# Wie findet man die verantwortlichen Axiome? Axiom-Pinpointing in Beschreibungslogiken

Rafael Peñaloza Nyssen

Institut für Theoretische Informatik
Technische Universität Dresden
penaloza@tcs.inf.tu-dresden.de

**Abstract:** Axiom-Pinpointing bestimmt die für eine Konsequenz verantwortlichen Axiome einer Ontologie und unterstützt dadurch die Suche und Behebung von Fehlern. In der Arbeit [Peñ09b], deren Resultate hier zusammengefasst werden, wurde untersucht, unter welchen Bedingungen sich tableau-artige und auf automaten-basierte Schlußfolgerungsverfahren für Beschreibungslogiken stets zu Pinpointing-Verfahren erweitern lassen. Zusätzlich wurde die Komplexität des Pinpointing-Problems erforscht.

## 1 Motivation

Beschreibungslogiken [BCM+03] (BL) sind logikbasierte Wissensrepräsentationsformalismen, mit deren Hilfe man die wichtigen Begriffe (Konzepte) eines Anwendungsbereichs strukturiert und formal wohlfundiert beschreiben kann. Beschreibungslogiksysteme stellen ihren Benutzern Schlussfolgerungsverfahren zur Verfügung, die es erlauben, implizites Wissen aus dem explizit repräsentierten Wissen abzuleiten. Insbesondere wichtig ist hier das sogenannte Subsumtionsproblem (ist ein gegebenes Konzept Unterkonzept eines anderen), das Instanzproblem (ist ein gegebenes Individuum Instanz eines Konzeptes), das Erfüllbarkeitsproblem (ist ein Konzept in sich widerspruchsfrei) und das Konsistenzproblem (ist die gesamte Wissensbasis widerspruchsfrei). Beschreibungslogiken werden in verschiedenen Anwendungsbereichen (wie Sprachverarbeitung, Datenbanken, Konfiguration, Software Engineering) eingesetzt. Sie eignen sich aufgrund ihrer formalen Semantik und der Existenz praktikabler Schlussfolgerungsverfahren aber insbesondere sehr gut zur Definition sogenannter Ontologien. Zum Beispiel beruht der von der Ontologiearbeitsgruppe des WWW-Konsortiums entwickelte und inzwischen als Standard akzeptierte Vorschlag für eine Ontologiesprache für das Semantische Web, OWL, im wesentlichen auf einer ausdrucksstarken Beschreibungslogik. Auch mehrere große medizinische Ontologien (wie SNOMED CT, GALEN, NCI Thesaurus, FMA) und die biologische Gene Ontology (GO) verwenden als Repräsentationssprache Beschreibungslogiken.

Je größer solche Ontologien werden, desto fehleranfälliger wird der Prozess ihrer Erstellung. Die oben erwähnten Schlussfolgerungsverfahren können bei der Aufdeckung von Fehlern helfen, indem sie dem Ontologie-Ingenieur zeigen, welche Konsequenzen die von ihm erstellte Ontologie hat. Ist ein Konzept unerfüllbar oder die ganze Ontologie inkonsis-

tent, so ist dies klarerweise ein Fehler. Aber auch unintuitive Subsumtions- oder Instanzbeziehungen können auf einen Fehler hinweisen. Zum Beispiel ist die (offenbar unintendierte) Subsumtionsbeziehung zwischen „Amputation of Finger“ und „Amputation of Hand“ (d.h. jede Amputation eines Fingers ist auch eine Amputation der ganzen Hand) eine Konsequenz von SNOMED CT. Um einen derartigen Fehler zu beheben, muss man versuchen, die fehlerhaften Axiome in der Ontologie zu finden. Bei einer sehr großen Ontologie (wie SNOMED CT mit fast 400 000 Axiomen) ist dies per Hand sehr schwierig. Ontologie-Editoren müssen dem Ontologie-Ingenieur daher Hilfsmittel zur Verfügung stellen, die es ihm erlauben, die für eine Konsequenz verantwortlichen Axiome zu finden. Dies ist das Ziel des *Axiom Pinpointing*. Es geht hier also darum, für eine gegebene Konsequenz minimale Teilmengen der Ontologie (MinAs) zu berechnen, aus denen diese Konsequenz bereits folgt. Dies erleichtert es, die Gründe für unintuitive oder falsche Konsequenzen zu finden, und ist damit ein erster Schritt bei der Fehlerkorrektur. Für das Amputationsbeispiel folgt die falsche Subsumtionsbeziehung zwischen „Amputation of Finger“ und „Amputation of Hand“ z.B. bereits aus sechs Konzeptdefinitionen in SNOMED CT.

