

Diplomarbeit

# Erklärung und Korrektur von Nicht-Subsumtion in Ontologien



Julian Lambertz

Universität Ulm

2007

Institut für Künstliche Intelligenz

1. Gutachter Prof. Dr. Friedrich von Henke
2. Gutachter Dr. Thorsten Liebig

---

---

## Zusammenfassung

Durch die fortschreitende Entwicklung des Semantic Web beschäftigen sich auch zunehmend Menschen mit der Modellierung von Ontologien, denen detailliertere Kenntnisse über die zu Grunde liegenden Beschreibungslogiken fehlen. Da die in diesem Kontext verwendeten Inferenzdienste auf sehr komplizierten Algorithmen beruhen, sind die von Reasonern gezogenen Schlussfolgerungen selbst für Experten oftmals schwer nachzuvollziehen. Dies ist aber insbesondere dann essentiell, wenn es darum geht, Fehler in einer Modellierung zu finden. Eine Vereinfachung des Umgangs mit den entsprechenden Anwendungen ist daher von großem Interesse. Die Erklärung von solchen Schlussfolgerungen in möglichst einfacher Weise ist ein Nicht-Standardinferenzdienst, der dem Benutzer die Modellierung von Ontologien erleichtern kann.

Bei der Modellierung einer Ontologie kommt es häufig vor, dass der Benutzer eine Subsumtion zweier Konzepte erwartet, sich diese aber bei Überprüfung mit einem Reasoner als nicht gültig herausstellt. Die Ursachen einer solchen unerwünschten Nicht-Subsumtion zu finden und zu beseitigen, ist in vielen Fällen eine schwierige und zeitintensive Aufgabe.

Es gibt bereits ein große Zahl von Arbeiten, die sich mit der Erklärung und Korrektur von Unerfüllbarkeit beschäftigen. Auch die Erklärung von Subsumtion wurde beispielsweise in [LH05] bereits thematisiert. Im Gegensatz hierzu ist zum Thema der Nicht-Subsumtion noch verhältnismäßig wenig zu finden.

Diese Arbeit beschäftigt sich mit der Erklärung und Korrektur von Nicht-Subsumtion. Die Ideen zur Erklärung basieren auf [LH05]. Es werden Vorschläge vorgestellt, wie eine in einen Tableau-Reasoner integrierte Erklärungskomponente eine Nicht-Subsumtion von zwei Konzepten einer TBox erklären kann. Der Fokus liegt hierbei auf einer natürlichsprachlichen und strukturierten Erklärung, die auch für weniger erfahrene Benutzer ohne Kenntnis des Tableauverfahrens verständlich und nachvollziehbar sein und bei der Suche nach Fehlern helfen soll.

Weiterhin werden Möglichkeiten zur automatischen tableaubasierten Suche nach Modellierungsfehlern vorgestellt, die eine Nicht-Subsumtion verursacht haben können. Das Verfahren basiert auf einer Untersuchung typischer Modellierungsfehler von unerfahrenen Benutzern, die in [RDH<sup>+</sup>04] erfasst wurden. Es wird versucht, solche potentiellen Fehler in den betroffenen Konzeptdefinitionen zu finden. Die

---

Idee ist, dem Benutzer Änderungen aufzuzeigen, durch die die Gültigkeit der gewünschten Subsumtion erreicht werden könnten.

Es wurde untersucht, wie gefundene Änderungsvorschläge bewertet werden können, wobei festgestellt wurde, dass dies sehr schwierig und nicht eindeutig möglich ist. Der hier vorgestellte Ansatz ist eine Bewertung von Änderungen auf Grund ihrer Auswirkung auf die Konzepthierarchie der Ontologie. Zur Erfassung der Änderung in der Konzepthierarchie wurde eine einfache Metrik entwickelt.

Die erarbeiteten Ideen wurden in einer Form einer prototypischen Implementierung umgesetzt. Der tableaubasierter Reasoner liest eine TBox und eine Subsumtionsfragestellung ein und prüft die Subsumtion. Hierbei generiert er im Falle einer Nicht-Subsumtion eine Erklärung und versucht, Änderungsvorschläge zum Erreichen der Subsumtion zu finden. Die Erklärung wird dem Benutzer über eine grafische Benutzerschnittstelle strukturiert präsentiert. Er hat die Option sich etwaige Änderungsvorschläge anzeigen lassen.

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>1</b>
1.1	Semantic Web . . . . .	1
1.2	Motivation . . . . .	2
1.3	Überblick über die Arbeit . . . . .	3
<b>2</b>	<b>Grundlagen</b>	<b>5</b>
2.1	Beschreibungslogik . . . . .	5
2.2	Ontologien . . . . .	6
2.3	Inferenzdienste . . . . .	7
2.3.1	Erfüllbarkeit . . . . .	7
2.3.2	Subsumtion . . . . .	8
2.4	Erklären . . . . .	8
2.5	Tableauverfahren . . . . .	10
2.5.1	Funktionsweise . . . . .	10
2.5.2	Grafische Darstellung des Tableaus . . . . .	12
2.5.3	Lazy Unfolding . . . . .	14
2.6	Aktuelle Inferenzsysteme . . . . .	16
<b>3</b>	<b>Erklären</b>	<b>17</b>
3.1	Allgemeines . . . . .	17
3.2	Vorgehen . . . . .	17
3.3	Basisfälle der (Nicht-)Subsumtion . . . . .	20
3.4	Vereinfachungen . . . . .	22
3.5	Sonderfälle . . . . .	22
3.6	Beispiel . . . . .	23
<b>4</b>	<b>Korrektur</b>	<b>25</b>
4.1	Grundlegendes . . . . .	25
4.2	Typische Fehler bei der Ontologie-Modellierung . . . . .	26
4.2.1	Domain- und Range-Einschränkungen . . . . .	27
4.2.2	Partielle statt vollständige Definition . . . . .	27
4.2.3	Universelle statt existentielle Qualifikation . . . . .	28
4.2.4	Falsche Verwendung der Negation mit Quantoren . . . . .	30
4.2.5	Annahme einer Closed World . . . . .	30
4.2.6	Vergessene Terme . . . . .	32

4.3	Vorgehen . . . . .	33
4.3.1	Die Quantoren betreffende Fehler . . . . .	34
4.3.2	Partielle vs. Vollständige Definition . . . . .	37
4.3.3	Vergessene Terme . . . . .	37
4.3.4	Domain- und Range-Einschränkungen . . . . .	38
4.4	Bewertung von Änderungen . . . . .	38
4.4.1	Wirkung von Änderungen auf die Konzepthierarchie . . . . .	39
4.4.2	Probleme . . . . .	40
4.5	Finden von Fehlern durch den Benutzer . . . . .	40
<b>5</b>	<b>Implementierung</b>	<b>43</b>
5.1	Funktionsweise . . . . .	43
5.2	Grafische Benutzerschnittstelle . . . . .	44
<b>6</b>	<b>Thematisch verwandte Arbeiten</b>	<b>47</b>
6.1	Nicht-Subsumtion . . . . .	47
6.2	Unerfüllbarkeit . . . . .	49
<b>7</b>	<b>Bewertung und Ausblick</b>	<b>51</b>
7.1	Vorgehensweise . . . . .	51
7.1.1	Korrektur . . . . .	51
7.1.2	Erklären . . . . .	52
7.1.3	Allgemeines . . . . .	53
7.2	Implementierung . . . . .	53

## Abbildungsverzeichnis

2.1	Syntax und Semantik von $\mathcal{SHF}$ (aus [BCM <sup>+</sup> 03]) . . . . .	6
2.2	Beispiel einer einfachen Subsumtionshierarchie von Konzepten . . . . .	9
2.3	Tableau-Erweiterungsregeln für $\mathcal{ALC}$ [BS01] . . . . .	12
2.4	Tableau-Knoten ohne Nachfolger . . . . .	13
2.5	Tableau-Knoten mit Nachfolgern mit bzw. ohne Clash . . . . .	13
2.6	Tableau-Knoten mit disjunktivem Nachbarn . . . . .	14
2.7	Beispiel für Lazy-Unfolding . . . . .	15
3.1	Beispiel Tableaughraph . . . . .	24
4.1	Vergessenes $dom(r, E)$ bzw. $range(r, F)$ . . . . .	27
4.2	Partielle statt vollständige Definition eines Konzepts $C$ . . . . .	28
4.3	Positives Vorkommen von $\forall r.(..)$ anstatt $\exists r.(..)$ . . . . .	29
4.4	Negiertes Vorkommen von $\forall r.(..)$ anstatt $\exists r.(..)$ . . . . .	29
4.5	Positives Vorkommen von $\exists r.(¬X)$ anstatt $¬\exists r.X$ . . . . .	30
4.6	Positives Vorkommen von $¬\exists r.X$ anstatt $\exists r.(¬X)$ . . . . .	31
4.7	Negiertes Vorkommen von $\exists r.(¬X)$ anstatt $¬\exists r.X$ . . . . .	31
4.8	Negiertes Vorkommen von $¬\exists r.X$ anstatt $\exists r.(¬X)$ . . . . .	32
4.9	Vergessenes <i>Person</i> in einer der Definitionen der linken Seite . . . . .	33
5.1	TBox-Editor Ansicht . . . . .	45
5.2	Visualisierung einer Erklärung . . . . .	46
5.3	Visualisierung von Änderungsvorschlägen . . . . .	46



---

# 1 Einleitung

## 1.1 Semantic Web

Das Internet ist ohne Frage eine der größten technischen Errungenschaften der vergangenen hundert Jahre. Die Menge der verfügbaren Daten und Informationen ist in den letzten zwanzig Jahren explosionsartig gestiegen. So gibt es heutzutage im Prinzip keine Ware, Dienstleistung oder Information, die nicht auch im Internet gefunden werden kann. Doch insbesondere bei der Suche im Internet stößt man selbst mit guten Suchmaschinen oft auf Probleme, da die Verarbeitung der Informationen durch den Computer bisher rein syntaktischer Art ist. Der Computer hat keine Möglichkeit, relevante von irrelevanten Ergebnissen zu unterscheiden, da hierzu ein Verständnis der Semantik nötig wäre. Diese Interpretation bleibt bisher dem menschlichen Benutzer überlassen. Nur er kann Inhalte und Zusammenhänge verstehen.

Die Semantic-Web-Initiative setzt an diesem Punkt an. Ihr Ziel ist es, das bestehende Internet um eine semantische Ebene zu erweitern, so dass die semantische maschinelle Verarbeitung des Wissens im Internet ermöglicht wird.

Bezogen auf das Beispiel der maschinellen Suche könnte eine Anwendung also unter Hinzunahme entsprechender semantischer Zusatzinformationen eines Semantic Web nicht nur nach dem Vorkommen von bestimmten Begriffen suchen, sondern auch verwandte speziellere oder allgemeine Begriffe berücksichtigen. Gleichzeitig könnten Ergebnisse weggelassen werden, die den syntaktisch gleichen Begriff zwar enthalten, jedoch in einem völlig anderen Kontext stehen.

Die Anwendungsvisionen gehen allerdings noch viel weiter. Derart hinterlegte Semantik würde es auch ermöglichen, eigenständig benötigte Informationen aus verschiedenen Quellen zusammenzutragen. Ein prominentes Szenario in diesem Zusammenhang ist die automatische Reiseplanung nach Angabe von gewünschten Reiseparametern wie Reiseziel, -ort und -dauer. Die Anwendung könnte hiermit selbstständig nach verfügbaren Hotels und Transportmöglichkeiten suchen, da sie in der Lage wäre, die benötigten Informationen zu interpretieren.

## 1.2 Motivation

Das W3C, das Gremium zur Standardisierung der im Internet verwendeten Technologien, hat OWL 2004 als Ontologiesprache definiert ([MvH04]) und hiermit den Weg für ihre Nutzung als Sprache des Semantic Web geebnet. Die formale Grundlage von OWL sind die so genannten Beschreibungslogiken.<sup>1</sup>

Im Zuge dieser Entwicklung befassen sich auch Menschen mit der Erstellung von Ontologien, denen ein detailliertes Wissen über die zu Grunde liegenden Beschreibungslogiken und die zum Einsatz kommenden Schlussfolgerungsalgorithmen fehlt. Es ist nicht zuletzt deshalb wichtig, dem Benutzer beim Umgang mit Ontologien und Reasonern Unterstützung zu bieten.

Insbesondere beim Umgang mit größeren Ontologien ist es oftmals sehr schwierig, die von einem Reasoner getätigten Schlussfolgerungen nachzuvollziehen. Dies kann insbesondere dann von Bedeutung sein, wenn es darum geht, Fehler in einer Ontologie zu finden und zu beseitigen. Selbst für Experten kann dies eine komplizierte und zeitintensive Aufgabe sein, da auch kleine Fehler sehr weitreichende und unerwartete Auswirkungen haben können. Der Benutzer benötigt also die Möglichkeit, sich nicht nur das Ergebnis einer Anfrage zeigen zu lassen, sondern auch wie dieses zustande gekommen ist. Außerdem wäre es natürlich hilfreich, auf mögliche Fehler hingewiesen zu werden.

Diese Arbeit befasst sich mit Erklärung und Korrektur von Nicht-Subsumtion. Bei der Modellierung von Ontologien kommt es häufiger vor, dass der Benutzer eine Subsumtionsbeziehung erwartet, die sich jedoch bei Überprüfung als nicht gültig herausstellt. In vielen Fällen können wenige oder nur ein kleiner Fehler in der Modellierung die Ursache für eine nicht geltende Subsumtion sein. Um dem Benutzer eine Hilfestellung anzubieten, sollte ihm eine Erklärung für die Nicht-Subsumtion präsentiert werden, d.h. nachvollziehbar dargelegt werden, warum sie nicht gilt. Auch wären Änderungsvorschläge hilfreich, die die Nicht-Subsumtion korrigieren, d.h. zur gewünschten Subsumtion führen könnten.

Der hier verfolgte Ansatz ist, die Erklärung in den Tableau-Reasoner zu integrieren. Hierbei soll allerdings eine nicht geltende Subsumtion so erklärt werden, dass dies auch für Benutzer verständlich ist, die die Funktionsweise des Tableauverfahrens nicht kennen. Dies bedeutet insbesondere, dass die Tatsache, dass es sich um einen Widerspruchsbeweis handelt, verdeckt werden muss. Durch eine strukturierte Darstellung soll dem Benutzer das Finden von Fehlern erleichtert werden. Des Weiteren wird auch versucht, konkrete Korrekturvorschläge zu unterbreiten, die das Problem lösen könnten und der Frage nachgegangen, wie Korrekturvorschläge

---

<sup>1</sup>Beschreibungslogiken und Ontologien werden in Kapitel 2 eingeführt

hinsichtlich ihrer Qualität bewertet werden könnten.

Zur Erklärung und Korrektur von Nicht-Subsumtion ist diese Arbeit eine der ersten. Im Vergleich zu den bisherigen Publikationen, die in Kapitel 6.1 vorgestellt werden, sollen hier erste praktische Ansätze diskutiert werden.

## 1.3 Überblick über die Arbeit

Diese Arbeit behandelt die tableaubasierte Erklärung und Korrektur von Nicht-Subsumtion bezüglich einer TBox.

Im Kapitel 2 werden zunächst die wichtigsten theoretischen Grundlagen dargelegt.

Kapitel 3 beschreibt, wie die im Rahmen dieser Arbeit entwickelte Erzeugung von Erklärungen für Nicht-Subsumtion unter Verwendung des Tableaubeweises funktioniert.

Kapitel 4 zeigt auf, wie im Falle einer Nicht-Subsumtion mögliche Ursachen an Hand von typischen Modellierungsfehlern gefunden werden können. Außerdem wird diskutiert, wie Änderungsvorschläge bewertet werden können.

Die im Zusammenhang dieser Arbeit erstellte prototypische Implementierung der theoretischen Ideen ist Thema von Kapitel 5.

In Kapitel 6 werden andere, thematisch ähnliche Ansätze vorgestellt.

Den Schluss bildet Kapitel 7 mit einer Bewertung der vorgestellten Ideen und der Implementierung.



---

## 2 Grundlagen

Im folgenden Kapitel werden die für diese Arbeit relevanten theoretischen Grundlagen vorgestellt. Zunächst wird in 2.1 die Beschreibungslogik als formale Grundlage von OWL eingeführt. Dann wird in 2.2 der Begriff der Ontologie erläutert. Es folgen die typischen Schlußfolgerungsdienste Erfüllbarkeit und Subsumtion in 2.3 und Erklären als Nicht-Standardinferenzdienst in 2.4. In 2.5 wird das Tableauverfahren als wichtigstes Beweisverfahren im beschreibungslogischen Kontext beschrieben und abschließend werden in 2.6 zwei aktuelle Inferenzsysteme vorgestellt.

### 2.1 Beschreibungslogik

Bei Beschreibungslogiken handelt es sich um eine logische Sprachfamilie mit klar definierter Syntax und Semantik, die eine entscheidbare Teilmenge der Prädikatenlogik darstellt. Beschreibungslogik ist die formale Grundlage für OWL, die Ontologiesprache des Semantic Web (s. auch [MvH04]).

Die wesentlichen Sprachelemente sind Konzepte und binären Relationen auch Rollen genannt. Ein Konzept steht für eine Klasse von von Dingen, eine Relation für eine Beziehung zwischen zwei Konzepten. So wäre z.B. *Vater* ein Konzept, *hat-Kind* eine Relation.

Es existieren sehr viele Beschreibungslogiken, die sich aufgrund der erlaubten Konstruktoren unterscheiden. Die Konstruktoren ermöglichen die Definition von komplexen Konzepten und Relationen aus atomaren. Durch eine größere Anzahl von erlaubten Konstrukten bekommt man einerseits eine erhöhte Ausdrucksmächtigkeit, andererseits steigt auch die Komplexität der Inferenzdienste. Die verschiedenen Beschreibungslogiken werden für gewöhnlich mit Buchstabenfolgen benannt, die für die erlaubten Konstruktoren stehen. Der in dieser Arbeit vorgestellte Ansatz bezieht sich im Wesentlichen auf die Beschreibungslogik *SHF*, deren Konstruktoren in Abbildung 2.1 zusammengefasst sind. Für genauere Informationen zu verschiedenen Beschreibungslogiken sei auf [BCM<sup>+</sup>03] verwiesen.

