsistenzwert aus dem Maß der kritischen Atomen Aufschluss darüber, welche Schwere die nicht erfüllten Atome in den Vorbedingungen haben. Auch die Beispiele aus dem Kapitel 6.6 spiegeln dies wider. Für das Beispiel der Logistik berechnet der Inkonsistenzwert eine $0,286$. Dies gibt Informationen darüber wie viele Operationen im Pseudo-Pfad nicht ausführbar sind. Hingegen berechnet das Maß der kritischen Atome für das Beispiel der Logistik einen Inkonsistenzwert von $0,714$. Damit ist ersichtlich, dass die Atome, die nicht erfüllt sind, eine hohe Wertigkeit besitzen. Somit führen besonders kritische Atome zur Inkonsistenz des Beispiels Logistik. Im Gegensatz führen im Beispiel Wumpus Atome mit einer geringeren Wertigkeit zur Inkonsistenz.

Ist die Wertigkeit einer Operation gleich 1, dann bildet das Maß der kritischen Atome auf den gleichen Inkonsistenzwert ab, wie das Maß der Vorbedingungen. Dies ist im Beispiel des Kapitels 4.2.3 der Fall.

Das Maß der Vorbedingung zeigt somit wie viele Operationen auf der Pseudo-Lösung nicht ausführbar sind, während das Maß der kritischen Atome Informationen über die Wertigkeiten der nicht erfüllten Atome gibt.

7.2 Ausblick

Die Maße der Vorbedingung und der kritischen Atome berechnen die Inkonsistenz bezüglich der Vorbedingungen auf einer Pseudo-Lösung. Liegt das Interesse jedoch nicht bei der Anzahl an ignorierten Vorbedingungen oder wie kritisch die nicht erfüllten Atome sind, dann kann ein Maß entwickelt werden, dass nur die Anzahl an nicht erfüllten Atome beschreibt. Dies entspricht dem Maß der kritischen Atome, wenn alle Atome eine Wertigkeit von einer 1 aufweisen.

Keins der definierten Maße gewichtet den Zeitschritt der Operationen. Ist zum Beispiel ein Weg von einem Roboter durch ein Hindernis blockiert, dann ist die Inkonsistenz größer, je früher der Roboter auf dieses Hindernis stößt. Somit ist es möglich Operationen nach deren Ausführungsreihenfolge zu gewichten. Operationen, die im Pfad näher am Startzustand nicht ausführbar sind, erhalten eine größere Gewichtung als eine Operation, die kurz vor dem Zielzustand nicht ausführbar ist.

8 Fazit

Dadurch, dass sich die Inkonsistenzmessung auf Planungsprobleme anwenden lässt, kann eine Ordnung zwischen inkonsistenten Planungsproblemen geschaffen werden. Weiterhin wurden Maße für Planungsprobleme entwickelt, die jeweils unterschiedliche Informationen hervorbringen. Je nach Anwendung können so Planungsprobleme auf unterschiedliche Betrachtungsweisen sortiert und verglichen werden. Dazu ist ebenso eine Kombination von Maßen denkbar.

Legt die Anwendung Wert auf die Anzahl an erreichbaren Teilzielen, dann ist das Maß der Teilziele geeignet. Wird jedoch nach der geringsten Anzahl an zu ignorierenden Vorbedingungen gesucht, dann bietet sich das Maß der Vorbedingung an. Besitzen Planungsprobleme kritische Atome, dann kann die Anwendung darauf abzielen das Planungsproblem zu finden, welches die meisten kritischen Atome erfüllt. Dafür eignet sich das Maß der kritischen Atome. Dies ist im Beispiel des Roboters gegeben, der Getränke zwischen zwei Räumen transportieren soll. Spielen diese Kriterien keine Rolle und es wird nach einer Pseudo-Lösung gesucht, die so viele Operationen wie möglich ausführen kann, ohne dass eine Vorbedingung fehlschlägt, dann kann ein Maß bezüglich der Zeitschritte hilfreich sein. Dadurch können Planungsprobleme extrahiert werden, deren Operationen bereits nah am Startzustand nicht mehr auf Zustände ausführbar sind. Ist keine Operation auf einen Zustand anwendbar und das Ziel ist zum Beispiel durch eine Operation erreichbar, dann würde das zweite Planungsproblem auf einen kleineren Inkonsistenzwert abgebildet werden. Zusätzlich könnte ein Maß der Zeitschritte mit dem Maß der Vorbedingung kombiniert werden.

Bei der Anwendung der Inkonsistenzmessung auf die Planungsprobleme konnte jedoch kein vorgestelltes Maß alle Eigenschaft erfüllen. Besonders die Eigenschaft der Monotonie im Ziel ist für die drei Maße, die eine Pseudo-Lösung suchen, nicht erfüllt. Die Anzahl an Eigenschaften auf einem Planungsproblem ist jedoch um einiges größer als bei der Inkonsistenzmessung. Dies liegt daran, dass Eigenschaften für den Startzustand, das Ziel und die Operationen definierbar sind. Kann ein Maß nicht alle Eigenschaften erfüllen, dann kann es dennoch angewandt werden und je nach Anwendungsfall nützlich sein. Neben den Eigenschaften der Inkonsistenzmessung konnten auch Besonderheiten im Planungsproblem festgestellt werden. Die daraus resultierenden kritischen Atome wurden zu einem Maß erweitert.