## 2 Beschreibungslogiken

Beschreibungslogiken (BL) sind eine gut untersuchte Familie logikbasierte Wissensrepräsentationsformalismen [BCM$^+$03]. Beginnend mit atomaren Konzepten (die einstelligen Prädikaten aus der Prädikatenlogik entsprechen) und Rollen (binären Prädikate), erlauben sie die Konstruktion komplexer Konzepte. Eine konkrete BL wird durch die Menge der Konstruktoren, die sie hierfür zur Verfügung stellt, charakterisiert. Typische BL-Konstruktoren sind die Booleschen Konnektiven: Negation ($\neg$), Konjunktion ($\sqcap$) und Disjunktion ($\sqcup$), sowie eingeschränkte existentielle ($\exists$) und universelle ($\forall$) Quantoren. Das Konzept $\exists r.C$ beschreibt die Individuen $x$, für die es ein Individuum $y$ gibt, das mit $x$ über $r$ verbunden ist und zu $C$ gehört. Dual wird $\forall r.C$ von den Individuen $x$ erfüllt, für die jedes mit $x$ über $r$ verbundene $y$ zu $C$ gehört. Die kleinste BL, die diese fünf Konstruktoren zur Verfügung stellt, wird $\mathcal{ALC}$ genannt. Wenn man sich auf die Konstruktoren Konjunktion und existenzielle Restriktion ($\exists r.C$) sowie das universelle Konzept ($\top$, das alle Individuuen enthält) einschränkt, so erhält man die BL $\mathcal{EL}$. Diese BL ist zwar relativ ausdrucksschwach, hat aber den Vorteil, dass für sie alle Schlussfolgerungsprobleme in polynomieller Zeit lösbar sind. Eine Wissensbasis (*Ontologie*) ist eine endliche Menge von Axiomen. Typische BL-Axiome sind Konzeptinklusionen ($C \sqsubseteq D$) und Instanzaussagen ($C(a)$), wobei $C, D$ (möglicherweise komplexe) Konzepte sind und $a$ ein Individuenname ist.

Die formale Semantik von BLen beruht auf dem Begriff der Interpretation. Eine Interpretation ist ein Tupel $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, wobei $\Delta^\mathcal{I}$ eine nicht-leere Menge (Domäne genannt) und $\cdot^\mathcal{I}$ eine Interpretationsfunktion ist, die jeden Individuennamen auf ein Element von $\Delta^\mathcal{I}$, jeden Konzeptnamen auf eine Teilmenge von $\Delta^\mathcal{I}$ und jeden Rollenamen auf eine binäre Relation über $\Delta^\mathcal{I}$ abbildet. Diese Funktion wird, entsprechend der Semantik der Konzeptkonstruktoren, induktiv auf komplexe Konzepte erweitert. Eine Interpretation $\mathcal{I}$ erfüllt das Axiom $C \sqsubseteq D$ genau dann, wenn $C^\mathcal{I} \subseteq D^\mathcal{I}$ gilt, und sie erfüllt $C(a)$ genau dann, wenn

$a^{\mathcal{I}} \in C^{\mathcal{I}}$ gilt. Ein *Modell* einer Ontologie $\mathcal{T}$ ist eine Interpretation, die alle Axiome in $\mathcal{T}$ erfüllt. Wir können nun das Subsumtionsproblem und das Instanzproblem als Beispiele für typische BL Schlussfolgerungsprobleme formal definieren:

- Ein Konzept $C$ *subsumiert* das Konzept $D$ bezüglich $\mathcal{T}$ ($\mathcal{T} \models C \sqsubseteq D$) genau dann, wenn $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ für jedes Modell $\mathcal{I}$ von $\mathcal{T}$ gilt.
- Ein Individuum $a$ ist eine *Instanz* des Konzeptes $C$ bezüglich $\mathcal{T}$ ($\mathcal{T} \models C(a)$) genau dann, wenn $a^{\mathcal{I}} \in C^{\mathcal{I}}$ für jedes Modell $\mathcal{I}$ von $\mathcal{T}$ gilt.

Beschreibungslogiksysteme stellen hochoptimierte Schlussfolgerugsverfahren zur Verfügung, mit deren Hilfe man diese und andere Schlussfolgerungsprobleme entscheiden kann. Allerdings ist man manchmal nicht nur daran interessiert, ob eine Konsequenz gilt, sondern auch warum dies so ist; d.h. man will wissen, welche Axiome für diese Konsequenz verantwortlich sind. Die Bestimmung solcher Axiome wird in der BL-Forschung *Axiom-Pinpointing* genannt.

## 3 Axiom-Pinpointing

Wir führen hier Axiom-Pinpointing nicht für eine spezifische BL und ein spezifisches Schlussfolgerungsproblem ein, sondern geben einen allgemeinen Rahmen für die Definition und Behandlung dieses Problems an. Dazu betrachten wir sogenannte *axiomatisierte Eingaben*. Eine axiomatisierte Eingabe ist ein Tupel $(\mathcal{I}, \mathcal{T})$, wobei $\mathcal{I}$ *Eingabe* und $\mathcal{T}$ *Ontologie* genannt wird. Die Ontologie ist dabei eine endliche Menge, deren Elemente wir *Axiome* nennen. Eine *Eigenschaft* $\mathcal{P}$ ist eine Menge axiomatisierter Eingaben, wobei wir $(\mathcal{I}, \mathcal{T}) \in \mathcal{P}$ als „$\mathcal{I}$ folgt aus $\mathcal{T}$ “ interpretieren. Um zu erklären, warum $\mathcal{I}$ aus $\mathcal{T}$ folgt, versucht Axiom-Pinpointing alle minimalen Teilmengen $\mathcal{T}' \subseteq \mathcal{T}$, aus denen $\mathcal{I}$ noch folgt, zu bestimmen. Betrachten wir z.B. die Subsumtion von Konzepten bezüglich einer Ontologie. Wenn eine abgeleitete Subsumtion unerwünscht ist (wie bei unserem Amputationsbeispiel in SNOMED CT), kann man versuchen, die eine möglichst kleine Menge von Axiome, aus denen diese Subsumtion noch folgt, zu finden.