Konstruktor	Beschreibung	Semantik
$\top$	allgemeinstes Konzept	$\Delta^{\mathcal{I}}$
$\perp$	speziellstes Konzept	$\emptyset$
$A$	atomares Konzept	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
$r$	atomare Relation	$r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
$r \sqsubseteq s$	Rollenhierarchie	$r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$
$r \in r^+$	transitive Relation	$r^{\mathcal{I}} = (r^{\mathcal{I}})^+$
$r _C$	Domain-Einschränkung	$(r _C)^{\mathcal{I}} = \{(d, e) \mid (d, e) \in r^{\mathcal{I}} \Rightarrow d \in C\}$
$r _D$	Range-Einschränkung	$(r _D)^{\mathcal{I}} = \{(d, e) \mid (d, e) \in r^{\mathcal{I}} \Rightarrow e \in D\}$
$C \sqcap D$	Konjunktion	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
$C \sqcup D$	Disjunktion	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
$\neg C$	Negation	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
$\exists r.C$	Existenzquantifikation	$\{a \in \Delta^{\mathcal{I}} \mid \exists b : (a, b) \in r^{\mathcal{I}} \Rightarrow b \in C^{\mathcal{I}}\}$
$\forall r.C$	Allquantifikation	$\{a \in \Delta^{\mathcal{I}} \mid \forall b : (a, b) \in r^{\mathcal{I}} \Rightarrow b \in C^{\mathcal{I}}\}$
$\leq n r$	funktionale	$\{a \in \Delta^{\mathcal{I}} \mid  \{b \mid (a, b) \in r^{\mathcal{I}}\}  \leq n\}$
$\geq n r$	unqualifizierte	$\{a \in \Delta^{\mathcal{I}} \mid  \{b \mid (a, b) \in r^{\mathcal{I}}\}  \geq n\}$
$n \in \{0, 1\}$	Quantifikation	$n \in \{0, 1\}$

Abbildung 2.1: Syntax und Semantik von  $\mathcal{SHF}$  (aus [BCM<sup>+</sup>03])

## 2.2 Ontologien

Im Kontext der Wissensrepräsentation ist eine Ontologie eine Konzeptualisierung eines relevanten Weltausschnittes (auch *Domäne* genannt). Dies geschieht im beschreibungslogischen Kontext mit Hilfe der genannten Konzepte und Relationen.

Eine Ontologie kann in zwei Bestandteile unterschieden werden. Die sogenannte *TBox* (Taxonomy Box) beinhaltet die Axiome, die Konzepte und Relationen definieren. Man definiert mit Hilfe der Konstruktoren der Beschreibungslogik ein Vokabular. Die *ABox* (Assertions Box) enthält Aussagen über die konkreten Individuen einer Domäne mit Hilfe der in der *TBox* definierten Konzepte und Relationen, d.h. man bildet Instanzen.

Hierzu ein kleines Beispiel:

*TBox*:

$$\text{Mensch} \sqsubseteq \top$$

$$\text{Elternteil} \doteq \text{Mensch} \sqcap \exists \text{hat-Kind.Mensch}$$

*ABox*:

$$\text{Mensch}(\text{Peter})$$

$$\text{Mensch}(\text{Tom})$$

$$\text{hat-Kind}(\text{Peter}, \text{Tom})$$

Die *TBox* beschreibt, dass *Mensch* ein nicht näher spezifiziertes Konzept ist und ein *Elternteil* ein *Mensch* ist, der mindestens einen Füller der Relation *hat-kind* hat, der ein *Mensch* ist. Die *ABox* trifft nun die Aussage, dass *Peter* und *Tom* Instanzen des Konzeptes *Mensch* sind und dass *Peter* in der Relation *hat-Kind* zu *Tom* steht. Aus diesem expliziten Wissen ließe sich folgern, dass *Peter* eine Instanz des Konzeptes *Elternteil* ist.

## 2.3 Inferenzdienste

Eine wesentliche Eigenschaft der Beschreibungslogiken ist, dass durch ihre formale Syntax und Semantik die automatische Inferenz (Schlussfolgerung) von neuem Wissen aus bestehendem explizitem Wissen möglich ist. Die Lösung einer bestimmten Art von Inferenzproblem wird typischerweise Inferenzdienst genannt. Im Folgenden sollen die zwei typischen Inferenzdienste vorgestellt werden.

### 2.3.1 Erfüllbarkeit

Bei der Frage der Erfüllbarkeit eines Konzeptes geht es darum, ob ein Modell für das Konzept existiert, d.h. ob es widerspruchsfrei ist. Ist dies der Fall, so ist das Konzept *erfüllbar*, andernfalls sagt man das Konzept ist *unerfüllbar*. So ist beispielsweise das Konzept  $C \doteq A \sqcap \neg A$  unerfüllbar. Zwischen der im nachfolgenden Abschnitt erläuterte Subsumtion und der Erfüllbarkeit gilt folgende Äquivalenz:

$$C \sqsubseteq D \Leftrightarrow (C \sqcap \neg D) \text{ unerfüllbar}$$

Dieser Umstand wird beim Tableauverfahren ausgenutzt, ein Widerspruchsbeweisverfahren, dass in 2.5 eingeführt wird.

### 2.3.2 Subsumtion

Eine Subsumtionsbeziehung zwischen zwei Konzepten sagt aus, dass das eine (der Subsumee) spezieller ist als das andere (der Subsumer) und jede Instanz des Subsumees auch eine Instanz der Subsumers ist. Es besteht also eine Teilmengenbeziehung. So wird beispielsweise Vater von Mann subsumiert, da jeder Vater auch ein Mann ist. In diesem Fall ist also Mann der Subsumer und Vater der Subsumee.

Ein Konzept  $D$  subsumiert ein anderes  $C$ , wenn jedes Modell von  $C$  auch ein Modell von  $D$  ist, formal:

*Ein Konzept  $C$  wird bzgl. einer TBox  $\mathcal{T}$  von einem Konzept  $D$  subsumiert, wenn für jedes Modell  $\mathcal{I}$  von  $\mathcal{T}$  gilt:  $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ .*

*In diesem Fall schreibt man  $C \sqsubseteq_{\mathcal{T}} D$  oder  $\mathcal{T} \models C \sqsubseteq D$ <sup>1</sup>.*

Mit Hilfe von Subsumtionstests wird für gewöhnlich eine Hierarchie der modellierten Konzepte berechnet. Allgemeinere Konzepte stehen über von ihnen subsumierten, spezielleren Konzepten. Das allgemeinste Konzept  $\top$  steht ganz oben, da es per Definition alle anderen Konzepte subsumiert, das speziellste Konzept  $\perp$  ganz unten, da es von allen Konzepten subsumiert wird.

In Abbildung 2.2 ist ein kleines Beispiel für eine Hierarchie von Konzepten einer Familienontologie abgebildet. Die Subsumtionsbeziehung ist transitiv, z.B. wird *Tante* nicht nur von *Frau* sondern auch von *Mensch* subsumiert.

#### Nicht-Subsumtion

Eine Nicht-Subsumtion ( $C \not\sqsubseteq D$ ) liegt vor, wenn eine Subsumtion nicht gültig ist. Dies bedeutet, dass  $C \sqcap \neg D$  erfüllbar ist, also ein Modell existiert. Falls eine Subsumtion nicht gilt, ist der Subsumee auf jeden Fall erfüllbar, da eine Unerfüllbarkeit eine Äquivalenz zu  $\perp$  bedeuten würde und  $\perp$  per Definition von allen Konzepten subsumiert wird. Falls der Subsumer unerfüllbar ist, so bedeutet dies den Trivialfall einer Nicht-Subsumtion, es sei denn der Subsumee ist ebenfalls äquivalent zu  $\perp$ . Der Grund ist, dass  $\perp$  per Definition kein anderes Konzept subsumiert.

## 2.4 Erklären

Beim (auch *Explaining* genannten) Erklären handelt es sich um einen sogenannten Nicht-Standard Inferenzdienst. Der Sinn solcher Dienste ist die Unterstützung des

---

<sup>1</sup>Prädikatenlogische Semantik:  $\mathcal{T} \models \forall x : C(x) \Rightarrow D(x)$

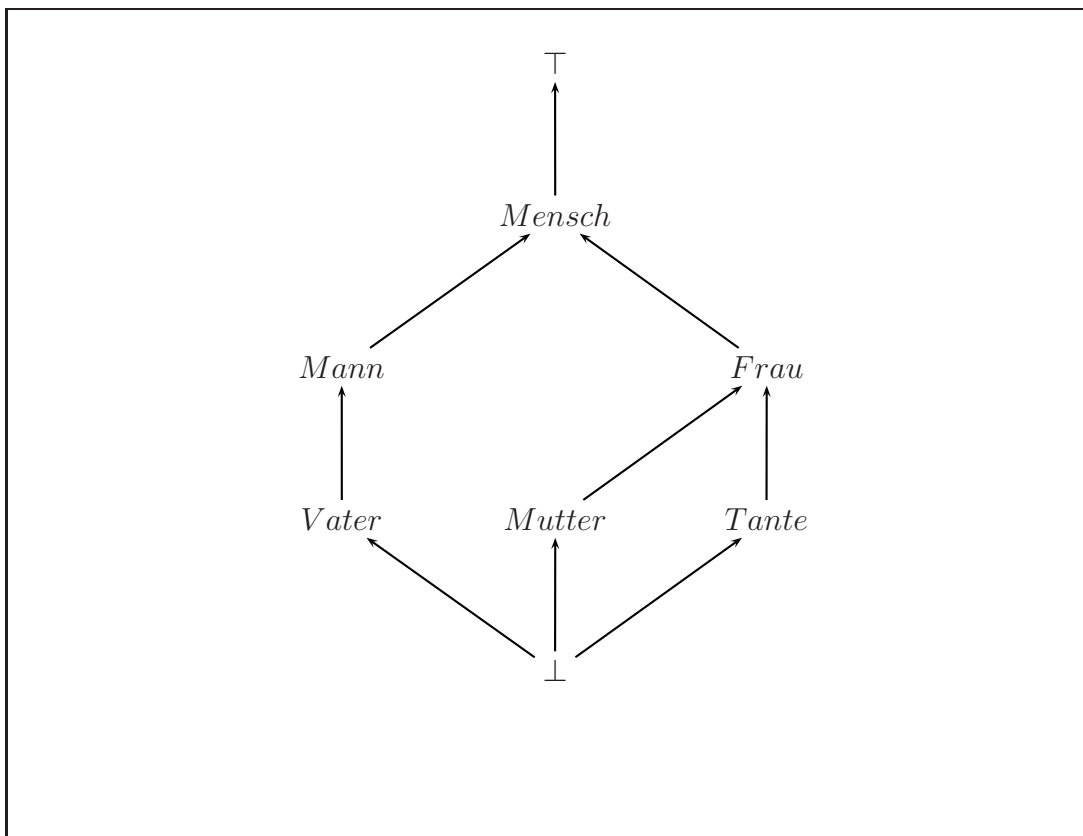


Abbildung 2.2: Beispiel einer einfachen Subsumtionshierarchie von Konzepten

Benutzers bei der Modellierung von Wissensbasen.

Die Zielsetzung des Erklärens ist, dem Benutzer die Gründe für eine Inferenz zu verdeutlichen. Da die im beschreibungslogischen Kontext verwendeten Beweisverfahren auf sehr komplizierten Algorithmen basieren, ist dies für den Benutzer eine sinnvolle Hilfe, die es ihm z.B. erleichtert, Fehler in seiner Modellierung zu finden.

## 2.5 Tableauverfahren

Das wichtigste Beweisverfahren im Kontext von Beschreibungslogiken ist das Tableauverfahren. Der Grund ist, dass es sich um ein vollständiges und korrektes Beweisverfahren handelt, das auch für sehr ausdrucksstarke Sprachen (wie OWL-DL) geeignet ist. Die meisten gängigen Reasoner verwenden daher das Tableauverfahren.

Es handelt sich um einen Widerspruchsbeweis, d.h. um die Gültigkeit einer Anfrage zu zeigen, wird diese in ein logisch äquivalentes Erfüllbarkeitsproblem umgewandelt, dessen Unerfüllbarkeit die Gültigkeit der Anfrage beweist. Für die Frage der Gültigkeit einer Subsumtion  $C \sqsubseteq D$  wird versucht, die Unerfüllbarkeit von  $C \sqcap \neg D$  zu zeigen, da folgende Äquivalenz gilt:

$$C \sqsubseteq D \Leftrightarrow C \sqcap \neg D \equiv \perp$$

Der Tableau-Algorithmus versucht ein Modell hierfür zu konstruieren und wenn dies gelingt, gilt die Subsumtion nicht, schlägt es hingegen fehl, gilt die Subsumtion.

### 2.5.1 Funktionsweise

Der Tableau-Algorithmus versucht systematisch ein Modell für eine logische Formel zu konstruieren. Intuitiv sucht er also nach einem Individuum, dass die Anfrage erfüllt. Beim Führen des Beweises wird ein sog. Tableau erzeugt, das man sich grafisch als Baum vorstellen kann, der die Zusammenhänge der Anfrage darstellt. Ein Knoten repräsentiert hierbei ein Individuum und eine Kante eine Relation zwischen zwei solchen Individuen.

Im Wurzelknoten steht zu Beginn der logische Ausdruck der Anfrage und das Tableau wird nun Schritt für Schritt aufgebaut. Für einen Tableau-Knoten, wird wie folgt vorgegangen: Zunächst werden die im Knoten stehenden Konzepte aufgefaltet, d.h. sie werden durch ihre Definition ersetzt. Für die weitere Verarbeitung wird die Negation durch logische Umformung direkt vor Konzepte und Relationsausdrücke gebracht. Diese sogenannte **Negationsnormalform** (NNF) wird durch

Anwendung folgender logischer Äquivalenzen erreicht:

$$\neg(A \sqcap B) \equiv \neg A \sqcup \neg B$$

$$\neg(A \sqcup B) \equiv \neg A \sqcap \neg B$$

$$\neg(\exists r.A) \equiv \forall r.\neg A$$

$$\neg(\forall r.A) \equiv \exists r.\neg A$$

$$\neg(\leq nr) \equiv \geq (n+1)r$$

$$\neg(\geq n.r) \equiv \leq (n-1)r$$

Anschliessend werden die sog. Tableauerweiterungsregeln (s. Abb. 2.3) auf den umgeformten Ausdruck angewendet. Für jeden im Sprachumfang vorkommenden Ausdruck existiert eine Regel, die eine entsprechende Veränderung des Tableaus beschreibt.

Durch einige der Regeln werden neue Nachfolger erzeugt, d.h. durch Ausdrücke, die die Existenz von Füllern einer Relation  $r$  fordert, entstehen neue Knoten, die über eine  $r$ -Kante mit dem Vaterknoten verbunden ist. Solche Nachfolger bzgl. einer Relation  $r$  nennt man  $r$ -Nachfolger. Regeln, die neue Knoten erzeugen nennt man generierende, solche, die neue Einschränkungen zu bestehenden Knoten hinzufügen einschränkende Regeln. So wird beispielsweise durch den Ausdruck  $\exists r.C$  die Existenz eines  $r$ -Nachfolgers der in  $C$  ist für den Knoten gefordert, in dem der Ausdruck vorkommt. Die dazu passende generierende  $\exists$ -Regel erzeugt daher einen neuen  $r$ -Nachfolger mit dem Konzept  $C$ .

Es werden immer zuerst die generierenden Regeln angewendet und anschließend die beschränkenden, da letztere auch neu erzeugte Knoten beschränken. Im nächsten Schritt wird der Algorithmus dann rekursiv auf die neu entstandenen Kindknoten angewendet.

Die  $\sqcup$ -Regel führt dazu, dass sich das Tableau im entsprechenden Knoten in mehrere Teile verzweigt. Kommt in einem Knoten eine Disjunktion vor, so spaltet er sich in mehrere disjunktive Knoten auf, die man als eigenständige Teilprobleme betrachten kann und die jeweils einen neuen sog. Pfad im Tableau bilden.

In einem Tableau-Knoten kann ein sogenannter Clash auftreten, d.h. die Aussagen eines Individuums, für das ein Knoten steht, sind widersprüchlich. Hierbei können folgende Fälle unterschieden werden:

1. im Knoten kommt sowohl  $A$  als auch  $\neg A$  für ein Konzept  $A$  vor
2. im Knoten kommt  $\geq n r$  und  $\leq m r$  mit  $m < n$  vor

Regel	Bedingung	Anwendung
$\sqcap$ -Regel	Knoten enthält $(C_1 \sqcap C_2)$ aber nicht $C_1$ und $C_2$	füge $C_1$ und $C_2$ hinzu
$\sqcup$ -Regel	Knoten enthält $(C_1 \sqcup C_2)$	verdopple den bestehenden Knoten und füge $C_1$ zum bestehenden und $C_2$ zum neuen hinzu
$\exists$ -Regel	Knoten enthält $\exists r.C$	erzeuge einen neuen $r$ -Nachfolger des Knotens und füge $C$ hinzu
$\forall$ -Regel	Knoten enthält $\forall r.C$	füge allen $s$ -Nachfolgern mit $s \sqsubseteq r$ $C$ hinzu
$\geq$ -Regel	Knoten enthält $\geq n$ $r$ aber weniger als $n$ $r$ -Nachfolger	erzeuge die fehlende Anzahl $r$ -Nachfolger
$\leq$ -Regel	Knoten enthält $\leq n$ $r$ aber mehr als $n$ $r$ -Nachfolger	verschmelze die $s$ -Nachfolger mit $s \sqsubseteq r$ zu $n$
$\forall_+ - Regel$	Knoten enthält $\forall s.C$ mit $r \sqsubseteq s$ und $r$ ist transitiv es gibt einen $r$ -Nachfolger	füge $\forall r.C$ allen $r$ -Nachfolgern hinzu

Abbildung 2.3: Tableau-Erweiterungsregeln für  $\mathcal{ALC}$  [BS01]

3. im Knoten kommt das speziellste Konzept  $\perp$  vor

Wenn in einem Knoten ein Clash vorliegt, so sagt man, der zugehörige Pfad im Tableau ist abgeschlossen. Damit ist das zugehörige Teilproblem unerfüllbar und die Knoten des Pfades werden nicht weiter expandiert. Ein Tableau heißt abgeschlossen, wenn alle seine Pfade abgeschlossen sind, in diesem Fall existiert kein Modell für die Anfrage. Falls in einem Pfad für keinen Knoten mehr eine Erweiterungsregel anwendbar ist und kein Clash im Pfad aufgetreten ist, so existiert ein Modell für die Anfrage.