Alles in allem konnten vier Maße entwickelt werden, die Planungsprobleme mit einer Funktion auf einen Wert abbilden. Dadurch ist es möglich die Planungsprobleme anhand ihres Inkonsistenzwertes zu ordnen. Durch die Normalisierung ist sogar ein direkter Vergleich möglich. Planungsprobleme mit einem Inkonsistenzwert nahe der 1 sind somit inkonsistenter und damit weiter vom Ziel entfernt als ein Planungsproblem, dessen Inkonsistenzwert auf einen Wert nahe der 0 abbildet.

Literatur

[1] *Planungsproblem travel.pddl*. https://www.ida.liu.se/~TDDC17/info/labs/planning/strips/. – Besucht am 28.12.2018

[2] *Planungsproblem wumpus.pddl*. https://www.ida.liu.se/~TDDC17/info/labs/planning/pddl/. – Besucht am 28.12.2018

[3] Strips: A new approach to the application of theorem proving to problem solving. In: *Artificial Intelligence*

[4] AMMOURA, Meriem ; RADDAOUI, Badran ; SALHI, Yakoub ; OUKACHA, Brahim: On Measuring Inconsistency Using Maximal Consistent Sets. In: *ECSQARU* Bd. 9161, Springer, 2015 (Lecture Notes in Computer Science), S. 267–276

[5] ERIKSSON, Salomé ; RÖGER, Gabriele ; HELMERT, Malte: *Unsolvability Certificates for Classical Planning*. https://aaai.org/ocs/index.php/ICAPS/ICAPS17/paper/view/15734. Version: 2017

[6] FOX, Maria ; LONG, Derek: PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. In: *CoRR* abs/1106.4561 (2011). http://arxiv.org/abs/1106.4561

[7] GHALLAB, Malik ; KNOBLOCK, Craig ; WILKINS, David ; BARRETT, Anthony ; CHRISTIANSON, Dave ; FRIEDMAN, Marc ; KWOK, Chung ; GOLDEN, Keith ; PENBERTHY, Scott ; SMITH, David ; SUN, Ying ; WELD, Daniel: PDDL - The Planning Domain Definition Language. (1998), 08

[8] GRANT, John ; HUNTER, Anthony: Measuring Consistency Gain and Information Loss in Stepwise Inconsistency Resolution, 2011, S. 362–373

[9] HUNTER, Anthony ; KONIECZNY, Sébastien: Measuring Inconsistency Through Minimal Inconsistent Sets. In: *Proceedings of the Eleventh International Conference on Principles of Knowledge Representation and Reasoning*, AAAI Press, 2008 (KR'08). – ISBN 978–1–57735–384–3, 358–366

[10] HUNTER, Anthony ; KONIECZNY, Sébastien: On the measure of conflicts: Shapley Inconsistency Values. In: *Artificial Intelligence* 174 (2010), Nr. 14, 1007 - 1026. http://dx.doi.org/https://doi.org/10.1016/j.artint.2010.06.001. – DOI https://doi.org/10.1016/j.artint.2010.06.001. – ISSN 0004–3702

[11] NAU, Dana ; GHALLAB, Malik ; TRAVERSO, Paolo: *Automated Planning: Theory & Practice*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 2004. – ISBN 1558608567

[12] PEDNAULT, Edwin P. D.: ADL: Exploring the Middle Ground Between STRIPS and the Situation Calculus. In: *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1989. – ISBN 1–55860–032–9, 324–332

[13] PELLIER, D. ; FIORINO, H.: PDDL4J: a planning domain description library for java. In: *Journal of Experimental & Theoretical Artificial Intelligence* 30 (2018), Nr. 1, 143-176. http://dx.doi.org/10.1080/0952813X.2017.1409278. – DOI 10.1080/0952813X.2017.1409278

[14] PELLIER, Damien: *GitHub der PDDL4J-Bibliothek*. https://github.com/pellierd/pddl4j. – Besucht am 28.12.2018

[15] PELLIER, Damien ; FIORINO, Humbert: PDDL4J: a planning domain description library for java. In: *Journal of Experimental & Theoretical Artificial Intelligence* (2017), 12, S. 1–34. http://dx.doi.org/10.1080/0952813X.2017.1409278. – DOI 10.1080/0952813X.2017.1409278

[16] THIMM, Matthias: Inconsistency measures for probabilistic logics. In: *Artificial Intelligence* 197 (2013), 1 - 24. http://dx.doi.org/https://doi.org/10.1016/j.artint.2013.02.001. – DOI https://doi.org/10.1016/j.artint.2013.02.001. – ISSN 0004–3702

[17] THIMM, Matthias: On the Compliance of Rationality Postulates for Inconsistency Measures: A More or Less Complete Picture. In: *KI* 31 (2017), Nr. 1, S. 31–39