Damit die Bestimmung minimaler Axiomenmengen sinnvoll ist, müssen wir uns offensichtlich auf Eigenschaften einschränken, die monoton sind, d.h. wenn eine Eingabe $\mathcal{I}$ aus einer Ontologie $\mathcal{T}$ folgt, dann folgt $\mathcal{I}$ aus jeder Obermenge $\mathcal{T}' \supseteq \mathcal{T}$. Wir nennen solche Eigenschaften *Konsequenzeigenschaften* (kurz K-Eigenschaften). Das Subsumtionsproblem und das Instanzproblem definieren zum Beispiel jeweils eine K-Eigenschaft. Axiom-Pinpointing berechnet alle MinAs für eine axiomatisierte Eingabe bezüglich einer K-Eigenschaft.

**Definition 1 (MinA)** *Sei $\mathcal{P}$ eine K-Eigenschaft und $(\mathcal{I}, \mathcal{T}) \in \mathcal{P}$ eine axiomatisierte Eingabe. Eine Teilmenge $\mathcal{S} \subseteq \mathcal{T}$ wird eine* MinA *genannt, wenn (i) $(\mathcal{I}, \mathcal{S}) \in \mathcal{P}$, und (ii) für jedes $\mathcal{T}' \subset \mathcal{S}$ gilt $(\mathcal{I}, \mathcal{T}') \notin \mathcal{P}$.*

Für eine gegebene K-Eigenschaft und eine gegebene axiomatisierte Eingabe kann es exponentiell viele MinAs geben [BPS07]. Deswegen sucht man nach einer kompakten Kodie-

rung der Menge aller MinAs: der *Pinpointing-Formel*. Für die formale Definition dieser Formel geht man davon aus, dass eine bijektive Funktion $\mathsf{lab}$ gegeben ist, welche die Ontologie $\mathcal{T}$ auf eine Menge von Aussagenvariablen abbildet. Für eine Teilmenge $\mathcal{S} \subseteq \mathcal{T}$ bezeichne $\mathsf{lab}(\mathcal{S})$ die Menge der Aussagenvariablen $\mathsf{lab}(\mathcal{S}) := \{\mathsf{lab}(t) \mid t \in \mathcal{S}\}$.

**Definition 2 (Pinpointing-Formel)** *Sei $\mathcal{P}$ eine K-Eigenschaft und $(\mathcal{I}, \mathcal{T}) \in \mathcal{P}$ eine axiomatisierte Eingabe. Eine* Pinpointing-Formel *für $(\mathcal{I}, \mathcal{T})$ ist eine monoton Boolesche Formel $\varphi$ über $\mathsf{lab}(\mathcal{T})$, sodass für jede Teilmenge $\mathcal{S} \subseteq \mathcal{T}$ gilt: $(\mathcal{I}, \mathcal{S}) \in \mathcal{P}$ gdw. $\mathsf{lab}(\mathcal{S}) \models \varphi$.*

Aus der Pinpointing-Formel kann man alle MinAs extrahieren, ohne die Eigenschaft $\mathcal{P}$ selbst betrachten zu müssen.

**Beispiel 3** *Das Subsumtionsproblem zwischen zwei Konzeptnamen bezüglich einer Ontologie kann wie folgt als eine K-Eigenschaft aufgefasst werden:*

$$\mathcal{P} := \{((C, D), \mathcal{T}) \mid C \sqsubseteq_{\mathcal{T}} D\}.$$

*Als Beispiel betrachten wir die axiomatisierte Eingabe $\Gamma := ((A, B), \mathcal{T})$, wobei $\mathcal{T}$ aus den folgenden Axiomen besteht:*

$$ax_1\colon\ A \sqsubseteq C, \qquad ax_2\colon\ A \sqsubseteq D, \qquad ax_3\colon\ D \sqsubseteq C, \qquad ax_4\colon\ C \sqcap D \sqsubseteq B \tag{1}$$

*Es ist leicht zu sehen, dass $\Gamma \in \mathcal{P}$; die MinAs sind $\{\{ax_1, ax_2, ax_4\},\ \{ax_2, ax_3, ax_4\}\}$, und $(ax_1 \vee ax_3) \wedge ax_2 \wedge ax_4$ ist eine Pinpointing-Formel.*

Es gibt zwei große Klassen von Axiom-Pinpointing-Techniken: die *Black-Box*- und die *Glass-Box*-Technik. Black-Box-Methoden verwenden ein gegebenes Entscheidungsverfahren für die Eigenschaft, ohne dieses zu ändern. Das Pinpointing-Verfahren ruft dieses Entscheidungsverfahren wiederholt auf und nutzt dessen Antworten zur Berechnung einer oder aller MinAs. Black-Box-Methoden sind einfach zu implementieren, solange es einen Algorithmus gibt, der die K-Eigenschaft entscheidet. Jedoch benötigen solche Methoden oft eine große Zahl von Aufrufen des externen Entscheidungsverfahrens. Black-Box-Methoden, die für BL-Axiom-Pinpointing benutzt werden, finden sich z.B. in [BS08, KPHS07, SHCH07, Sun09].