## 2.5.2 Grafische Darstellung des Tableaus

Im Folgenden soll kurz erläutert werden, wie in dieser Arbeit Tableaus bzw. Teile hiervon grafisch dargestellt werden. In dieser Arbeit geht es immer um Tableaus zur Klärung einer Subsumtion. Die vom vermeintlichen Subsumee kommenden Terme stehen auf der linken Seite, die vom vermeintlichen Subsumer auf der rechten. Außerdem wird zwischen  $r$ -Nachfolgern unterschieden, die von der linken Seite, von der rechten Seite oder von beiden erzeugt bzw. erzeugt und beschränkt werden. Diese Trennung ist für die Erklärung und Korrektur notwendig.

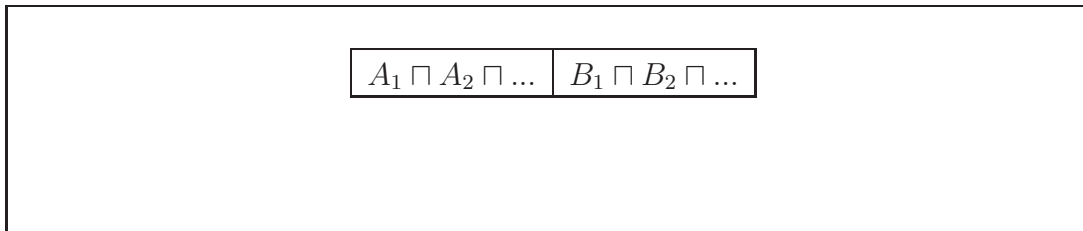


Abbildung 2.4: Tableau-Knoten ohne Nachfolger

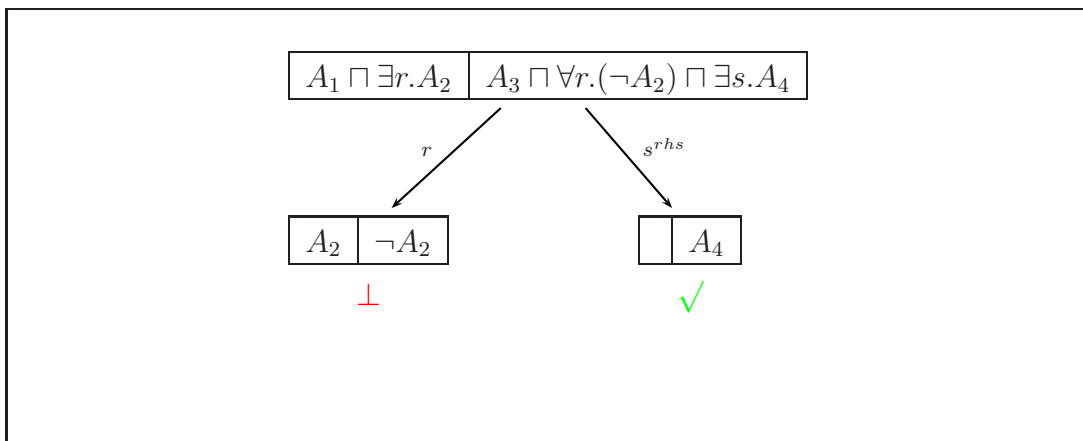


Abbildung 2.5: Tableau-Knoten mit Nachfolgern mit bzw. ohne Clash

Es werden die folgenden Notationen verwendet:

$A, B$  atomare Konzepte

$C, D$  komplexe (auffaltbare) Konzepte

$r^{lhs}$  Nachfolger der durch die linke Seite (**l**eft-**h**and **s**ide) erzeugt bzw. erzeugt und beschränkt wird

$r^{rhs}$  Nachfolger der durch rechte Seite (**r**ight-**h**and **s**ide) erzeugt bzw. erzeugt und beschränkt wird

$r$  Nachfolger der durch beide Seiten erzeugt/beschränkt wird

Es ist klar, dass ein Knoten, der ein  $r^{lhs}$ - oder ein  $r^{rhs}$ -Nachfolger ist, keine  $r$ -Nachfolger haben kann, sondern nur Nachfolger des selben Typs.

Ein Tableaunode ist als Rechteck dargestellt. Linke und rechte Seite sind durch eine senkrechte Trennlinie separiert (Abb. 2.4).

Nachfolger eines Knoten bzgl. einer Relation  $r$  werden durch gerichtete Kanten visualisiert, wobei der Name der Relation an der Kante steht. Knoten in denen ein Clash vorliegt sind durch ein  $\perp$  gekennzeichnet, abgeschlossene Knoten ohne Clash durch ein  $\checkmark$  (Abb. 2.5).

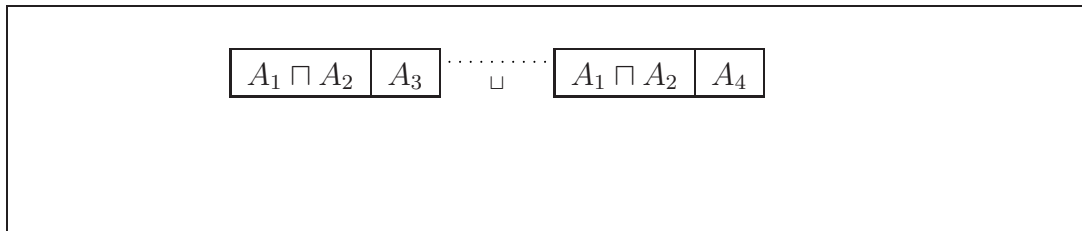


Abbildung 2.6: Tableau-Knoten mit disjunktivem Nachbarn

Wenn in einem Knoten eine Disjunktion auftaucht, führt dies dazu, dass der Knoten für jedes Disjunkt einen disjunktiven Nachbarn bekommt. Ein solcher ist der Ausgangspunkt eines neuen Pfades im Tableau. Dies wird grafisch mit einer gepunkteten ungerichteten Kante dargestellt und die Alternativen Knoten stehen nebeneinander (Abb. 2.6).

### 2.5.3 Lazy Unfolding

Die Inferenzdienste haben eine hohe Komplexität, weswegen heutige Reasoner hochoptimiert sind. Es existieren eine ganze Reihe von Optimierungen für das Tableauverfahren, die auf eine Verkürzung der Laufzeit abzielen. Im Zusammenhang dieser Arbeit wurde jedoch auf Optimierungen weitgehend verzichtet, da sie meist schwer nachvollziehbare Änderungen im Tableau vornehmen, die für das Erreichen des Ziels der Erklärung und Korrektur nicht von Vorteil sind. Daher soll hier nur auf die Optimierungsmethode des *Lazy Unfolding* eingegangen werden. Dieses Verfahren greift nicht so stark in das Tableauverfahren ein und wurde daher in dieser Arbeit umgesetzt.

#### Funktionsweise

Die Konzepte eines Tableau-Knotens werden aufgefaltet, d.h. sie werden durch ihre Definition ersetzt. Als Optimierung werden nur die Konzepte aufgefaltet, die aktuell auch aufgefaltet werden müssen. In Relationsausdrücken vorkommende Konzepte werden zunächst nicht aufgefaltet. Dies führt dazu, dass unter Umständen ein Teil des Aufwands für das Auffalten eingespart werden kann, da nicht alle Konzepte im Laufe eines Beweises aufgefaltet werden müssen. Im Beispiel in Abbildung 2.7 würde man sich den Aufwand für das Auffalten des Konzepts  $F$  sparen, falls schon im Vaterknoten ein Clash auftreten würde. Ohne Lazy Unfolding würden alle Konzepte aufgefaltet.

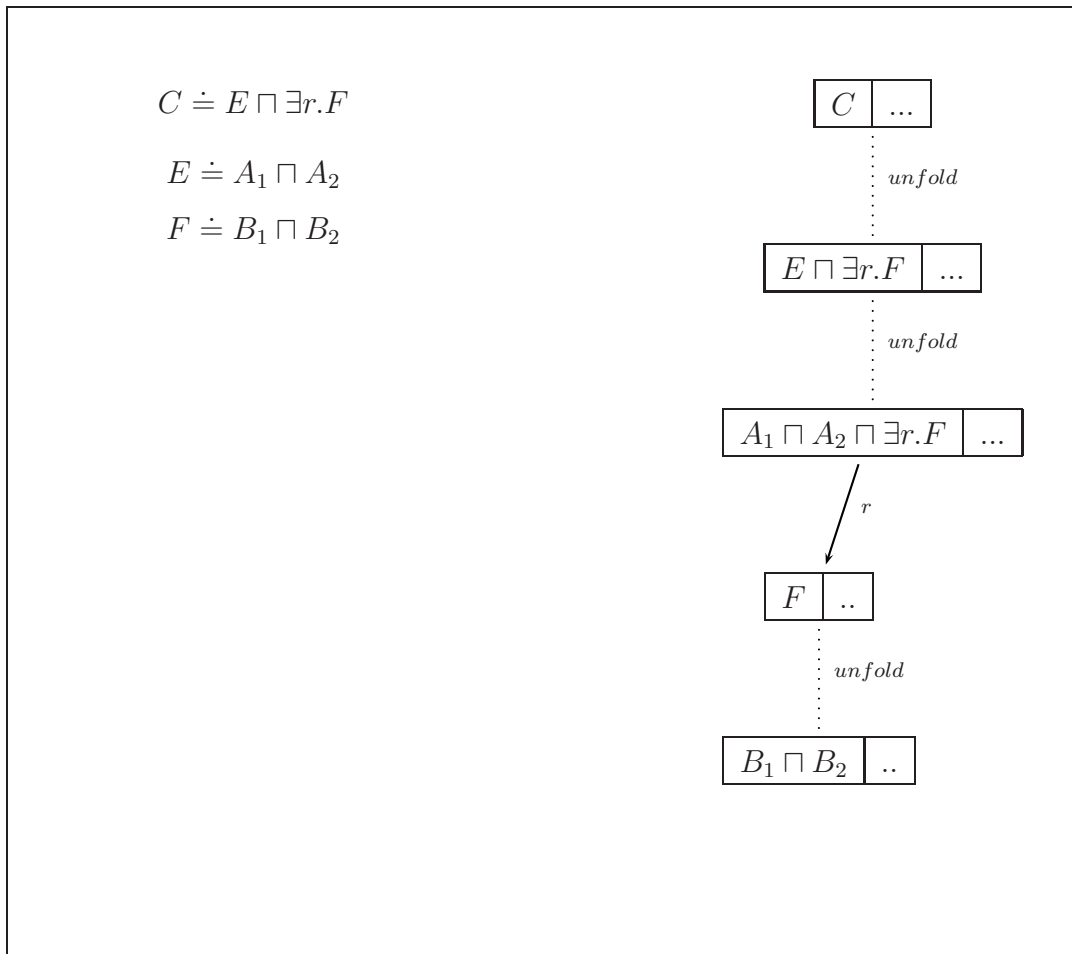


Abbildung 2.7: Beispiel für Lazy-Unfolding

## 2.6 Aktuelle Inferenzsysteme

Es existieren eine Reihe von tableaubasierten Reasonern für die beschreibungslogische Inferenz. Hier sollen nur kurz zwei Vertreter vorgestellt werden.

**RACER** ist ein bekannter Reasoner, der an der TU Hamburg-Harburg entwickelt wurde, und aus dem mittlerweile die Firma Racer Systems hervorgegangen ist. Das hochoptimierte performante System ist in LISP implementiert und wird kommerziell vertrieben. Weitere Informationen finden sich in [HM01].

**Pellet** ist ein in Java implementierter *open source*-Reasoner, der von der MIND-SWAP Gruppe an der Universität Maryland in den USA entwickelt wird und speziell auf Semantic Web Anwendungen und den Sprachumfang OWL-DL zugeschnitten ist. Weitere Informationen finden sich in [SPG<sup>+</sup>06].

---

## 3 Erklären

In diesem Kapitel soll das Vorgehen zur Generierung einer Erklärung einer Nicht-Subsumtion unter Verwendung des Tableauealküls erläutert werden. Der hier vorgestellte Ansatz baut auf [LH05] auf.

In 3.1 werden zunächst die Zielsetzung und andere allgemeine Aspekte der Erklärung von Nicht-Subsumtion beschrieben. Weiter wird in 3.2 das Vorgehen beim Generieren einer Erklärung erläutert. In 3.3 wird geklärt, in welchen Fällen eine Subsumtion bzw. Nicht-Subsumtion nicht weiter erklärt wird. In 3.4 geht es um die Frage, wie die Erklärung einfacher gemacht werden kann. Es folgen Sonderfälle der (Nicht-)Subsumtion in 3.5 und abschließend ein Beispiel in 3.6.

### 3.1 Allgemeines

Eine Nicht-Subsumtion  $C \not\subseteq D$  bedeutet das Vorliegen eines nicht abgeschlossenen Tableaus. Es existiert mindestens ein Pfad, der nicht abgeschlossen ist und der auch nicht durch Anwendung von Tableauerweiterungsregeln weiter expandiert werden kann. Somit ist es zwar möglich, dass einige Pfade des Tableaus abgeschlossen sind, d.h. Teile der Subsumtion gelten. Allerdings gibt es mindestens einen Pfad, für den dies nicht zutrifft.

Eine Erklärung für eine Nicht-Subsumtion soll dem Benutzer möglichst transparent die Ursachen der selbigen verdeutlichen. Daher ist es notwendig, eine solche Erklärung übersichtlich und klar strukturiert zu gestalten. Für das Verständnis soll ein grundlegendes Wissen über Beschreibungslogiken ausreichen, ein Kenntnis des Tableauverfahrens aber nicht vorausgesetzt werden. Das bedeutet, die Erklärung muss so gestaltet werden, dass ihre Verständlichkeit auch ohne dieses Wissen gewährleistet ist.

### 3.2 Vorgehen

Die Idee dieser Arbeit ist, eine nicht gültige Subsumtion ähnlich einer geltenden Subsumtion zu erklären. Der hier beschreibende Ansatz baut auf [LH05] und theo-

retischen Überlegungen aus [BFH00] auf.

In [LH05] wird ein Tableau-Beweiser für die Sprache  $\mathcal{AL}\mathcal{E}\mathcal{H}\mathcal{F}_{\mathcal{R}^+}$  vorgestellt, der Subsumtionsbeziehungen erklärt. Die entscheidende Feststellung ist, dass die beim Tableau-Beweis verwendete Strategie eines Widerspruchsbeweises für den Benutzer wenig intuitiv und damit auch schwer nachvollziehbar ist. Deshalb wird dieses bei der Erklärung vor dem Benutzer verborgen, und stattdessen eine Erklärung generiert, die sich auf die ursprünglichen Definitionen von Subsumer und Subsumee bezieht. Die MEX (**M**y **E**xplainer) genannte prototypische Implementierung erzeugt parallel zum Aufbau des Tableaus eine natürlichsprachliche Erklärung. Es werden mit Anwendung jeder Erweiterungsregel bzw. Auffaltung atomare Teilerklärungen generiert, die die Zerlegung des Problems beschreiben. Für jede Regel sowie für das Auffalten werden vorgefertigte generische natürlichsprachliche Formulierungen verwendet. Tritt in einem Knoten eine Disjunktion auf, so verzweigt sich die Erklärung analog zum Tableau zu mehreren Teilerklärungen.

Beim Auftreten eines Clashes wird eine Erklärung für die Gültigkeit der Subsumtion eingefügt. Um die Frage einer Subsumtion  $C \sqsubseteq D$  zu klären, wird wie bereits beschrieben ein Tableaubeweis mit der Anfrage  $C \sqcup \neg D$  geführt. Dies bedeutet, dass alle vom Subsumer ( $D$ ) stammenden Ausdrücke im Beweis negiert vorkommen. Da eine wichtige Zielsetzung der Erklärung ist, diese auch ohne Kenntnis des Tableauverfahrens nachvollziehbar zu machen, soll diese Negation vor dem Benutzer verborgen bleiben. Hierzu wird diese beim Erzeugen der Erklärungen durch erneute Negation der rechten Seite rückgängig gemacht. Für folgenden Tableau-knoten, bei dem (wie in 2.5.2 beschrieben) der Subsumee links und der Subsumer rechts stehen

$$\boxed{C \sqcap D \quad \neg E \sqcup \neg F}$$

würde z.B. daher in der Erklärung vermerkt, dass die Subsumtion  $C \sqcap D \sqsubseteq E \sqcap F$  geprüft wird [LH05].

Subsumer und Subsumee werden getrennt gespeichert, um im Laufe des Tableaubeweises ihre Unterscheidung zu ermöglichen, wie dies auch bei [LH05] der Fall ist. In normalen Reasonern ohne Erklärungskomponente würde die Anfrage aus Effizienzgründen als ein Term verarbeitet.

Beim hier vorgestellten Ansatz wird ebenfalls das Tableau nachgebildet, d.h. die Erklärung einer Nicht-Subsumtion funktioniert strukturell genauso wie die einer Subsumtion. Ein wichtiger Unterschied ist, dass die Ergebnisse von Teilsubsum-

tionen beim Aufbau des Tableaus zunächst offen bleiben müssen, da eine Nicht-Subsumtion nur bedeutet, dass mindestens eine Teilsubsumtion nicht gültig ist.

Die Erklärung bildet wie bei [LH05] das Tableau nach, d.h. sie hat eine Baumstruktur. In dessen Wurzel wird die Subsumtionsfragestellung erklärt, die als Eingabe vom Benutzer stammt. Von hier beginnend untergliedert sich die Erklärung analog zum Tableaugraphen. Für jeden Tableaunknoten wird eine Erklärungsfragment generiert. Die Erklärung wird während des Tableaubeweises erzeugt. Jede Erklärung besteht aus mehreren Fragmenten, die textuell die wichtigen Zusammenhänge beschreiben, wie in [LH05]. So wird immer zunächst beschrieben, welche Fragestellung geklärt werden soll, wie dies aufgefaltet aussieht und dann aus welchen Teilen sich das Problem zusammensetzt. Die Erklärungen werden hierfür in Haupt- und Teilerklärungen unterschieden.

Eine Haupterklärung beschreibt, was im nächsten Schritt zu klären ist. Sie nimmt das Ergebnis bereits vorweg und fasst ein oder mehrere Teilerklärungen zusammen, die das Problem weiter zerlegen, um die Verständlichkeit zu erhöhen. Die Ergebnisse werden erst beim Wiederaufstieg eingefügt, wenn sie feststehen. Jede der Teilerklärungen steht für eine disjunktive Verzweigung im Tableau, die somit auch ein eigenes Ergebnis haben kann. Diese Zerlegung orientiert sich somit am Tableaugraph, ohne jedoch die Kenntnis des Tableauperfahrens vorauszusetzen.

Am Anfang wird für den Wurzelknoten eine Haupterklärung generiert, anschließend für jede disjunktive Verzweigung des Tableaus eine Teilerklärung. Die Erklärung für die Wurzel eines Tableaus hat z.B. folgende Form:

Es wird geprüft, ob  $C \sqsubseteq D$  gilt.  
 Dies ist äquivalent zu dem aufgefalteten Problem:  $C_1 \sqcap C_2.. \sqsubseteq D_1 \sqcap D_2...$   
 Die Subsumtion gilt/lässt sich nicht nachweisen, Erklärung:

Die Teilerklärungen beschreiben die einzelnen Teilprobleme, die für das Ergebnis in der Haupterklärung verantwortlich sind. Dies bedeutet, dass in den zu einer Haupterklärung gehörigen Teilerklärungen beschrieben wird, welche Teilsubsumtionen überprüft werden müssen, um die übergeordnete Fragestellung zu klären. Sie haben die Form:

Es wird geprüft, ob  $C \sqsubseteq D_1$  gilt.  
 Dies ist der Fall/lässt sich nicht nachweisen, Erklärung:

Es ist klar, dass eine Haupterklärung nur dann eine (Teil-)Subsumtion zum Ergebnis hat, wenn alle ihre Teilerklärungen eine Subsumtion beschreiben, da jede Teilerklärung für eine Alternative (disjunkten Pfad) im Tableau steht. Ebenso erklärt im Falle einer Nicht-Subsumtion mindestens eine Teilerklärung eine Nicht-Subsumtion.

### 3.3 Basisfälle der (Nicht-)Subsumtion

Es gibt verschiedene Fälle, in denen die Subsumtion bzw. Nicht-Subsumtion nicht weiter erklärt wird. Dies ist zunächst in allen Blättern des Tableaus der Fall, das heißt in Knoten, die keine  $r$ -Nachfolger besitzen. Hier gilt entweder, dass ein Clash vorliegt und die zugehörige Teilsubsumtion gilt, oder dass kein Clash vorliegt und sich die Teilsubsumtion somit nicht nachweisen lässt.

Für den aufgefalteten Ausdruck  $A \sqsubseteq B$  reicht eine Teilerklärung der Art

...  
Die Subsumtion  $A \sqsubseteq B$  lässt sich nicht nachweisen.

Dagegen kommen im Falle eines Clashes im Knoten je nach Art des Clashes verschiedene Erklärungen in Frage, wie in [LH05] beschrieben. Im einfachsten Fall tritt ein Clash der Art  $A \sqcap \neg A$  auf. In diesem Fall kommt es darauf an, auf welcher Seite das negierte, am Clash beteiligte Konzept auftritt. Handelt es sich um die rechte Seite, lautet die Erklärung:

...  
Die Subsumtion gilt, da die linke Seite mit  $A$  die rechte enthält und lediglich zusätzliche Einschränkungen hinzufügt.

oder im Fall der linken Seite:

...  
Die Subsumtion gilt, da die linke Seite mit  $\neg A$  die rechte enthält und lediglich zusätzliche Einschränkungen hinzufügt.

Fall es sich um einen Cardinality-Clash bezüglich einer Relation  $r$  handelt, so gibt es folgende zwei Möglichkeiten:

...  
Die Subsumtion gilt, da die Anzahl der Füller von  $r$  der linken Seite der auf der rechten Seite geforderten Zahl genügt.

bzw.

...  
Die Subsumtion gilt, da die Beschränkung der Anzahl der Füller von  $r$  der linken Seite der auf der rechten Seite genügt.

Aber auch in Knoten des Tableaus, die kein Blatt sind, kann eine (Nicht-)Subsumtion auftreten, die nicht weiter erklärt wird. Dies kommt dann vor, wenn alle Nachfolger des Knotens hierfür irrelevant sind.

Ein Nachfolger bezüglich einer Relation  $r$  wird immer dann erklärt, wenn er für die Subsumtion bzw. Nicht-Subsumtion relevant ist. Dies ist der Fall, wenn er nicht nur vom Subsumer bzw. nicht nur vom Subsumee erzeugt und beschränkt wurde, d.h. nicht nur ein  $r^{lhs}$ - bzw.  $r^{rhs}$ -Nachfolger ist. Andernfalls sagt dieser Nachfolger lediglich etwas über die Erfüllbarkeit bzw. Unerfüllbarkeit des Subsumers bzw. Subsumees aus, wie z.B. in einem Knoten mit  $A \sqsubseteq \forall r. \neg B$ . In diesem Fall ist er für die Erklärung normalerweise irrelevant, außer er erklärt einen Spezialfall der (Nicht)-Subsumtion, auf die 3.5 in noch genauer eingegangen wird.

Für jeden  $r$ -Nachfolger im Tableau, der für die Erklärung relevant ist, entsteht wiederum eine Haupterklärung, da mit Erzeugung eines solchen Nachfolgers ein neues Subtableau erzeugt wird, das bzgl. der Erklärung als eigenständiges Teilproblem betrachtet werden kann. Lediglich die Entstehung des  $r$ -Nachfolgers durch die entsprechende Relation  $r$  muss gesondert erläutert werden. Eine solche Haupterklärung sieht dann folgendermaßen aus:

**Betrachte die Relation  $r$ :**

Es wird geprüft, ob  $E \sqsubseteq F$  gilt.

Dies ist äquivalent zu dem aufgefalteten Problem: ...

Die Subsumtion gilt/lässt sich nicht nachweisen, Erklärung:

Wenn ein Knoten keine für die Erklärung relevanten Nachfolger mehr besitzt, wird die (Nicht)-Subsumtion an dieser Stelle nicht weiter erklärt. Für den Tableau-knoten:

$\exists r.A$	$\neg B$
---------------	----------

existiert zwar ein  $r^{lhs}$ -Nachfolger, aber die generierte Erklärung lautet:

<p>.. Die Subsumtion <math>\exists r.A \sqsubseteq B</math> lässt sich nicht nachweisen.</p>
--

## 3.4 Vereinfachungen

Falls eine Subsumtion gilt, die aus mehreren Teilproblemen besteht, so kann es sein, dass die Zerlegung die Verständlichkeit nicht erhöht, sondern das Problem eher unübersichtlicher macht. So wäre beispielsweise für diese Subsumtion  $A \sqcap B \sqcap C \sqsubseteq A \sqcap B$  die Generierung von Teilerklärungen überflüssig. Daher werden in einem solchen Fall die Teilerklärungen entfernt und die Haupterklärung um die entsprechende Aussage erweitert:

...

Dies ist äquivalent zu dem aufgefalteten Problem:  $A \sqcap B \sqcap C \sqsubseteq A \sqcap B$

**Die Subsumtion gilt, da die linke Seite mit  $A \sqcap B$  die rechte enthält und lediglich zusätzliche Einschränkungen hinzufügt.**

Dies funktioniert wie in [LH05] beschrieben.

Eine weitere Vereinfachung in diesem Sinne ist die Sortierung der Teilerklärungen nach geltenden und nicht geltenden Subsumtionsbeziehungen. So werden dem Benutzer unter jeder Haupterklärung zunächst die Teile der Subsumtion präsentiert, die erfüllt sind und anschließend die, die nicht erfüllt sind.

## 3.5 Sonderfälle

Es gibt einige Sonderfälle der (Nicht-)Subsumtion, die hier kurz erläutert werden sollen. Sie treten immer dann ein, wenn eine Seite äquivalent zum allgemeinsten Konzept  $\top$  bzw. zum speziellsten Konzept  $\perp$  ist.

Ist die linke Seite äquivalent zu  $\perp$ , so gilt die Subsumtion per Definition, da  $\perp$  das speziellste Konzept ist. Ebenso, falls die rechte Seite äquivalent zu  $\perp$  und damit der Subsumer äquivalent zu  $\top$  ist, da  $\top$  das allgemeinste Konzept ist und per Definition alles subsumiert [LH05].

In diesen Fällen wird eine Erklärung dieser Art generiert:

Die linke Seite ist äquivalent zu  $\perp$   
und wird per Definition von allem subsumiert.

bzw.

Die rechte Seite ist äquivalent zu  $\top$   
und subsumiert per Definition alles.

Ist die linke Seite äquivalent zu  $\top$ , ist dies normalerweise ein Trivialfall einer Nicht-Subsumtion, es sei denn, die rechte Seite ist äquivalent zu  $\perp$ , d.h. der Subsumer ebenfalls äquivalent zu  $\top$ . Analoges gilt, falls die rechte Seite äquivalent zu  $\top$  ist und damit der Subsumer äquivalent zu  $\perp$ .

In diesen Fällen wird eine Erklärung dieser Art generiert:

Die rechte Seite ist äquivalent zu  $\perp$   
und subsumiert per Definition nichts (außer  $\perp$ ).

bzw.

Die linke Seite ist äquivalent zu  $\top$   
und wird per Definition von nichts subsumiert (außer von  $\top$ ).

Die genannten Fälle können bereits in der Wurzel auftreten, aber auch an einer anderen Stelle weiter unten im Tableau. Um eine passende Erklärung einzufügen, geschieht dies erst beim rekursiven Wiederaufstieg beim Erzeugen des Tableaus, wenn das Ergebnis bereits feststeht.

## 3.6 Beispiel

Für die folgende TBox (Tableau in Abb. 3.1)

$$A \doteq A1 \sqcap A2$$

$$B \doteq B1 \sqcap B2$$

$$C \doteq A2 \sqcap \exists r.B$$

$$D \doteq A \sqcap \exists r.B1$$

würde folgende Erklärung generiert:

- Es wird geprüft ob  $C \sqsubseteq D$  gilt
- dies ist äquivalent zum aufgefalteten Problem:  $A2 \sqcap \exists r.B \sqsubseteq A1 \sqcap A2 \sqcap \exists r.B1$
- die Subsumtion ist nicht gültig, da mindestens eine der folgenden Teilsubsumtionen nicht nachweisbar ist:
  - Es wird geprüft ob  $A2 \sqcap \exists r.B \sqsubseteq \exists r.B1$  gilt:

### 3 Erklären

- \* für die Relation  $r$ :
- \* es wird geprüft ob  $B \sqsubseteq B1$  gilt:
- \* dies ist äquivalent zum aufgefalteten Problem:  $B1 \sqcap B2 \sqsubseteq B1$  die Subsumtion gilt
- Es wird geprüft ob  $A2 \sqcap \exists r.B \sqsubseteq A2$  gilt:  
die Subsumtion gilt
- Es wird geprüft ob  $A2 \sqcap \exists r.B \sqsubseteq A1$  gilt:  
folgende Subsumtion lässt sich **nicht** nachweisen:  $A2 \sqcap \exists r.B \sqsubseteq A1$

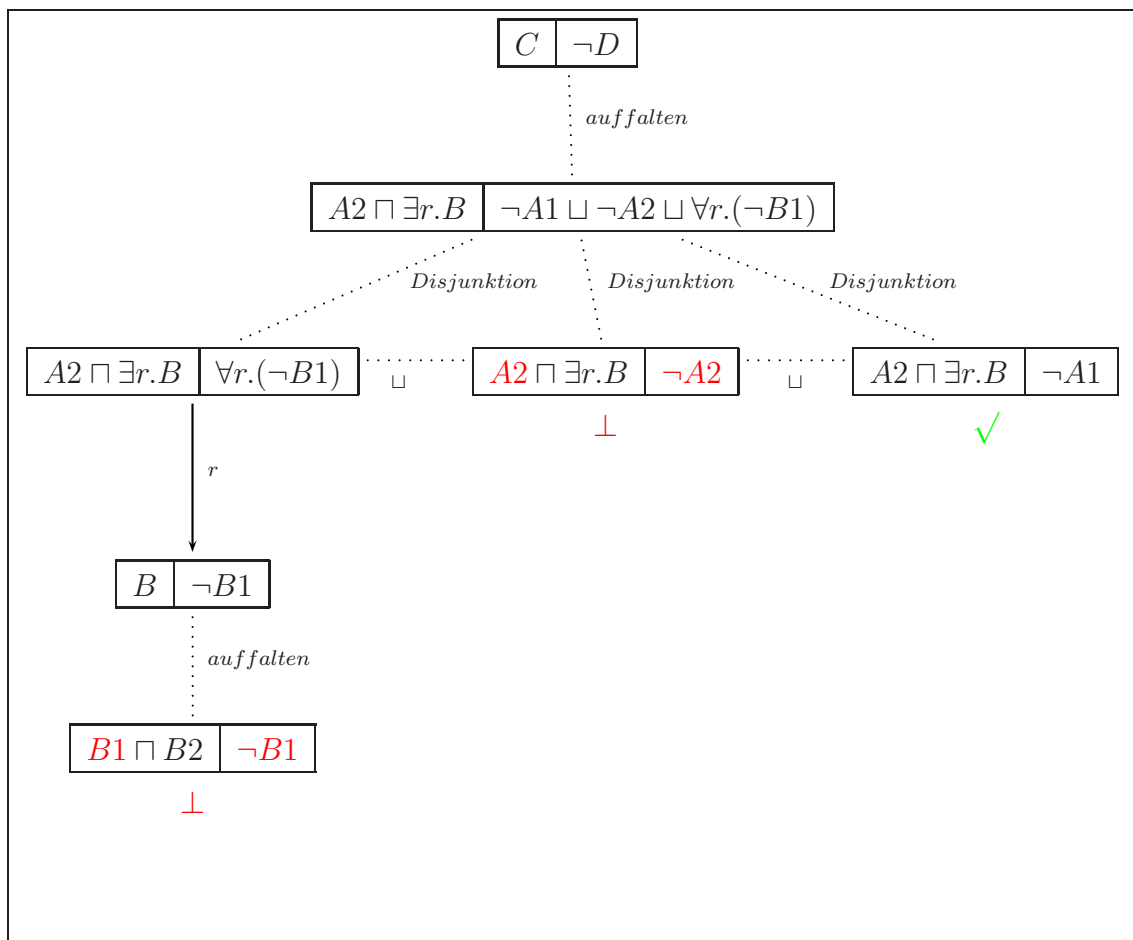


Abbildung 3.1: Beispiel Tableaughraph

---

## 4 Korrektur

Im Falle einer unerwarteten Nicht-Subsumtion stellt sich für den Benutzer die Frage, warum die Subsumtion nicht gilt. Erwartet der Benutzer eigentlich eine Subsumtion, so ist es möglich, dass ein oder mehrere Modellierungsfehler die Ursache sind. Im Folgenden sollen Vorschläge vorgestellt werden, wie automatisiert nach Änderungen gesucht werden kann, die zu einer Subsumtion führen könnten. Das Verfahren konzentriert sich auf die Suche nach typischen Modellierungsfehlern und -problemen von Benutzern, die in einer Studie erfasst worden sind. Der hier vorgestellte Ansatz beruht auf einer tableaubasierten Fehlersuche, d.h. beim Führen des Tableaubeweises zur Klärung der Subsumtion wird versucht, potentielle Fehler zu entdecken.

In 4.1 wird zunächst auf die spezielle Problematik bei der Suche nach Ursachen einer Nicht-Subsumtion eingegangen. Anschließend wird in 4.2 beschrieben, welche Fehler bei der Modellierung von Ontologien häufig gemacht werden. Es wird dargelegt, wie diese Fehler eine Nicht-Subsumtion bewirken können. In 4.3 wird darauf eingegangen, wie mit diesem Wissen eine automatische Suche nach Fehlern aussehen kann. Um die Bewertung von gefundenen Änderungen geht es in 4.4. Zum Abschluss folgen in 4.5 Ausführungen über die Bedeutung des Benutzers bei der Fehlersuche.

### 4.1 Grundlegendes

Das Finden der Ursachen einer Nicht-Subsumtion in einer Ontologie ist keine einfache Aufgabe. In [Kal06] wird beschrieben, warum die Ursache einer nicht gültigen Subsumtionsbeziehung deutlich schwieriger zu finden ist als beispielsweise die Ursachen eines unerfüllbaren Konzeptes. Da im letzteren Fall ein abgeschlossenes Tableau mit mindestens einem Clash vorliegt, findet man mit den Ursachen der Clashes auch die Ursachen der Unerfüllbarkeit. So kann man die am Clash beteiligten Ausdrücke sowie die Definitionen, in denen diese ihren Ursprung haben, angeben. Eine Nicht-Subsumtion liegt vor, wenn das Tableau nicht abgeschlossen ist. Damit hat man auch keinen direkten Ansatz zur Fehlersuche. Es gibt unendlich viele Möglichkeiten, diese zu korrigieren, d.h. eine oder mehrere Definitionen der Ontologie so abzuändern, dass das Tableau abgeschlossen ist. Eine so zielgerichtete

Vorgehensweise, wie z.B. im Fall unerfüllbarer Konzepte, ist nicht möglich.

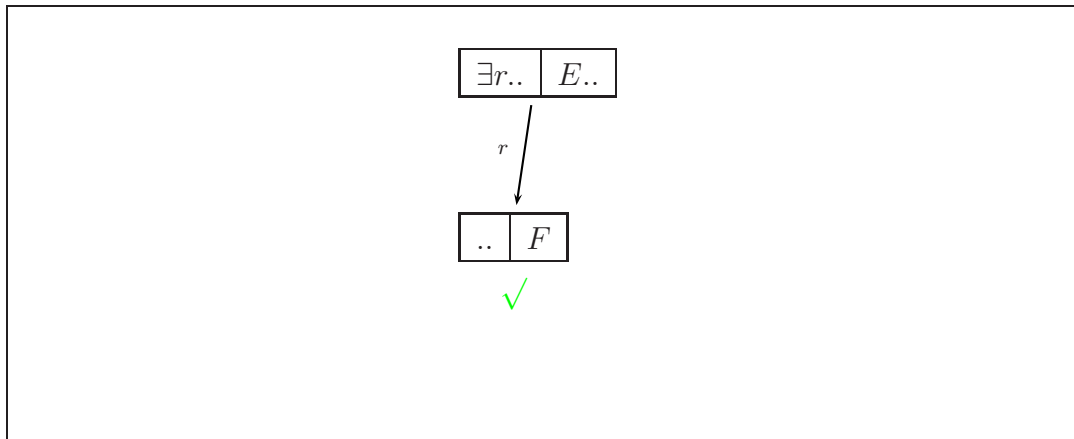
Insbesondere wenn mehrere Fehler für eine Nicht-Subsumtion verantwortlich sind, ist deren Entdeckung problematisch. Schließlich führt dies dazu, dass die richtige Kombination mehrerer Änderungen gefunden werden muss, was eine erhöhte Komplexität zur Folge hat. Grundsätzlich ist es immer möglich, dass ein oder mehrere Fehler die Nicht-Subsumtion verursachen, die in ihrem Zusammenwirken zu komplex sind, als dass sie entdeckt und dem Benutzer vorgeschlagen werden könnten.