Glass-Box-Methoden modifizieren das ursprüngliche Entscheidungsverfahren. Die Idee hinter dieser Methode ist es, ein Verfahren zu produzieren, das direkt eine Pinpointing-Formel oder alle MinAs liefert, statt nur die K-Eigenschaft zu entscheiden. Ein erstes Verfahren, das eine Pinpointing-Formel für Inkonsistenz von Ontologien der BL $\mathcal{ALC}$ berechnet, wurde in [BH95] beschrieben. Später wurde in [SC03] eine Pinpointing-Methode für Unerfüllbarkeit von Konzepten, auch in der BL $\mathcal{ALC}$, entwickelt. Beide Methoden modifizieren dazu das bekannte tableau-artige Schlussfolgerungsverfahren für $\mathcal{ALC}$. Der einzige wirkliche Unterschied liegt darin, dass das Verfahren von [SC03] direkt MinAs berechnet, während das Verfahren von [BH95] eine Pinpointing-Formel erzeugt. Aufbauend auf diese Arbeiten wurden in der Literatur weitere Pinpointing-Verfahren für verschiedene Beschreibungslogiken und Schlussfolgerungsprobleme entwickelt, die auf Modifikationen existierender tableau-artiger Schlussfolgerungsverfahren beruhen [PSK05, LMP06].

Anstatt für jedes einzelne tableau-artige Verfahren jeweils eine neue Modifikation zu entwickeln, die MinAs oder die Pinpointing-Formel berechnet, wird in dieser Arbeit das grundsätzliche Problem, für welche Arten von Schlussfolgerungsverfahren man stets eine derartige Modifikation vornehmen kann, untersucht. Insbesondere wird für tableau-artige und automaten-basierte Verfahren untersucht, wie man aus diesen auf möglichst allgemeine Weise Verfahren gewinnen kann, die eine Pinpointing-Formel berechnen.

### 3.1 Tableau-artige Axiom-Pinpointing

Tableau-artige Verfahren benutzen Regeln, um implizites Wissen explizit zu machen. Wir geben hier einen allgemeinen Rahmen an, in dem man derartige Verfahren beschreiben kann. Das gegenwärtige Wissen wird durch eine Menge sogenannter $S$-Zustände beschrieben. Ein *$S$-Zustand* ist ein Tuple $(A, \mathcal{T})$, wobei $A$ eine Menge von Assertionen und $\mathcal{T}$ eine Menge von Axiomen ist. Gegeben eine axiomatisierte Eingabe $\Gamma$, verwandelt ein tableau-artiges Verfahren zunächst $\Gamma$ in eine Menge *Anfangs-$S$-Zustände*. Diese Menge wird dann durch die Anwendung von Regeln verändert. *Regeln* haben die Form

$$(B_0, \mathcal{S}) \rightarrow \{B_1, \ldots, B_m\},$$

wobei $B_0, \ldots, B_m$ Mengen von Assertionen sind und $\mathcal{S}$ eine Menge von Axiomen ist. Diese Regel ist auf einen $S$-Zustand $\mathfrak{S} = (A, \mathcal{T})$ anwendbar, wenn $B_0 \subseteq A$ und $\mathcal{S} \subseteq \mathcal{T}$ gilt. Ihre Anwendung ersetzt $\mathfrak{S}$ durch $m$ neue $S$-Zustände $(A \cup B_i, \mathcal{T}), 1 \leq i \leq m$. Um eine unnötige Wiederholung von Regelanwendungen zu vermeiden, können die Regeln nur dann angewendet werden, wenn sie tatsächlich das derzeitige explizite Wissen erweitern; d.h. es müssen alle neuen $S$-Zustände wirklich verschieden von $\mathfrak{S}$ sind. Das Verfahren wendet solange Regeln an, bis die erreichte Menge von $S$-Zuständen *saturiert* ist, d.h. keine Regel mehr darauf anwendbar ist.

Die verschiedenen $S$-Zustände in einem bestimmten Schritt der Ausführung des Verfahrens repräsentieren die nicht-deterministischen Entscheidungen, die getroffen werden können. Wenn das Verfahren saturiert ist, werden alle $S$-Zustände auf *Konflikte* untersucht. Die axiomatisierte Eingabe ist genau dann *akzeptiert* (d.h. es gilt die K-Eigenschaft), wenn nach der Saturation alle $S$-Zustände einen Konflikt enthalten. Die genaue Definition unseres allgemeinen Rahmens für tableau-artige Verfahren können in [BP07, BP10b] nachgelesen werden.