Bei der Suche nach Fehlern muss man alle am Beweis beteiligten Definitionen berücksichtigen. Dies schließt auch die im Subsumee bzw. Subsumer direkt oder indirekt vorkommenden Konzepte und Relationen mit ein. Deren Definitionen können ebenso relevante Fehler enthalten wie die Definitionen von Subsumer und Subsumee selbst.

## 4.2 Typische Fehler bei der Ontologie-Modellierung

In [RDH<sup>+</sup>04] werden die typischen Fehler bei der Erstellung von Ontologien untersucht. Hierbei greifen die Autoren auf Erfahrungen zurück, die sie bei der Lehre von OWL gemacht haben. Es hat sich gezeigt, dass es bestimmte Fehler gibt, die insbesondere von unerfahrenen Benutzer sehr häufig gemacht werden. Auf Grund der genannten Schwierigkeiten bei der Suche nach möglichen Ursachen von Nicht-Subsumtion wird hier der Ansatz verfolgt, beim Auftreten einer Nicht-Subsumtion nach diesen typischen Fehlern zu suchen.

Diese Fehler sollen im Folgenden zunächst kurz erläutert werden. Dabei wird insbesondere darauf eingegangen, wie sie eine Subsumtion verhindern können. Die folgenden einfachen Fälle berücksichtigen nicht, dass eine Änderung auch sehr viel indirekter und weniger offensichtlich zum Clash führen kann, nämlich in nachfolgenden Schritten des Tableau-Algorithmus. Es soll lediglich schematisch dargelegt werden, wie durch die genannten Fehler ein Clash verhindert werden kann. Bei der Wirkung eines Fehlers muss immer unterschieden werden, ob der entsprechende Term negiert oder positiv im Tableau vorkommt. Stammt der Term aus der Definition des Subsumers oder einer Definition, die von diesem verwendet wird, so findet er sich im Tableau auf der rechten Seite und ist negiert. Andernfalls kommt er auf der linken Seite positiv vor. Je nach Sprachumfang kann das negierte bzw. positive Vorkommen natürlich auch von der Seite unabhängig sein, falls komplexe Negation erlaubt ist.

Abbildung 4.1: Vergessenes  $dom(r, E)$  bzw.  $range(r, F)$ 

### 4.2.1 Domain- und Range-Einschränkungen

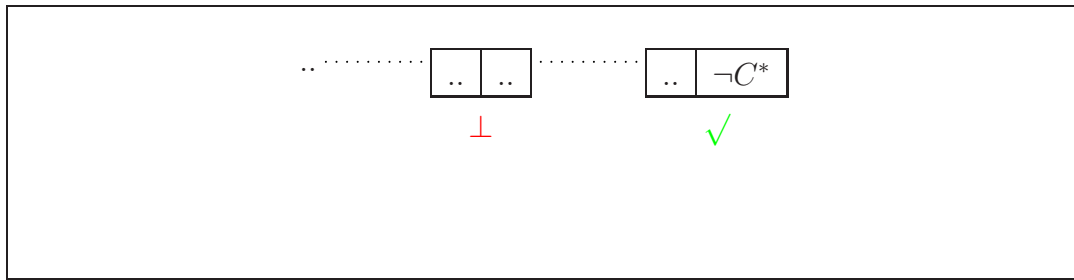
Domain- und Range-Einschränkungen fügen implizite Einschränkungen hinzu, die bei der Modellierung einer Ontologie oftmals an anderer Stelle nicht mehr bedacht werden. Wenn z.B. für *hat-Kind* als Domain *Mensch* definiert ist, wird *Mensch* jedem Knoten hinzugefügt, der einen *hat-Kind*-Nachfolger besitzt [RDH<sup>+</sup>04].

Jedoch hat eine unbedachte Einschränkung keine Nicht-Subsumtion zur Folge, da für jeden Füller einer Relation  $r$  auf der rechten Seite sowieso ein entsprechender Füller einer Subrelation von  $r$  auf der linken vorhanden sein muss, der dann auch dieselben oder speziellere Einschränkungen hat.

Anders sieht es bei vergessenen Einschränkungen aus. Wenn auf der linken Seite ein Füller einer Relation  $r$  gefordert wird und die Subsumtion nicht gilt, so kann beispielsweise eine fehlende Domain-Einschränkung die Ursache sein. In Abbildung 4.1 ist schematisch dargestellt, wie ein Teil eines Tableaus durch eine vergessene Domain- bzw. Range-Einschränkung einen nicht abgeschlossenen Pfad verursachen kann.

### 4.2.2 Partielle statt vollständige Definition

Der Unterschied zwischen einer partiellen und einer vollständigen Definition ist dem unerfahrenen Benutzer ebenfalls oft nicht klar. Bei einer partiellen Definition werden nur die notwendigen, nicht aber die hinreichenden Bedingungen angegeben. Dies hat zur Folge, dass nichts unter einem so definierten Konzept subsumiert wird, außer natürlich per Definition  $\perp$  und solche Konzepte, die das Konzept selbst oder

Abbildung 4.2: Partielle statt vollständige Definition eines Konzepts  $C$ 

dessen Unterkonzepte direkt spezialisieren [RDH<sup>+</sup>04].

Eine solche Definition wird üblicherweise in eine neue vollständige Definition umgewandelt, um eine Auffaltung während des Tableaubeweises zu ermöglichen. Das partiell definierte Beispiel  $C \sqsubseteq A \sqcap B$  statt  $C \doteq A \sqcap B$  würde umgewandelt in  $C \doteq (A \sqcap B) \sqcap C^*$  und zusätzlich ein neues Konzept  $C^*$  partiell so definiert  $C^* \sqsubseteq \top$ .

Taucht also auf der rechten Seite ein solches partiell definiertes Konzept auf, so kann hierdurch eine Subsumtion verhindert werden. Die Konjunktion auf der rechten Seite führt zu mehreren disjunkten Pfaden im Tableau, von denen einer das neue Konzept  $C^*$  umfasst (s. Abb. 4.2). Falls die partielle Definition ein Fehler des Benutzers ist, hätte genau diese Alternative einen nicht geschlossenen Pfad im Tableau zur Folge.

### 4.2.3 Universelle statt existentielle Qualifikation

Laut [RDH<sup>+</sup>04] kommt es häufiger vor, dass in der Definition eines Konzeptes eine Allquantifikation anstatt einer existentiellen Quantifikation verwendet wird. Der Grund ist, dass Benutzer oft fälschlicherweise davon ausgehen, dass die Verwendung einer Allquantifikation gleichzeitig bedeutet, dass auch ein Füller der Relation existiert.

Wenn in einer Definition fälschlicherweise eine universelle ( $\forall r$ ) statt einer existentiellen ( $\exists r$ ) Qualifikation steht, kann dies eine Nicht-Subsumtion verursachen. Je nach Art des Vorkommens (positiv oder negiert) bewirkt ein solcher Fehler, dass im Tableau ein Allquantor an Stelle eines Existenzquantors vorkommt oder umgekehrt. Die Abbildungen 4.3 und 4.4 zeigen schematisch die Auswirkungen auf das Tableau.

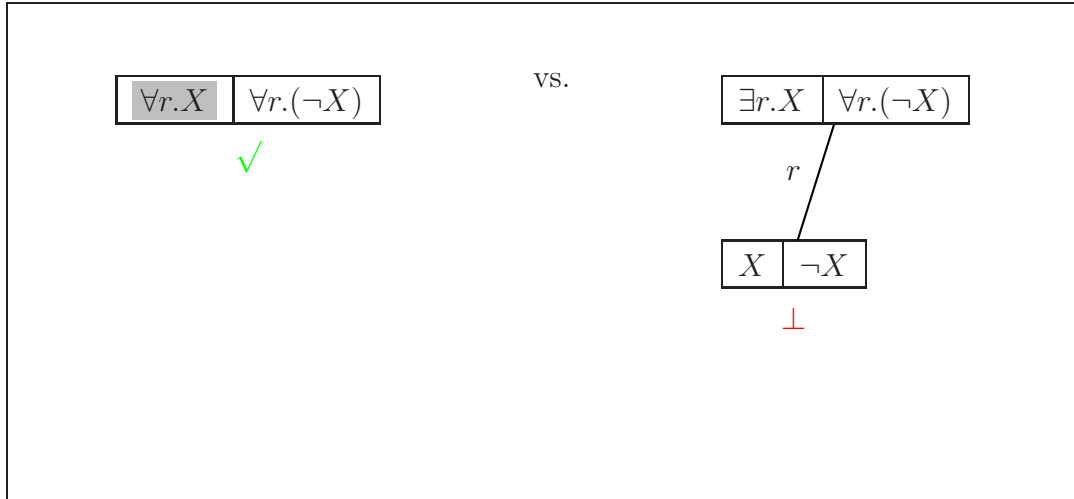


Abbildung 4.3: Positives Vorkommen von  $\forall r.(..)$  anstatt  $\exists r.(..)$

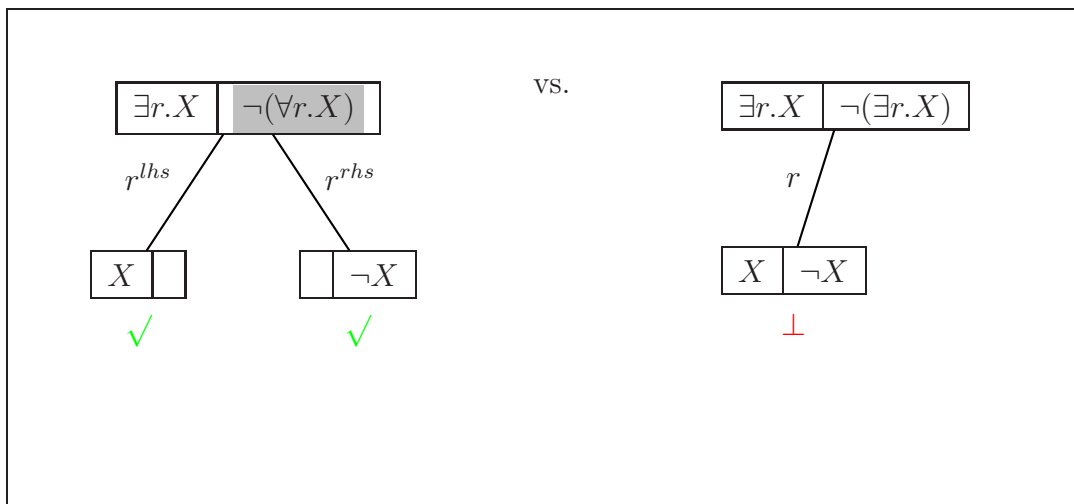
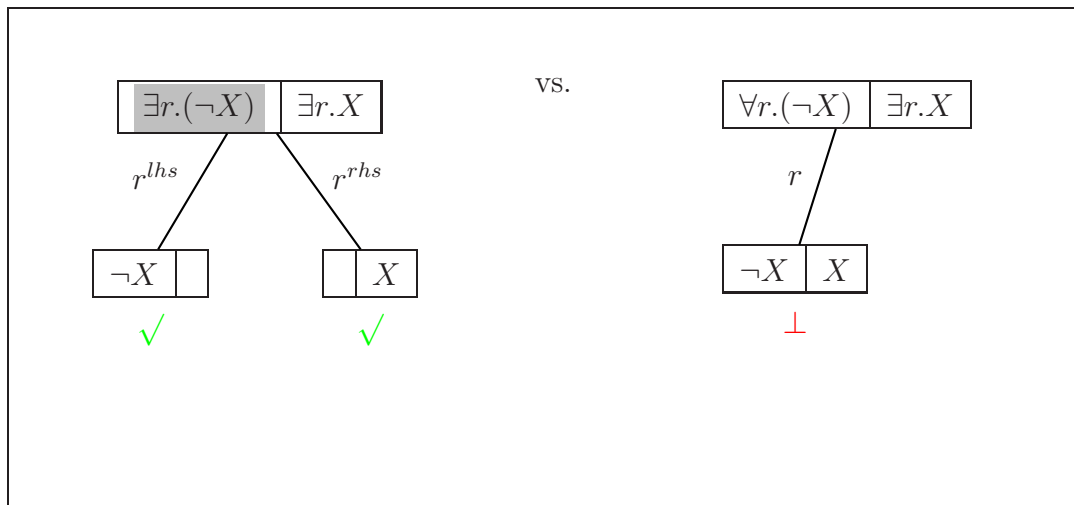


Abbildung 4.4: Negiertes Vorkommen von  $\forall r.(..)$  anstatt  $\exists r.(..)$

Abbildung 4.5: Positives Vorkommen von  $\exists r.(\neg X)$  anstatt  $\neg\exists r.X$ 

#### 4.2.4 Falsche Verwendung der Negation mit Quantoren

Auch ist es ein gängiger Fehler, die Negation im Zusammenhang mit den Quantoren falsch zu verwenden [RDH<sup>+</sup>04].

Der Unterschied von  $\exists r.(\neg Vater)$  und  $\neg\exists r.Vater$  ist, dass man im ersten Fall einen Füller für die Relation  $r$  der kein  $Vater$  ist fordert, im zweiten jedoch, dass es keinen Füller der Relation  $r$  gibt, der  $Vater$  ist. Wenn  $\exists r.(\neg X)$  und  $\neg\exists r.X$  vertauscht werden, gibt es vier Möglichkeiten, da ein Vertauschen in zwei Richtungen und wiederum ein positives sowie ein negiertes Vorkommen des Ausdrucks in Frage kommen. Auch diese Fehler bewirken eine Änderung von Existenzquantoren in Allquantoren bzw. umgekehrt.

In den Abbildungen 4.5 bis 4.8 sind die Auswirkungen solcher Fehler schematisch dargestellt.

#### 4.2.5 Annahme einer Closed World

Vielen Benutzern ist auch der Unterschied zwischen Closed- und Open-World-Reasoning nicht bewusst. Sie gehen fälschlicherweise davon aus, dass etwas automatisch nicht gilt, wenn man es nicht angibt (Closed-World), wie es beispielsweise bei der Abfrage von Daten in Datenbanken üblich ist. Im Kontext von OWL ist jedoch Open-World-Reasoning üblich, d.h. was nicht angegeben wird kann trotzdem gelten [RDH<sup>+</sup>04].

Dies kann sehr verschiedene Folgen haben. Der Benutzer gibt etwas an und geht

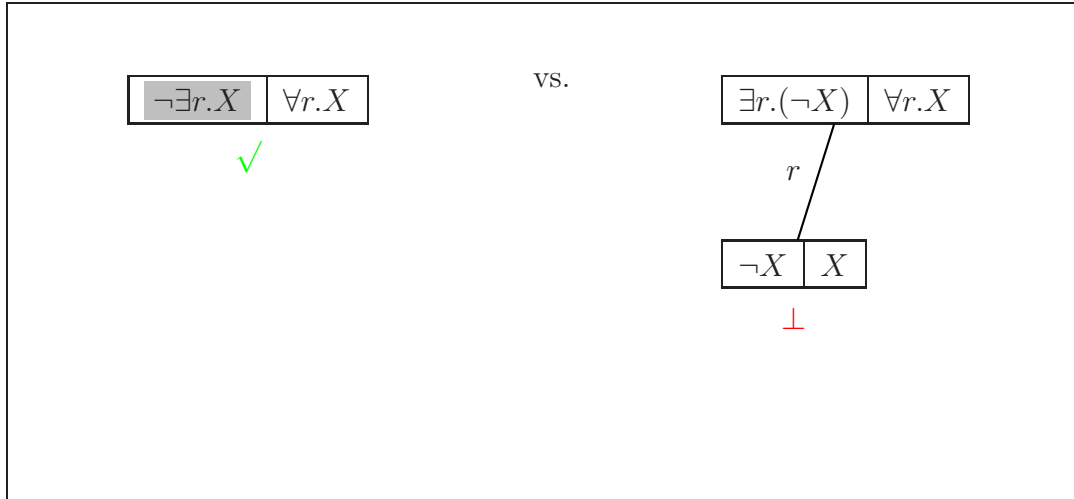


Abbildung 4.6: Positives Vorkommen von  $\neg\exists r.X$  anstatt  $\exists r.(¬X)$

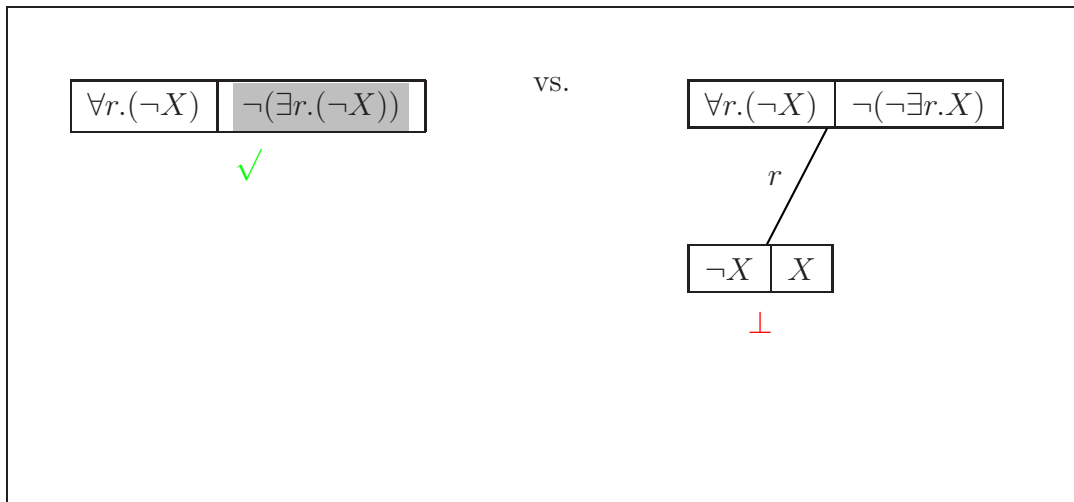
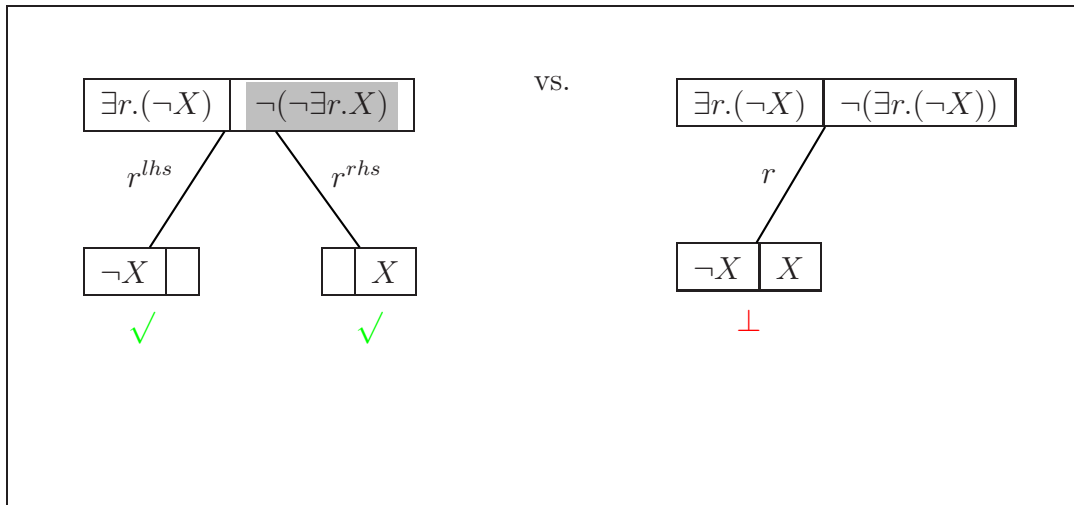


Abbildung 4.7: Negiertes Vorkommen von  $\exists r.(¬X)$  anstatt  $\neg\exists r.X$

Abbildung 4.8: Negiertes Vorkommen von  $\neg\exists r.X$  anstatt  $\exists r.(\neg X)$ 

davon aus, dass alles andere nicht gilt. Das bedeutet, dass zusätzlich benötigte Einschränkungen weggelassen werden. So kann es sein, dass der Benutzer die Existenz von Füllern einer Relation fordert und davon ausgeht, dass keine anderen Füller existieren können. Entsprechend kann es passieren, dass die abschließende Allquantifikation vom Benutzer weggelassen wird, da er sie für nicht nötig hält.