Wir skizzieren nun die dort entwickelte Methode, die ein tableau-artiges Entscheidungsverfahren für eine K-Eigenschaft $\mathcal{P}$ in ein Verfahren, dessen Ausgabe eine Pinpointing-Formel ist, umwandelt. Das so erhaltene Verfahren wird die *Pinpointing-Erweiterung* des Entscheidungsverfahren genannt. Die Grundidee hinter der Pinpointing-Erweiterung ist die Einführung einer *Tracing-Technik*. Bei dieser Technik erhält jede Assertion $a$ eine Markierung $\mathsf{lab}(a)$, die die Axiome, die für das Vorhandensein dieser Assertion in einen bestimmten $S$-Zustand verantwortlich sind, beschreibt. Die Tableau-Regeln werden so verändert, dass durch eine Regelanwendung eingeführte Assertionen mit einer geeigneten Markierung versehen werden und die Markierung bereits vorhandener Assertionen geeignet modifiziert wird. Zusätzlich muss auch die Definition der Regelanwendbarkeit

angepasst werden.

Genauer betrachtet markiert die Tracing-Technik Assertionen in $S$-Zuständen mit einer monotonen Boolesche Formel wie folgt. Zunächst wird eine Assertion $a$ in einem Anfangs-$S$-Zustände mit der Tautologie $\top$ markiert, wenn sie unabhängig von Axiomen ist, und mit der Formel $\bigvee_{t \in \mathcal{T}_a} \mathsf{lab}(t)$, wenn die Anwesenheit von $a$ vom Vorhandensein der Axiome in $\mathcal{T}_a \subseteq \mathcal{T}$ abhängt.

Wenn die Regel $(B_0, \mathcal{S}) \to \{B_1, \ldots, B_m\}$ auf einen $S$-Zustand $\mathfrak{S}$ angewandt wird, dann wird $\mathfrak{S}$ durch $m$ neue $S$-Zustände ersetzt. Da die Regel nur wegen des Vorhandenseins der Assertionen in $B_0$ und der Axiome in $\mathcal{S}$ anwendbar ist, bekommen die neuen Assertionen, die bei dieser Regelanwendung in die $S$-Zustände aufgenommen werden, eine Markierung $\psi$, die dies ausdrückt:

$$\psi := \bigwedge_{a \in B_0} \mathsf{lab}(a) \wedge \bigwedge_{s \in S} \mathsf{lab}(s).$$

Da es vorkommen kann, dass eine Assertion $a$ in der rechten Seite der Regel in dem Man muss aber auch den Fall betrachten, bei dem eine Assertion $a$ bereits mit einer Markierung $\mathsf{lab}(a) = \phi$ im $S$-Zustand vorhanden ist. Durch die Regelanwendung wird die Markierung für diese Assertion dann zu $\phi \vee \psi$ geändert, wobei $\psi$ wie oben definiert ist. Die Disjunktion drückt aus, dass es mehr als eine Möglichkeit dafür geben kann, dass $a$ generiert wird. Wenn man die Pinpointing-Formel berechnet, muss man alle diese verschiedenen Möglichkeiten beachten.

Wie bei dem ursprünglichen Tableau-Verfahren kann eine Regel nur angewendet werden, wenn die Anwendung tatsächlich die Menge aller $S$-Zustände erweitert. Für das Entscheidungsverfahren bedeutet dies, dass jeder neue $S$-Zustand mehr Assertionen als $\mathfrak{S}$ enthalten muss. In der Pinpointing-Erweiterung ist es möglich, einen neuen $S$-Zustand zu erhalten, der genau die gleiche Menge von Assertionen enthält, die Regel aber trotzdem anwendbar ist, weil dadurch die Markierungen einiger Assertionen zu allgemeineren Boolesche Formeln abgeändert werden.

Wenn die Menge der (markierten) $S$-Zustände saturiert ist, d.h. wenn es keine anwendbare Regel mehr gibt, sucht man nach Konflikten in diesen End-$S$-Zuständen. Jeder Konflikt $C$ hat seine eigene Markierung, die ausdrückt, welche Axiome für $C$ verantwortlich sind. Diese Markierung erhält man aus den Markierungen der für den Konflikt verantwortlichen Assertionen und Axiomen. Ein einzelner $S$-Zustand kann mehr als einen Konflikt enthalten, aber einer genügt für das ursprüngliche Tableau-Verfahren, um die axiomatisierte Eingabe zu akzeptieren. Deswegen müssen die Markierungen der verschiedenen Konflikte eines $S$-Zustands disjunktiv kombiniert werden. Die Eingabe wird nur akzeptiert, wenn jeder $S$-Zustand einen Konflikt enthält. Deshalb werden die Markierungen aller $S$-Zustände durch eine Konjunktion verbunden. In [BP07, BP10b] wurde bewiesen, dass die so erhaltene Formel wirklich eine Pinpointing-Formel ist; d.h. wenn $\mathfrak{S}_1, \ldots, \mathfrak{S}_n$ alle $S$-Zustände nach der Saturation sind, dann ist die folgende Formel eine Pinpointing-Formel:

$$\bigwedge_{i=1}^{n} \bigvee_{C \text{ Konflikt in } \mathfrak{S}_i} \mathsf{lab}(C).$$

Man benötig eine saturierte Menge von $S$-Zuständen, damit man die Pinpointing-Formel

in der oben beschriebenen Weise berechnen kann. Deshalb ist es wichtig, dass die Pinpointing-Erweiterung stets terminiert. Da man von einem tableau-basierten Entscheidungsverfahren ausgeht, ist die Terminierung des Ausgangsverfahrens gegeben. Es stellt sich aber die Frage, ob sich diese Eigenschaft stets auf die Pinpointing-Erweiterung überträgt. Leider musste diese Frage negativ beantwortet werden: in [BP07, BP10b] werden Beispiele für terminierende Tableau-Verfahren angegeben, deren Pinpointing-Erweiterung nicht terminiert. Es ist sogar möglich zu zeigen, das es unentscheidbar ist, ob die Pinpointing-Erweiterung eines terminierenden Tableau-Verfahrens ebenfalls terminiert. Dies war ein sehr überraschendes Ergebnis, da bei der Modifikation spezieller tableau-artiger Verfahren zu Pinpointing-Verfahren die Terminierung nie ein großes Problem darstellte (oder zumindest von den Autoren der Arbeiten nicht als eine solches angesehen wurde).

Um die Terminierung der Pinpointing-Erweiterung garantieren zu können, wurde eine eingeschränktere Klasse von Tableau-Verfahren (sogenannte *Forest Tableaux*), die baumartige Modelle erzeugen, betrachtet. Schränkt man diese noch etwas weiter ein (zu sogenannten *Ordered Tableaux*), so erhält man eine Klasse tableau-artiger Verfahren, die stets terminieren und für die sich die Terminierung auch auf ihre Pinpointing-Erweiterung überträgt. Nicht terminierende Tableau-Verfahren können manchmal in terminierende umgewandelt werden, indem man die sogenannte *Blockiertechnik* einsetzt. Für Forest Tableaux kann man eine Variante dieser Blockertechnik einführen, und zeigen, dass auch die Pinpointing-Erweiterung eines solches Verfahrens korrekt und terminierend ist [BP10b].

### 3.2 Automaten-basiertes Axiom-Pinpointing

Automaten-basierte Verfahren reduzieren das Schlussfolgerungsproblem in der Logik auf das Leerheitsproblem des verwendeten Automatenmodells. In dieser Arbeit werden Verfahren betrachtet, die Büchi-Automaten verwenden, welche auf unendlichen Bäumen laufen. Das Leerheitsproblem für Büchi-Automaten kann durch ein iteratives Verfahren, das polynomielle Zeit (in der Anzahl der Zustände) erfordert, entschieden werden [Rab70, VW86, BT01]. Obwohl automaten-basierte Entscheidungsverfahren häufig für BLen eingesetzt werden (siehe z.B. [BHP08, BT01, CDGL99, LS00]), wurden vor unserer Arbeit keine Versuche unternommen, Pinpointing-Erweiterungen derartiger Verfahrens zu konstruieren. Die Schwierigkeit beim Entwurf dieser Pinpointing-Erweiterungen liegt darin, das der Leerheitstest für die Büchi-Automaten nicht konstruktiv ist: er konstruiert keinen akzeptierenden Lauf des Automaten, sondern beweist nur indirekt, dass ein solcher Lauf existiert. Folglich erfährt man durch diesen Test nicht, wie die erfolgreichen Läufe des Automaten aussehen, was die Verwendung einer Tracing-Technik unmöglich macht.

Um überhaupt eine Chance zu haben, ein automaten-basiertes Verfahren zu einem Pinpointing-Verfahren zu erweitern, muss man davon ausgehen, dass bekannt ist, wie die Axiome die Konstruktion des Automaten beeinflussen. Dazu wurde der allgemeine Rahmen des *axiomatischen Automaten* eingeführt. Dieser enthält in gewisser Weise für eine axiomatisierte Eingabe $\Gamma = (\mathcal{I}, \mathcal{T})$ alle Automaten für die Eingaben $(\mathcal{I}, \mathcal{S})$ mit $\mathcal{S} \subseteq \mathcal{T}$. Der Einfluss der Axiome auf die Konstruktion des Automaten wird durch zwei *Beschränkungsfunktionen*, $\Delta\mathsf{res}$ und $\mathcal{I}\mathsf{res}$, ausgedrückt. Sie bilden jedes Axiom in der Ein-

gabe auf eine Menge von Übergänge bzw. eine Menge von Anfangszustände ab. Wenn ein Axiom $t$ zu $\mathcal{S}$ gehört, dann können nur die Übergänge in $\Delta\mathsf{res}(t)$ und die Anfangszustände in $\mathcal{I}\mathsf{res}(t)$ verwendet werden, um einen akzeptierenden Lauf des Automaten zu konstruieren. Je mehr Axiome in $\mathcal{S}$ enthalten sind, desto weniger akzeptierende Läufe existieren damit. Dies steht mit der Tatsache in Einklang, dass die betrachtete Eigenschaft monoton ist und $\mathcal{I}$ aus $\mathcal{S}$ folgt, wenn *kein* akzeptierender Lauf existiert. Obwohl der Begriff der axiomatischen Automaten neu ist, sind bekannte Automaten-Konstruktionen für BLen und Temporallogiken in der Tat axiomatische Automaten, in denen die Beschränkungsfunktionen in der Konstruktion implizit beschrieben sind.