Hierzu ein Beispiel:

$$\text{MutterMitSohn} \doteq \text{Frau} \sqcap \exists \text{hat-Kind.Mann}$$

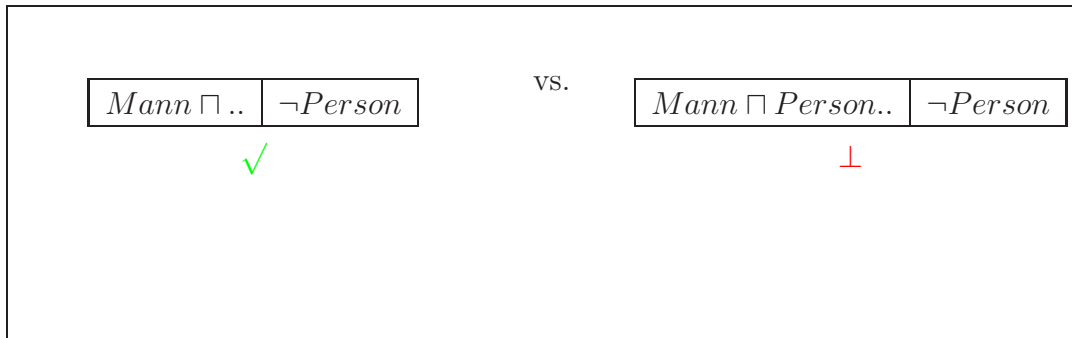
$$\text{ElternteilOhneTochter} \doteq \exists \text{hat-Kind.}\top \sqcap \forall \text{hat-Kind.Mann}$$

Die Definition von *MutterMitSohn* sagt nichts darüber aus, dass nicht auch ein Füller der Relation *hat-Kind* existieren darf, der eine *Frau* ist. Unter Umständen wird hier die Erwartung enttäuscht, dass *ElternteilOhneTochter* *MutterMitSohn* subsumiert.

Dieser Fall wird jedoch von dem im folgenden Abschnitt beschriebenen Vorgehen abgedeckt. Diese Arbeit beschäftigt sich daher im Weiteren nicht speziell mit der Problematik einer fälschlichen Annahme einer Closed World.

#### 4.2.6 Vergessene Terme

Es kann natürlich auch immer passieren, dass der Benutzer einfach vergisst, etwas einer Definition hinzuzufügen.

Abbildung 4.9: Vergessenes *Person* in einer der Definitionen der linken Seite

Im folgenden Beispiel fehlt für die Gültigkeit der Subsumtion  $Vater \sqsubseteq Elternteil$  die Eigenschaft *Person* auf der linken Seite.

$$Person \sqsubseteq Mensch$$

$$Mann \sqsubseteq Mensch$$

$$Vater \doteq Mann \sqcap \exists hat-K. \top$$

$$Elternteil \doteq Person \sqcap \exists hat-K. \top$$

In diesem Beispiel ist es möglich, dass in einer der Definitionen *Mann* oder *Vater* das Hinzufügen von *Person* vergessen wurde. Abbildung 4.9 zeigt einen Tableau-Knoten, in dem dies der Fall ist. Eigentlich liegt hier keine spezielle Auswirkung auf das Tableau vor, d.h. jeder Knoten, der auf einem nicht geschlossenen Pfad im Tableau liegt, sieht so aus.

## 4.3 Vorgehen

Im Falle einer nicht gültigen Subsumtion wird nach Symptomen der genannten typischen Fehler gesucht. Die Entscheidung, was ein Fehler ist und was nicht muss letztendlich immer der Benutzer treffen. Die Idee ist daher potentielle Fehler aufzudecken und dem Benutzer dann die möglichen Änderungen zur Korrektur der Nicht-Subsumtion vorzuschlagen. Die Suche nach potentiellen Fehlern ist in den Tableau-Reasoner integriert. Beim Führen eines Tableau-Beweises muss für jeden Term in jedem Knoten nachvollziehbar sein, aus welcher Definition der Term stammt. Es kann vorkommen, dass ein Term aus mehreren Definitionen aufgefaltet wird und daher eine Änderung auch in verschiedenen Definitionen vorgenommen werden kann. Dies führt dann zu mehreren Änderungsvorschlägen.

Eine Nicht-Subsumtion bedeutet ein nicht abgeschlossenes Tableau. Die nicht abgeschlossenen Pfade bilden den Ausgangspunkt für die Suche nach möglichen Ursachen. In Knoten eines nicht abgeschlossenen Pfades im Tableau wird beim Wiederaufstieg nach Symptomen der beschriebenen Fehler gesucht. Die Überprüfung beim Wiederaufstieg vermeidet unnötige Überprüfungen, da in dieser Phase sicher ist, dass weiter unten im Tableau kein Clash aufgetreten ist. Dabei wird in mehreren Phasen nach den verschiedenen Typen von Fehlern gesucht.

Je nach Situation können auch mehrere Fehler für eine Nicht-Subsumtion verantwortlich sein. Das hier beschreibende Verfahren sucht aber nicht nach Kombinationen von Änderungen, da dies den Aufwand stark erhöhen würde. Es werden auch Änderungen vorgeschlagen, die für sich genommen die Nicht-Subsumtion nicht beheben, was sich aber auf ihre Bewertung auswirkt (s. 4.4).

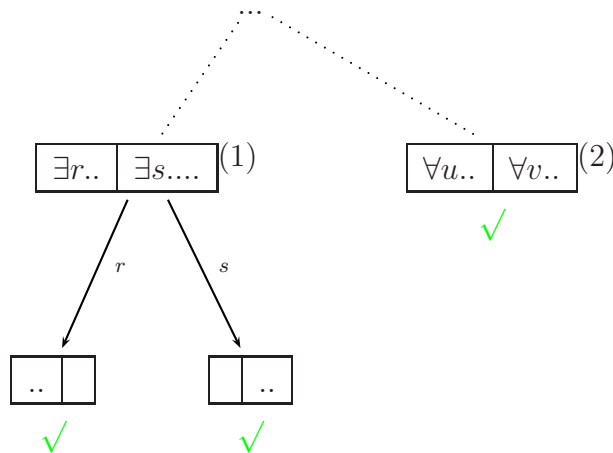
### 4.3.1 Die Quantoren betreffende Fehler

Die Symptome der die Quantoren betreffenden Fehler lassen sich in zwei Muster aufteilen. Dies gilt sowohl für die falsche Verwendung der Negation als auch für die Verwendung der Allquantifikation an Stelle der existentiellen Quantifikation.

Das erste Muster sind  $r$ -Nachfolger über eine gleiche Relation (bzw. Subrelation, s.u.). Das zweite Muster sind Allquantoren ohne einen entsprechenden Nachfolger auf der anderen Seite, auf den sie wirken.

Es wird immer nach gemeinsamen Relationen beider Seiten gesucht, da zum Entstehen einer nicht trivialen Subsumtion schließlich auch gemeinsame Nachfolger entstehen müssen.

Schematisch sieht das so aus (es gelte  $r \sqsubseteq s$  und  $u \sqsubseteq v$ ):



Hier sieht man im Knoten (1) das erste und im Knoten (2) das zweite Muster. Wird ein solches Muster in einem Knoten gefunden, so liegt ein potentieller Fehler vor.

Die Suche nach dem ersten Muster läuft wie folgt: Für jeden Ausdruck mit Existenzquantor auf der linken Seite wird nach allen passenden auf der rechten Seite gesucht. Das bedeutet nach Ausdrücken mit Existenzquantor bzgl. der gleichen Relation oder einer Superrelation auf der rechten Seite. Die Suche nach dem zweiten Muster funktioniert dann so: Für jeden Ausdruck mit Allquantor auf der linken Seite wird nach allen passenden auf der rechten Seite gesucht. Das bedeutet eine Suche nach Ausdrücken mit Allquantor wiederum bezüglich der gleichen Relation oder einer Superrelation auf der rechten Seite.

Beide Muster können durch einen potentiellen Fehler auf beiden Seiten verursacht worden sein. Im nächsten Schritt wird mit Hilfe der gespeicherten Herkunft aller gefundenen Ausdrücke in den betroffenen Definitionen überprüft, ob dort ein potentieller Fehler vorkommt. Dies ist nötig, da durch Umformungen im Laufe des Tableau-Beweises der in der Definition vorkommende Term u.U. negiert worden ist. Kommt in einer Definition ein potentieller Quantoren-Fehler vor, so wird temporär der potentiell richtige Term in die Definition eingefügt und die Auswirkung berechnet.

Für alle gefundenen Änderungsvorschläge wird ein entsprechender Text generiert, der dem Benutzer als Empfehlung präsentiert wird z.B.:

Ersetzen Sie  $\forall r..$  in der Definition von ... durch  $\exists r...$

In folgendem Beispiel ist in  $Elternteil_F$  statt einem Existenzquantor ein Allquantor verwendet worden:

$$hat\text{-Sohn} \sqsubseteq hat\text{-Kind}$$

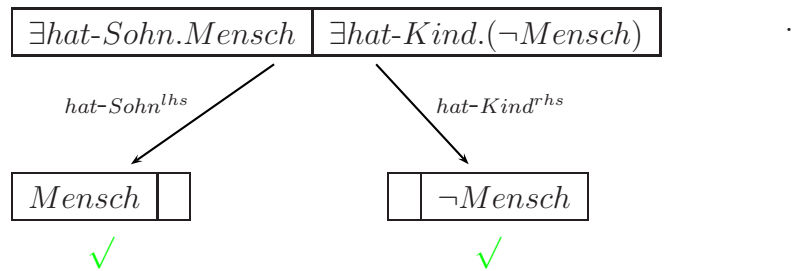
$$Mensch \sqsubseteq \top$$

$$Mann \sqsubseteq Mensch$$

$$VaterMitSohn \doteq Mann \sqcap \exists hat\text{-Sohn}.Mensch$$

$$Elternteil_F \doteq Mensch \sqcap \forall hat\text{-Kind}.Mensch$$

Dies führt dazu, dass  $VaterMitSohn$  nicht von  $Elternteil_F$  subsumiert wird, da folgender Tableau-Knoten (mit zwei Nachfolgern) einen nicht abgeschlossenen Pfad im zugehörigen Tableau bildet:



Beim Wiederaufstieg würde nun in diesem Knoten nach den beiden genannten Mustern gesucht. Zum Ausdruck  $\exists hat\text{-Sohn}.Mensch$  auf der linken Seite würde der Ausdruck  $\exists hat\text{-Kind}.(\neg Mensch)$  auf der rechten Seite gefunden. Dies passt zum ersten Muster. Für  $\exists hat\text{-Sohn}.Mensch$  wird festgestellt, dass es genau in dieser Form aus der Definition von  $VaterMitSohn$  stammt. Dies passt nicht in die hier behandelten Fehlerschemata. Anders sieht es mit  $\exists hat\text{-Kind}.(\neg Mensch)$  aus. Der Ausdruck kommt in  $Elternteil_F$  in der Form  $\forall hat\text{-Kind}.Mensch$  vor, was einer potentiell falschen Allquantifikation entspricht. Daher würde für die entsprechende Teil-Nicht-Subsumtion dieser Änderungsvorschlag generiert:

Die partielle Subsumtion ist nicht gültig.  
 Um dies zu ändern, versuchen Sie folgendes:

1. Ersetzen Sie  $\forall hat\text{-Kind}.Mensch$  in der Definition von  $Elternteil_F$  durch  $\exists hat\text{-Kind}.Mensch$

### 4.3.2 Partielle vs. Vollständige Definition

Jede partielle Definition wird wie beschrieben in eine neue, vollständige Definition umgewandelt. Üblicherweise wird hierbei ein neues Konzept mit einem bestimmten reservierten Suffix, z.B. einem \* definiert. Falls ein solches Konzept also in einem Knoten eines nicht abgeschlossenen Pfades einzeln auf der rechten Seite auftaucht, so deutet dies darauf hin, dass eine mögliche Ursache in der partiellen Definition liegt. Allerdings muss darauf geachtet werden, dass bei einer solchen Definition  $Mann \sqsubseteq Mensch$  nicht davon auszugehen ist, dass ein Fehler vorliegt. Ein entsprechender Vorschlag wird daher nur dann unterbreitet, wenn die Definition aus einem komplexeren Term besteht, wie  $Elternteil_F$  im folgenden Beispiel:

$$Vater \doteq Mann \sqcap \exists hat-Kind.Mensch$$

$$Elternteil_F \sqsubseteq Mensch \sqcap \exists hat-Kind.Mensch$$

Der entscheidende Tableau-Knoten für die Fragestellung  $Vater \sqsubseteq Elternteil_F$  sähe dann so aus:

$Mann \sqcap \exists hat-Kind.Mensch$	$Elternteil_F^*$	⋮	..	..
✓		⊔	⊥	

Der Hinweis für den Benutzer lautet dann:

Vielleicht ist das Problem die partielle Definition von  $Elternteil_F$

### 4.3.3 Vergessene Terme

Vergessene Terme können im Prinzip immer die Ursache einer Nicht-Subsumtion sein. Es ist klar, dass eine Teilsubsumtion durch Hinzufügen der rechten Seite zur linken Seite gültig wird. Dies bedeutet, dass in jeder Teilerklärung einer Nicht-Subsumtion vorgeschlagen werden kann, die rechte Seite zu den die linke Seite definierenden Konzeptdefinitionen hinzuzufügen. Um die Fülle der Vorschläge ein wenig einzuschränken wird ein solcher Vorschlag nur dann gemacht, wenn in einem Knoten auf der rechten Seite nur ein einzelner nicht komplexer Ausdruck steht. In einem solchen Fall werden die Definitionen, aus denen die Terme der linken Seite stammen, gesammelt und für jeden jeweils die Änderung geprüft. Es wird dann je ein Vorschlag dieser Art generiert:

Sie könnten ... zur Definition von: ... hinzufügen

Hierbei wird immer auch überprüft, ob dies dazu führt, dass die linke Seite unerfüllbar wird und somit nur eine triviale Teilsubsumtion erreicht würde. In einem solchen Fall wird der Benutzer davor gewarnt, dass ein Hinzufügen sicher nicht die intendierte Lösung ist:

... zu den Definitionen von: ... hinzuzufügen macht keinen Sinn, da die linke Seite hierdurch unerfüllbar würde.

Dieses eher naive Vorgehen berücksichtigt nicht, dass in einem Knoten auf der linken Seite beispielsweise das Konzept  $C$  auch fehlen kann, weil im Vorgänger  $\exists r.C$  fehlt. Hier wären also durchaus komplexere Vorschläge möglich, was aber in dieser Arbeit nicht weiter verfolgt wurde.

#### 4.3.4 Domain- und Range-Einschränkungen

Für vergessene Domain- und Range-Einschränkungen ist prinzipiell ein ähnliches Vorgehen denkbar, wie für vergessene Terme im letzten Abschnitt. Für jeden Knoten auf einem nicht abgeschlossenen Pfad könnte für jeden  $r^{lhs}$ -Nachfolger eine fehlende Domain-Einschränkung für die entsprechende Relation  $r$  bzw. im Nachfolger eine Range-Einschränkung vorgeschlagen werden. Im Rahmen dieser Arbeit wurde dies jedoch nicht umgesetzt.

### 4.4 Bewertung von Änderungen

Mit den genannten Ideen zum Finden von Änderungsvorschlägen werden häufig mehrere Änderungen für Tableaunknoten von nicht abgeschlossenen Pfaden gefunden. Wichtig ist daher die Bewertung von Änderungsvorschlägen bezüglich der Frage, ob sie vom Benutzer intendiert sein könnten.

Wie man “gute“ Änderungsvorschläge zur Behebung einer Nicht-Subsumtion finden kann bzw. woran man diese erkennen kann ist ein generelles Problem, mit dem sich auch [EKS06, Kal06] beschäftigen (s. auch 6.1). So ist es sicherlich sinnvoll, die triviale Subsumtion zu vermeiden, d.h. Änderungen, die lediglich die rechte Seite äquivalent  $\top$  bzw. die linke Seite äquivalent  $\perp$  werden lassen und damit eine (Teil-)Subsumtion herstellen. Dies wird in [Kal06] als Heuristik vorgeschlagen.