Man kann nun jeden axiomatischen Automat in einen gewichteten Automaten umwandeln, sodass das sogenannte *Verhalten* des gewichteten Automaten genau die Pinpointing-Formel ist [BP08, BP10a]. Die Idee ist, dass das Gewicht eines Laufs $\ell$ ausdrückt, welche Axiome mit $\ell$ in Konflikt stehen, d.h. für welche Axiome $t$ es ein Übergang in $\ell$ gibt, der nicht in $\Delta\mathsf{res}(t)$ steht. Zu diesem Zweck wird das Gewicht der Übergänge wie folgt definiert:

$$\mathsf{wt}(q, q_1, \ldots, q_k) = \bigvee_{\{t \in \mathcal{T} \mid (q_0, q_1, \ldots, q_k) \notin \Delta\mathsf{res}(t)\}} \mathsf{lab}(t).$$

Auf ähnliche Weise wird die Funktion $\mathcal{I}\mathsf{res}$ in eine Anfangszuweisung für den gewichteten Automaten übersetzt.

Da noch kein Verfahren zur effektiven Berechnung des Verhaltens von gewichteten Büchi-Baumautomaten bekannt war, mussten wir ein derartiges Verfahren selbst entwickelt. Dieses Verfahren ist eine Verallgemeinerung des iterativen Entscheidungsverfahren für die Leerheit von Büchi-Baumautomaten, und benötigt ebenfalls polynomielle Zeit in die Anzahl der Zustände. Es wurde dann gezeigt, dass man mit diesem Ansatz Pinpointing-Formeln für die BL $\mathcal{SI}$ und die Temporallogik LTL in exponentieller Zeit [BP10a] und für die BL $\mathcal{EL}$ in polynomieller Zeit [Peñ09a] berechnen kann.

### 3.3 Komplexität

Hier ist es gelungen, Härteresultate für das Pinpointing in Logiken, deren Schlussfolgerungsprobleme polynomiell entscheidbar sind, zu zeigen. Dabei wurden drei Arten von Komplexitäts-Maßen betraschtet. Zunächst wurde gezeigt, dass unter anderem die folgenden Entscheidungsprobleme NP-hart sind: Gibt es eine MinA der Kardinalität $\leq n$?; Gibt es eine MinA, die das Axiom $t$ enthält?. Das zweite Maß betrachtet das Aufzählproblem (erzeuge alle MinAs). Hier wurde gezeigt, dass es nicht möglich ist, alle MinAs in Ausgabepolynomieller Zeit (d.h. in der Zahl der MinAs) zu erzeugen. Das letzte Maß zeigt dass es schwierig ist, die Anzahl der MinAs zu zählen.

## Danksagungen

Der Autor möchte sich bei Franz Baader, Martin Knechtel und Marcel Lippmann für ihre hilfreichen Kommentare und Korrekturvorschläge bedanken.

## Literatur

[BCM+03] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi und Peter F. Patel-Schneider, Hrsg. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[BH95] Franz Baader und Bernhard Hollunder. Embedding Defaults into Terminological Knowledge Representation Formalisms. *J. of Automated Reasoning*, 14:149–180, 1995.

[BHP08] Franz Baader, Jan Hladik und Rafael Peñaloza. Automata Can Show PSPACE Results for Description Logics. *Information and Computation*, 206(9,10):1045–1056, 2008.

[BP07] Franz Baader und Rafael Peñaloza. Axiom Pinpointing in General Tableaux. In Nicola Olivetti, Hrsg., *Proc. of TABLEAUX 2007*, Jgg. 4548 of *Lecture Notes in Artificial Intelligence*, Seiten 11–27, 2007. Springer-Verlag.

[BP08] Franz Baader und Rafael Peñaloza. Automata-based Axiom Pinpointing. In Alessandro Armando, Peter Baumgartner und Gilles Dowek, Hrsg., *Proc. of IJCAR 2008*, Jgg. 4667 of *Lecture Notes in Artificial Intelligence*, Seiten 226–241, 2008. Springer-Verlag.

[BP10a] Franz Baader und Rafael Peñaloza. Automata-based Axiom Pinpointing. *Journal of Automated Reasoning*, 2010. Special Issue: IJCAR 2008. To appear.

[BP10b] Franz Baader und Rafael Peñaloza. Axiom Pinpointing in General Tableaux. *Journal of Logic and Computation*, 20(1):5–34, February 2010.

[BPS07] Franz Baader, Rafael Peñaloza und Boontawee Suntisrivaraporn. Pinpointing in the Description Logic $\mathcal{EL}^+$. In Joachim Hertzberg, Michael Beetz und Roman Englert, Hrsg., *Proc. of KI'07*, Jgg. 4667 of *Lecture Notes in Artificial Intelligence*, Seiten 52–67, 2007. Springer-Verlag.

[BS08] Franz Baader und Boontawee Suntisrivaraporn. Debugging SNOMED CT Using Axiom Pinpointing in the Description Logic $\mathcal{EL}^+$. In *Proc. of KR-MED'08*, Jgg. 410 of *CEUR-WS*, 2008.

[BT01] Franz Baader und Stephan Tobies. The Inverse Method Implements the Automata Approach for Modal Satisfiability. In Rajeev Goré, Alexander Leitsch und Tobias Nipkow, Hrsg., *Proc. of IJCAR 2001*, Jgg. 2083 of *Lecture Notes in Artificial Intelligence*, Seiten 92–106, 2001. Springer-Verlag.

[CDGL99] Diego Calvanese, Giuseppe De Giacomo und Maurizio Lenzerini. Reasoning in Expressive Description Logics with Fixpoints based on Automata on Infinite Trees. In *Proc. of IJCAI'99*, Seiten 84–89, 1999.

[KPHS07] Aditya Kalyanpur, Bijan Parsia, Matthew Horridge und Evren Sirin. Finding All Justifications of OWL DL Entailments. In Karl Aberer, et.al., Hrsg., *Proc. of ISWC 2007 + ASWC 2007*, Jgg. 4825 of *LNCS*, Seiten 267–280, 2007. Springer-Verlag.

[LMP06] Kevin Lee, Thomas Meyer und Jeff Z. Pan. Computing Maximally Satisfiable Terminologies for the Description Logic ALC with GCIs. In Bijan Parsia, Ulrike Sattler und David Toman, Hrsg., *Proc. of DL 2006*, Lake District, UK, 2006.

[LS00] Carsten Lutz und Ulrike Sattler. The Complexity of Reasoning with Boolean Modal Logic. In *Proc. of AiML 2000*, 2000.

[Peñ09a] Rafael Peñaloza. Using Tableaux and Automata for Pinpointing in EL. In Valentin Goranko, Hrsg., *Proc. of AutoTab'09*, 2009.

[Peñ09b] Rafael Peñaloza. *Axiom-Pinpointing in Description Logics and Beyond*. Dissertation, Technische Universität Dresden, 2009.

[PSK05] Bijan Parsia, Evren Sirin und Aditya Kalyanpur. Debugging OWL ontologies. In Allan Ellis und Tatsuya Hagino, Hrsg., *Proc. of WWW'05*, Seiten 633–640. ACM, 2005.

[Rab70] Michael O. Rabin. Weakly Definable Relations and Special Automata. In Y. Bar-Hillel, Hrsg., *Proc. of Symposium on Mathematical Logic and Foundations of Set Theory*, Seiten 1–23. North-Holland Publ. Co., Amsterdam, 1970.

[SC03] Stefan Schlobach und Ronald Cornet. Non-Standard Reasoning Services for the Debugging of Description Logic Terminologies. In Georg Gottlob und Toby Walsh, Hrsg., *Proc. of IJCAI 2003*, Seiten 355–362, 2003. Morgan Kaufmann, Los Altos.

[SHCH07] Stefan Schlobach, Zhisheng Huang, Ronald Cornet und Frank Harmelen. Debugging Incoherent Terminologies. *Journal of Automated Reasoning*, 39(3):317–349, 2007.

[Sun09] Boontawee Suntisrivaraporn. *Polynomial-time Reasoning Support for Design and Maintenance of Large-scale Biomedical Ontologies*. Dissertation, Technische Universität Dresden, 2009.

[VW86] Moshe Y. Vardi und Pierre Wolper. Automata-theoretic Techniques for Modal Logics of Programs. *Journal of Computer and System Sciences*, 32:183–221, 1986.

**Rafael Peñaloza Nyssen** wurde am 5. Juli 1981 in Mexiko-Stadt, Mexiko geboren. Er hat im Dezember 2003 sein Studium der Angewandten Mathematik beim ITAM, Mexiko, mit dem Diplom abgeschlossen. Seine Diplomarbeit beschäftigte sich mit probabilistischen Netzwerken. Nach einem Jahr als Mitarbeiter bei Inffinix Software schrieb er sich für den Studiengang Computational Logic an der Technischen Universität Dresden ein. Im Jahre 2006 erhielte er dort den Grad des Master of Science mit der Arbeit *Optimization of Emptiness Test of Büchi Automata on Infinite Trees*. Danach wurde er Stipendiat im Graduiertenkolleg „Wissensrepräsentation“ an der Universität Leipzig. Seit 2009 ist er wissenschaftlicher Mitarbeiter an der TU Dresden. Mit seiner Dissertation zum Thema *Axiom-Pinpointing in Description Logics and Beyond* hat er dort seine Promotion im Jahre 2009 mit *summa cum laude* abgeschlossen.