Im Folgenden werden Überlegungen vorgestellt, wie Änderungen sich auf die Konzepthierarchie auswirken und inwiefern dies ein Kriterium für die Bewertung von Änderungsvorschlägen sein könnte.

#### 4.4.1 Wirkung von Änderungen auf die Konzepthierarchie

Jede Änderung an der Definition eines Konzeptes kann weitreichende unbeabsichtigte Auswirkungen haben. Potentiell können Änderungen dazu führen, dass einzelne oder auch mehrere Konzepte unerfüllbar werden. Sie können auch bewirken, dass sich die Konzepthierarchie insgesamt verändert und dies in einem Maße, das über die gewünschte Subsumtion hinaus geht und vielleicht nicht im Sinne des Benutzers ist.

Generell gilt, dass die Auswirkungen einer Änderung nicht lokal betrachtet werden können und die Folgen für die bestehende Konzepthierarchie nicht vorhersehbar sind. Eine Änderung an der Definition eines Konzeptes kann nicht nur die Hierarchie in der Nachbarschaft des Konzeptes verändern, sondern auch an jeder anderen Stelle in der Konzepthierarchie. So könnte eine Änderung an einem Konzept  $F$  nicht nur die bisherigen Sub- und Superkonzepte von  $F$  betreffen, sondern unter Umständen ebenfalls die hierarchische Einordnung von Konzepten, die Konstrukturen der Art  $\exists r.F$  verwenden.

Die hier verfolgte Idee ist, die Wirkung jedes Änderungsvorschlages auf die Konzepthierarchie zu untersuchen. Außerdem wird natürlich geprüft, ob eine Änderung die gewünschte Subsumtion gültig macht. Grundsätzlich ist es wahrscheinlich, dass der Benutzer an Änderungen interessiert ist, die zwar die gewünschte Subsumtion herstellen, aber die Konzepthierarchie ansonsten nicht oder kaum verändern.

Für jeden Änderungsvorschlag wird die Wirkung auf die gesamte Konzepthierarchie berechnet. Um dies zu ermöglichen, wird die entsprechende Änderung temporär durchgeführt und die Konzepthierarchie neu berechnet. Dann wird die folgende einfache Metrik angewendet. Ausgangspunkt für die Berechnung sind die Konzepthierarchien vor und nach der Änderung. Für jedes Konzept (außer  $\perp$ ) wird die Veränderung seiner direkten Superkonzepte gezählt, d.h. die Anzahl der weggefallenen plus die Anzahl der neu hinzugekommenen direkten Superkonzepte. Hiermit kann für jeden Änderungsvorschlag eine Zahl größer oder gleich Null berechnet werden, die die Anzahl der entstehenden Änderungen in der Konzepthierarchie angibt.

Die in jedem Knoten gefundenen Änderungsvorschläge werden danach sortiert, ob sie die gewünschte Subsumtion gültig machen und wie stark sie die Hierarchie verändern würden. Vorschläge, die die Subsumtion gültig machen, stehen vor den

übrigen. Erst als zweites Kriterium werden Vorschläge mit geringen vor solchen mit starken Auswirkungen platziert.

### 4.4.2 Probleme

Die genannten Ideen können dem Benutzer in manchen Fällen helfen, die richtigen Änderungsvorschläge auszuwählen. Das Problem ist jedoch, dass eine Änderung mit starken Auswirkungen auf die Konzepthierarchie auch besser sein kann als eine mit schwachen. Falls beispielsweise in einem Konzept, das sehr weit oben in der Konzepthierarchie steht, ein Fehler vorkommt, so würde eine Behebung des Fehlers zwar voraussichtlich starke Auswirkungen auf die Konzepthierarchie haben, allerdings wären diese wohl vom Benutzer gewollt.

Der Ansatz berücksichtigt auch nicht, dass möglicherweise eine Kombination verschiedener Änderungen die Subsumtion gültig macht, während dies die einzelnen Änderungen nicht erreichen. Solche Änderungsvorschläge würden nachrangig gegenüber einem Vorschlag behandelt, der die Subsumtion erreicht, aber nicht intendiert ist.

Generell kann nicht beurteilt werden welche Änderungen im Sinne des Benutzers sind und welche nicht. Nicht einmal die Konsistenzerhaltung der Wissensbasis ist ein Kriterium, das hier immer hilft. So ist kann es sein, dass auf dem Weg zur intendierten Modellierung mehrere Änderungen durchgeführt werden müssten, von denen einige für sich genommen die Wissensbasis inkonsistent machen würden [EKS06].

## 4.5 Finden von Fehlern durch den Benutzer

Wie bereits beschrieben ist es leicht möglich, dass die tatsächlichen Fehler in einer Ontologie auf die beschriebene Art und Weise nicht gefunden werden und entsprechende Änderungen nicht vorgeschlagen werden. Für jede nicht gültige Teilsubsumtion werden außerdem meist mehrere Änderungsvorschläge gefunden, von denen viele unter dem Gesichtspunkt der intendierten Modellierung nicht sinnvoll sind. Hinzu kommen die prinzipiellen Probleme bei der Bewertung von gefundenen Änderungen, die im letzten Abschnitt angesprochen wurden.

Trotz der vorgestellten Ideen muss die Suche nach Fehlern in der Modellierung daher im Wesentlichen vom Benutzer durchgeführt werden. Der Benutzer muss an Hand von Änderungsvorschlägen überprüfen, ob die betroffenen Definitionen tatsächlich fehlerhaft sind. Gleichzeitig muss er dabei im Auge behalten, wie sich

etwaige Änderungen auf die Konzeptionshierarchie auswirken würden und welche Änderungen in seinem Sinne sind.

Die Erklärung der Nicht-Subsumtion kann bei der Fehlersuche wertvolle Hilfe leisten. Aufgrund ihrer Strukturierung wird das Finden von Fehlern durch den Benutzer unterstützt, da ersichtlich ist, welche Teile des Subsumers nicht erfüllt sind. Im Vergleich zur direkten Suche in den Definitionen des Subsumers und des Subsumees bietet dies eine ganze Reihe von Vorteilen. Müsste der Benutzer die Fehler in den Definitionen eigenhändig suchen, so bliebe ihm nicht anderes, als die Nicht-Subsumtion Schritt für Schritt nachzuvollziehen. Erschwerend wäre vor allem die Tatsache, dass er quasi von Hand die in den Definitionen vorkommenden Konzepte auffalten müsste.

Aus den genannten Gründen ist die Suche und Korrektur von Fehlern eine Aufgabe, die letztendlich vom Benutzer durchgeführt werden muss. Die vorgestellten Ideen können ihn hierbei aber unterstützen.



---

## 5 Implementierung

Im Rahmen dieser Arbeit wurde ein prototypischer Tableau-Reasoner implementiert, der die vorgestellten Ideen umsetzt. Das Hauptziel war eine Umsetzung der Ideen zur Erklärung und Korrektur, weswegen Performanceaspekte nur am Rande berücksichtigt wurden.

Unterstützt wird die Beschreibungslogik  $\mathcal{SHF}$ , deren Konstruktoren in Abbildung 2.1 erläutert werden. Als Einschränkung sind nur atomare Domain- und Range-Einschränkungen erlaubt und die TBox darf keine zyklischen Definitionen beinhalten.

Das System ist in Java (Version 1.5) programmiert und nutzt für die Verwaltung der logischen Terme zur Laufzeit die ATerm-Bibliothek ([vdBdJKO00]), ein Termverwaltungssystem. Außerdem wird aus Effizienzgründen der Reasoner Pellet (Version 1.2) verwendet, um die Auswirkungen von Änderungen auf die Konzepthierarchie zu berechnen<sup>1</sup>. Für die Darstellung der Erklärung und von Korrekturvorschlägen wurde eine grafische Benutzerschnittstelle integriert. Es ist möglich, eine Subsumtionsfragestellung inklusive TBox in einer KRSS ähnlichen Syntax aus einer Datei zu laden. Als Entwicklungsumgebung wurde Eclipse 3.2 verwendet.

### 5.1 Funktionsweise

Im Folgenden wird das Vorgehen der Erklärungs- und Korrekturkomponente erläutert. Zunächst wird für eine gegebene TBox und eine Subsumtionsfragestellung ein Tableau-Beweis geführt, um die Frage zu klären, ob die Subsumtion gilt. Hierbei wird eine Erklärung für das Ergebnis generiert, die auch gültige Teilsubsumtionen erklärt. Falls die Subsumtion nicht gilt, wird für jeden Tableaunknoten, in dem kein Clash auftritt, nach potentiellen Änderungen gesucht. Für jede gefundene Änderung lässt das System von Pellet die Konzepthierarchie berechnen, die durch die Änderung entstehen würde. Dann wird der Änderungswert berechnet und überprüft, ob die Änderung die Subsumtion gültig machen würde. Die Änderungsvorschläge werden anhand ihrer Auswirkungen auf die Konzepthierarchie sortiert. Macht eine Änderung die Subsumtion gültig, so wird sie vor jeder Änderung prä-

---

<sup>1</sup>s. Kapitel 2.6

sentiert, für die dies nicht der Fall ist. Außerdem werden Vorschläge mit geringen vor solchen mit starken Auswirkungen auf die Konzepthierarchie positioniert.

Die Erklärung für die Nicht-Subsumtion wird dem Benutzer grafisch präsentiert und er kann sich zu den Teilen der Erklärung der Nicht-Subsumtion die gefundenen Änderungsvorschläge anzeigen lassen.

Für die Erklärung und Suche nach Korrekturvorschlägen müssen eine ganze Reihe von zusätzlichen Informationen gespeichert werden, die in Reasonern ohne Erklärungs- und Korrekturfunktionalität nicht benötigt werden. So werden, wie in 3.1 beschrieben, die linke und rechte Seite getrennt gespeichert, um an jeder Stelle im Tableau die von Subsumee und Subsumer stammenden Teile des Terms unterscheiden zu können. Außerdem wird beim Auffalten und Erzeugen von Nachfolgern gespeichert, woher die eingefügten Terme stammen, um die Auswirkungen von Änderungen berechnen zu können. Für den Fall eines Clashes werden ggf. die beteiligten Konzepte gespeichert, um die in 3.3 genannten Unterscheidungen vornehmen zu können.

## 5.2 Grafische Benutzerschnittstelle

Das System besitzt eine grafische Benutzerschnittstelle.

In der TBox-Ansicht kann mittels des Buttons "Load" eine TBox mit Subsumtionsfragestellung aus einer Datei geladen werden. Diese wird im Fenster angezeigt. Dort kann man auch eine geladene TBox verändern bzw. von Hand eine eigene eingeben. Mit dem Button "Retry" wird erneut die Subsumtion überprüft sowie eine Erklärung generiert und nach Änderungsvorschläge gesucht (s. Abb. 5.1).

Nachdem das System die Subsumtionsbeziehung geklärt hat, kann die Erklärung in der Erklärungs-Ansicht grafisch angezeigt werden (s. Abb. 5.2).

Die Erklärung wird als Baum angezeigt und die einzelnen Äste sind ausblendbar, so dass der Benutzer die für ihn aktuell unwichtigen Teile der Erklärung verbergen kann. Dies sorgt bei umfangreicheren Erklärungen für eine erhöhte Übersichtlichkeit. Teile der Erklärung, die eine Subsumtion beschreiben, sind grün eingefärbt, solche, die eine Nicht-Subsumtion beschreiben rot. Bei gültigen (Teil-)Subsumtionen wird die Übereinstimmung auf linker und rechter Seite mit grünem Rahmen versehen. Hinzu kommt eine Sortierung nach gültigen und nicht gültigen (Teil-)Subsumtionen. Die dargestellten Terme sind mit einem Tooltip versehen, der für jeden Bestandteil dessen Herkunft anzeigt, d.h. aus welcher Definition er stammt.

Bei nicht gültigen (Teil-)Subsumtionen wird durch einen Klick auf das Fragezeichen

eine Dialogbox geöffnet, die gefundene Änderungsvorschläge auflistet (s. Abb. 5.3). Vorschläge, die die Subsumtion gültig machen, sind grün gefärbt.

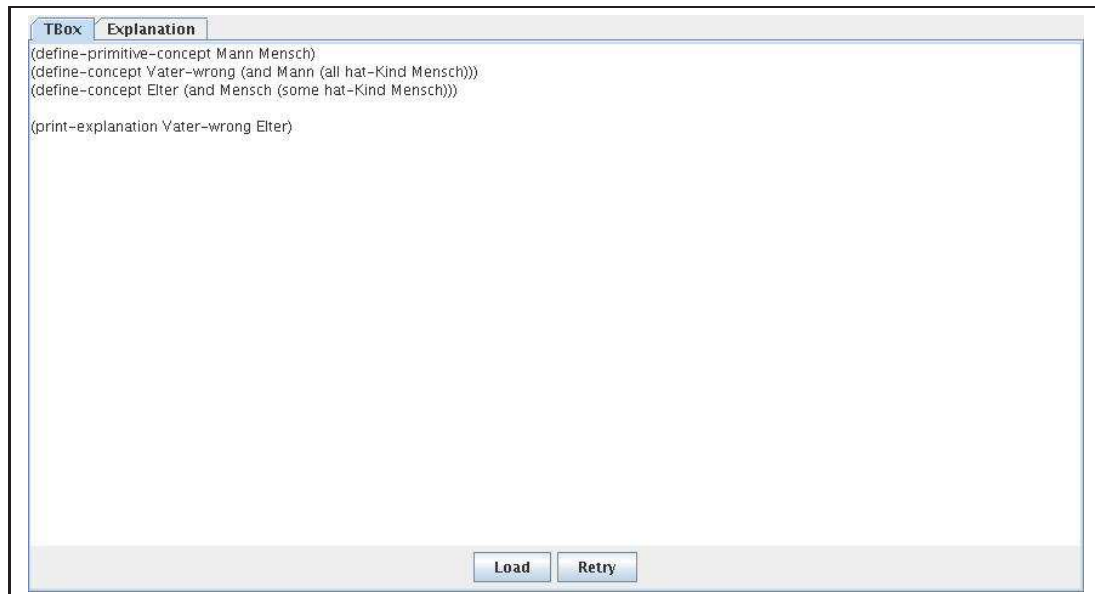


Abbildung 5.1: TBox-Editor Ansicht

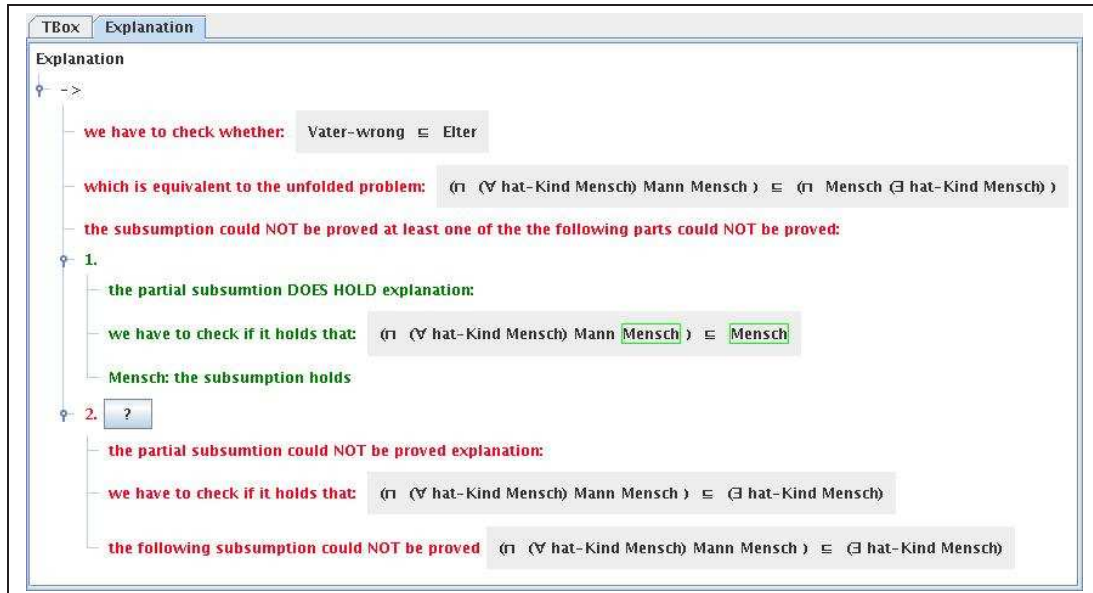


Abbildung 5.2: Visualisierung einer Erklärung

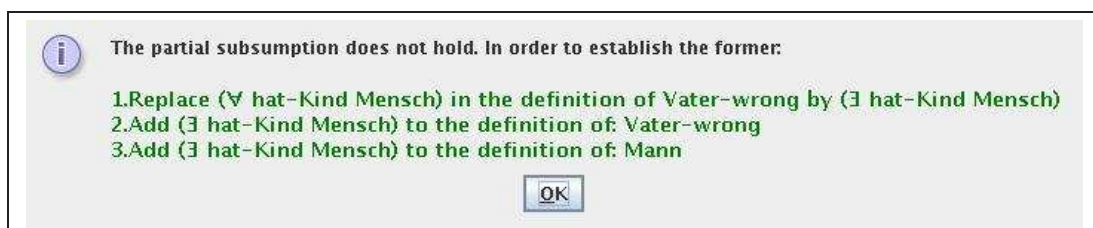


Abbildung 5.3: Visualisierung von Änderungsvorschlägen

---

## 6 Thematisch verwandte Arbeiten

Im den folgenden Abschnitten werden Arbeiten zu diesem bzw. ähnlichen Themen vorgestellt. Außer den folgenden wurde bereits in 3.2 eine Arbeit zum Erklären von Subsumtion vorgestellt ([LH05]), auf der die Erklärung von Nicht-Subsumtion in dieser Arbeit aufbaut.

### 6.1 Nicht-Subsumtion

Zum Thema der Erklärung und Korrektur von Nicht-Subsumtion existieren bisher sehr wenige Veröffentlichungen.

In [Kal06] wird u.a. auch auf die Thematik der Nicht-Subsumtion eingegangen. Allerdings wird dieses Thema nur im Ausblick kurz angesprochen. Warum eine bestimmte logische Schlussfolgerung **nicht** gilt ist demnach deutlich schwieriger zu erklären und zu beheben als ihre Gültigkeit. Dies ist auch im Falle einer nicht geltenden Subsumtion so. Bezogen auf einen geführten Tableau-Beweis liegt dann ein Tableau-Graph vor, der nicht abgeschlossen ist. Es existieren potentiell unendlich viele Möglichkeiten dies zu beheben, d.h. einen Clash zu erzeugen. Eine Eingrenzung auf bestimmte Axiome der Ontologie ist nicht möglich. Als triviale, aber eher unbefriedigende Lösung kann die Subsumtion als Axiom eingefügt werden. Eine besserer Ansatz ist jedoch, nicht-triviale Axiome vorzuschlagen, deren Einfügen die Subsumtion wahr macht. Der Autor schlägt einen heuristischen Ansatz mit zwei Heuristiken vor, um möglichst gute und hilfreiche Vorschläge zu unterbreiten. Die erste Heuristik ist, nur Clashes im Tableau zu erzeugen, die durch Interaktion von Subsumee und Subsumer hervorgerufen werden. Dies wird auch in dieser Arbeit angemerkt (s. 4.4). Hierdurch wird verhindert, dass Subsumer oder Subsumee für sich genommen unerfüllbar werden. Als zweite Heuristik zur Beurteilung einer Änderung wird die Größe des sog. *Justification-Sets* für die gewünschte Subsumtion vorgeschlagen. Ein *Justification-Set* ist die Menge von Axiomen der Wissensbasis, die für eine Schlussfolgerung verantwortlich sind, d.h. die Clashes im Tableau direkt oder indirekt mit verursacht haben. Die Idee ist, dass ein größeres *Justification-Set* auch eine weniger triviale Lösung bedeuten muss. Ein Vorschlag zur Berechnung eines *Justification-Sets* ist, die Herkunft jedes Axioms beim Aufbau des Tableaus zu speichern (*Tableau Tracing*). Dadurch kann für jeden

Clash die Menge von Axiomen angegeben werden, die direkt oder indirekt an ihm beteiligt sind und damit ein *Justification-Set* für eine Schlussfolgerung.

In [EKS06] geht es um das abduktive Schlussfolgern im Kontext von Beschreibungslogiken. Das abduktive Schließen beschäftigt sich allgemein mit dem Finden von erklärenden Hypothesen für eine Beobachtung.<sup>1</sup> Die Arbeit geht im Besonderen auf die Thematik der Nicht-Subumtion ein und macht Vorschläge, wie mit Hilfe von Techniken des abduktiven Schlussfolgers die Berechnung von Erklärungen und Korrekturvorschlägen aussehen könnte. So wird vorgeschlagen, nach Lösungen für das so genannte *TBox Abduction*-Problem zu suchen. Dieses beschreibt die Frage nach endlichen Mengen  $\mathcal{S}_T$  von TBox-Axiomen, durch deren Hinzufügen zu einer Wissensbasis  $\Gamma$  eine bis dahin nicht gültige Subsumtion  $C \sqsubseteq D$  gültig wird, formal:

$$\Gamma \cup \mathcal{S}_T \models C \sqsubseteq D$$

Die Menge aller Lösungen wird mit  $\mathcal{S}_T(\Gamma, C, D)$  bezeichnet. Lösungen für dieses Problem machen die Subsumtion gültig und können somit eine unerwünschte Nicht-Subsumtion “reparieren“. Weiter wird auch vorgestellt, wie gefundene Lösungen klassifiziert werden können. Diese orientieren sich an Standardklassifizierungen für Lösungen von abduktiven Problemen. Die vier genannten Kriterien für eine Lösung  $\mathcal{S}$  und eine Wissensbasis  $\Gamma$  sind:

1. **konsistent:**  $\Gamma \cup \mathcal{S} \not\models \perp$
2. **relevant:**  $\mathcal{S} \not\models C \sqsubseteq D$
3. **erklärend: relevant** und  $\Gamma \not\models C \sqsubseteq D$
4. **kreativ:** mit der Lösung werden neue Relationen bzw. Konzepte eingeführt

Die Autoren stellen fest, dass die verschiedenen Kriterien in ihrer Qualität nicht eindeutig zu beurteilen sind. So kann beispielsweise am Kriterium der Konsistenz bemängelt werden, dass es verhindert, dass Revisionen an der Wissensbasis vorgenommen werden können: Unter Umständen ist zwar die Wissensbasis  $\Gamma \cup \mathcal{S}$  inkonsistent, jedoch  $\Gamma' \cup \mathcal{S}$  (mit einer veränderten Wissensbasis  $\Gamma'$ ) konsistent. Das Kriterium würde solche Lösungen ausschließen, obwohl sie intendiert sein könnten.

Laut [EKS06] können abduktive Probleme im Allgemeinen unendlich viele verschiedene Lösungen haben. Deshalb wird noch die **Minimalität** als Unterscheidungskriterium für Lösungen vorgeschlagen. Die Unterscheidung kann syntaktischer Art sein, d.h. an Hand der Länge oder aber auch semantischer Art. Ein weiteres interessantes Kriterium ist, wie **erhaltend** eine Lösung ist, das bedeutet, wie stark die

---

<sup>1</sup>Für weitergehende Informationen zum Thema abduktives Schließen sei auf [AL97] verwiesen.

Auswirkungen auf die Konzepthierarchie sind. Beispielsweise könnte es sein, dass Lösungen bevorzugt werden sollen, die die Hierarchie wenig verändern, was auch in der vorliegenden Arbeit diskutiert wird (s. 4.4.1). Es könnte auch sein, dass in einer Ontologie, die als Basis für eine andere verwendet wird, die Hierarchie unverändert bleiben soll.

## 6.2 Unerfüllbarkeit

Es gibt bereits Methoden, um das Auffinden und die Korrektur ungewollt unerfüllbarer Konzepte in Ontologien zu ermöglichen. Wie in 4.1 beschrieben sind die Ursachen von unerfüllbaren Konzepten wesentlich einfacher zu finden als dies bei Nicht-Subsumtion der Fall ist. Die Gründe für unerfüllbare Konzepte lassen sich auf bestimmte Axiome der Ontologie eingrenzen. Die zwei folgenden Ansätze sollen dem Benutzer ein “Debugging“ einer Ontologie erleichtern, indem sie die Gründe für die Unerfüllbarkeit darlegen und versuchen, die Ursachen aufzuzeigen.

In [KPSCG] wird versucht unerfüllbare Konzepte in OWL-Ontologien zu “reparieren“ indem dem Benutzer Erklärungen für die Ursachen der Unerfüllbarkeit präsentiert werden. Hierzu wird eine sog. **Minimal Unsatisfiability Preserving SubTBox** für jedes unerfüllbare Konzept  $A$  berechnet ( $MUPS(A)$ , s. [SC03]). Dies ist die kleinste unerfüllbare Teilmenge der ursprünglichen TBox, da sie durch Herausnahme eines beliebigen Axioms erfüllbar wird. Man berechnet also die Menge von Axiomen, die durch ihr Zusammenwirken die Unerfüllbarkeit des Konzeptes  $A$  bewirken. In einem weiteren Schritt wird die Bedeutung der einzelnen Axiome für die Unerfüllbarkeit bewertet, was es dem Benutzer erleichtern soll, den Fehler zu beseitigen. So werden beispielsweise Axiome, die in mehreren  $MUPS$ -Mengen vorkommen, höher bewertet, da sie die Unerfüllbarkeit gleich mehrerer Konzepte bedingen. Außerdem werden die sonstigen Auswirkungen einer möglichen Entfernung eines Axioms in die Bewertung mit einbezogen. Es wird berechnet, wie viele zuvor berechnete Schlussfolgerungen ihre Gültigkeit verlieren, falls man das Axiom entfernt. Auch lässt sich die syntaktische Relevanz als Bewertung verwenden: Es wird gezählt, wie häufig eine OWL-Entität referenziert wird.

Vom Ansatz ähnlich wird in [WHR<sup>+</sup>05] vorgegangen. Für ein unerfüllbares Konzept werden zunächst die **basic debugging necessary conditions** (BDNC) bestimmt. Dies ist die Menge aller Axiome, die das Konzept definieren. Hieraus wird dann der **Unsatisfiable Core** (UC) extrahiert, welcher analog zum oben genannten  $MUPS$  definiert ist. Anschließend werden die **Debugging Super Conditions** (DSC) generiert, eine Menge globaler Axiome (insbesondere Domain-/Range-Einschränkungen und Disjoint-Axiome), die potentiell auch für die Unerfüllbarkeit verantwortlich sein können. Aus den DSC wird der **Most General Conflict** (MGC) gebildet, wo-

bei dieser ein Teilmenge der DSC ist, die nur jeweils die allgemeinsten Konzepte enthält. An Hand von Erfahrungen bezüglich der häufigsten Fehler bei der Erstellung von Ontologien wird die MGC-Menge analysiert.

---

## 7 Bewertung und Ausblick

### 7.1 Vorgehensweise

Zum Thema Erklärung bzw. Korrektur von Nicht-Subsumtion wurde bisher wenig geforscht. Diese Arbeit ist somit eine der ersten, die sich mit diesem Thema befasst. Die beschriebenen Ideen liefern brauchbare Ergebnisse und es ist machbar, Erklärungs- und Korrekturfunktionalität in das Tableauverfahren zu integrieren. Jedoch hat sich gezeigt, dass insbesondere die Korrektur von Nicht-Subsumtion eine sehr schwierige Aufgabe ist.

#### 7.1.1 Korrektur

Bei einer Nicht-Subsumtion existiert ein Modell für die Anfrage. Es können keine für die Nicht-Subsumtion direkt verantwortlichen Axiome gefunden werden. Aus diesem Grund ist die Suche nach Ursachen einer Nicht-Subsumtion sehr viel schwieriger als die Suche nach Ursachen von Unerfüllbarkeit oder Subsumtion. Bei letzteren liegen klare Ursachen für die Inferenz vor. Bei der Suche nach möglichen Ursachen einer Nicht-Subsumtion ist der Suchraum jedoch unendlich groß und unter gefundenen Lösungen intendierte von nicht intendierten zu unterscheiden nicht möglich.

Der in dieser Arbeit vorgestellte Ansatz ist die tableaubasierte Suche nach potentiellen Modellierungsfehlern in Konzeptdefinitionen und das Unterbreiten entsprechender Änderungsvorschläge. Die Grundlage stellt eine Untersuchung typischer Fehler unerfahrener Benutzer dar [RDH<sup>+</sup>04]. Durch diese Heuristik wird der Suchraum eingeschränkt, was bessere Ergebnisse verspricht als eine uninformierte Suche.

Es hat sich herausgestellt, dass die hier vorgestellten Methoden nicht immer zufriedenstellend funktionieren. Auch greifen die Methoden nur bei den genannten Fehlern. Trotzdem ist dies ein Konzept, das nicht zuletzt unerfahrenen Benutzern Hinweise auf Probleme geben kann und helfen kann, eine Nicht-Subsumtion zu korrigieren.

Dennoch bleibt der Suchraum trotz Anwendung der Heuristik sehr groß, es können

potentiell sehr viele Änderungen vorgeschlagen werden. Daher wurde im Rahmen dieser Arbeit eine Metrik zur Bewertung von gefundenen Änderungsvorschlägen entwickelt. Diese gibt darüber Auskunft, wie stark eine Änderung an einer Konzeptdefinition in die bisherige Konzepthierarchie eingreift, indem für jedes Konzept der TBox die Anzahl der Veränderungen seiner direkten Superkonzepte durch die Änderung gezählt wird. Es wird davon ausgegangen, dass der Benutzer oftmals an Änderungen interessiert ist, die die Subsumtion herstellen, die jedoch die Konzepthierarchie ansonsten wenig verändern. Ein Ergebnis dieser Arbeit ist aber, dass dies nicht zwingend so sein muss und auch Änderungen mit starken Auswirkungen auf die Konzepthierarchie intendiert sein können.

Die beschriebene Metrik ist ein erster interessanter Ansatz, jedoch aus den genannten Gründen sicherlich noch erweiterungsfähig. Eine objektive automatische Bewertung der Qualität einer Änderung ist jedoch nicht möglich. Dies ist auch ein Ergebnis dieser Arbeit. Der Benutzer muss letztendlich entscheiden, was eine sinnvolle Änderung ist und was nicht.

Eine Idee wäre, Angaben des Benutzers zu verwenden, wo er Fehler vermutet, um hier noch zielgerichteter vorzugehen. So wäre es vielleicht hilfreich, die Suche nach Korrekturmöglichkeiten konfigurierbar zu gestalten. Würde z.B. eine andere Ontologie als Basis für eine Modellierung verwendet, könnte der Benutzer angeben, dass er nur in der eigenen Modellierung Fehler vermutet. Hierdurch würde der Suchraum weiter eingeschränkt werden. In diese Richtung gehen auch Überlegungen in [EKS06].

### 7.1.2 Erklären

Die in das Tableauverfahren integrierte Erklärungskomponente generiert Erklärungen für Nicht-Subsumtion und funktioniert sehr ähnlich wie die tableaubasierte Erklärung von Subsumtion in [LH05]. Die Erklärung wird parallel zum Aufbau des Tableaus generiert und beschreibt in textueller Form schrittweise, welche Teilprobleme überprüft werden und was das Ergebnis ist. Um in der Erklärung die Unterscheidung von Subsumer und Subsumee zu ermöglichen werden diese immer getrennt gespeichert. Die durch das Beweisverfahren bedingte Negation des Subsumers wird vor dem Benutzer verdeckt, indem sie in Erklärungen rückgängig gemacht wird.

Ein wichtiger Unterschied zu [LH05] ist, dass die Ergebnisse in den Erklärungen von Teilsubsumtionen beim Aufbau des Tableaus zunächst offen bleiben müssen. Der Grund ist, dass eine Nicht-Subsumtion nur bedeutet, dass mindestens eine Teilsubsumtion nicht gültig ist. Das Ergebnis wird daher erst beim Wiederaufstieg im Tableau in die Erklärung eingefügt.

Nach eigener Meinung sind die generierten Erklärungen relativ gut verständlich. Es sollte auch für Benutzer ohne die Kenntnis des Tableauverfahrens nachvollziehbar sein, warum eine Subsumtion nicht gilt, sofern sie über grundlegende Kenntnisse von Beschreibungslogiken verfügen.

### 7.1.3 Allgemeines

Der hier vorgestellte Ansatz zur Erklärung und Korrektur bezieht sich nur auf den beschränkten Sprachumfang  $\mathcal{SHF}$ . Um den Sprachumfang von OWL-DL abzudecken, müsste er noch erweitert werden. Allerdings zeigt eine statistische Untersuchung von im Internet verwendeten Ontologien, dass weit über die Hälfte nicht über die Ausdrucksmächtigkeit von  $\mathcal{ALC}$  hinausgehen [WPH06]. Daher sind wohl auch für den hier behandelten Sprachumfang bereits reale Anwendungen denkbar.

Durch die in das Tableauverfahren integrierte Erklärungs- und Korrekturkomponente werden auch bei größeren Ontologien nur die Definitionen vom Verfahren berührt, die auch für die Fragestellung relevant sind. Dies ist unter dem Gesichtspunkt der Performance ein Vorteil gegenüber anderen Verfahren.

Zusammenfassend kann auch festgestellt werden, dass die in die Erklärungskomponente integrierte Korrektur wohl einer reinen Korrekturkomponente überlegen ist. Auf diese Weise hat der Benutzer die Option, sich das Zustandekommen der Nicht-Subsumtion erklären zu lassen und an Hand von Hinweisen durch die Korrekturkomponente mögliche Fehler zu entdecken.

Letztendlich ist es auf Grund der genannten Probleme immer der menschliche Benutzer, der mit seiner Intuition und seinem Abstraktionsvermögen in der Lage ist, die tatsächlichen Fehler zu entdecken und zu beseitigen.

## 7.2 Implementierung

Die vorgestellte prototypische Implementierung zeigt, dass eine in einen Tableaubeweiser integrierte Erklärungskomponente für Nicht-Subsumtion machbar ist. Auch die Berechnung von Änderungen, die potentiell die Gültigkeit einer gewünschten Subsumtionsbeziehung herstellen, lässt sich in einen Tableau-Reasoner einbetten. Jedoch bestehen die im Abschnitt 7.1 genannten prinzipiellen Einschränkungen.

Die grafische Darstellung bietet für den Benutzer einige Vorteile. Durch die strukturierte Visualisierung der Erklärung und die farbliche Hervorhebung gültiger bzw. ungültiger Teilsubsumtionen kann der Benutzer schnell und einfach erfassen, wel-

che Teile des Subsumers nicht erfüllt werden. Die Korrekturvorschläge werden dort präsentiert, wo auch die Erklärung für das potentielle Problem zu finden ist.

Da die Erklärung und Korrektur von Nicht-Subsumtion insbesondere als Unterstützung bei der Modellierung von Ontologien hilfreich ist, wäre es sicherlich sinnvoll, diese Funktionalität in einen entsprechenden Editor zu integrieren. Hierbei sollte es möglich sein, potentielle Fehler auch in den betroffenen Definitionen anzeigen zu lassen und Änderungsvorschläge direkt zu übernehmen.

## Literaturverzeichnis

- [AL97] A. Aliseda-Llera. *Seeking Explanations: Abduction in Logic, Philosophy of Science and Artificial Intelligence*. Final; reviewed, Universiteit van Amsterdam, January 01 1997.
- [BCM+03] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
- [BFH00] Alexander Borgida, Enrico Franconi, and Ian Horrocks. Explaining alc subsumption. In *ECAI*, pages 209–213, 2000.
- [BS01] Franz Baader and Ulrike Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 69(1):5–40, 2001.
- [EKS06] Corinna Elsenbroich, Oliver Kutz, and Ulrike Sattler. A case for abductive reasoning over ontologies. In *Proceedings of OWL: Experiences and Directions*, 2006.
- [HM01] Volker Haarslev and Ralf Möller. Description of the RACER system and its applications. In Carole A. Goble, Deborah L. McGuinness, Ralf Möller, and Peter F. Patel-Schneider, editors, *Working Notes of the 2001 International Description Logics Workshop (DL-2001), Stanford, CA, USA, August 1-3, 2001*, volume 49 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2001.
- [Kal06] Aditya Kalyanpur. *Debugging and Repair of OWL Ontologies*. PhD thesis, University of Maryland Institute for Advanced Computer Studies, Maryland, 2006.
- [KPSCG] Aditya Kalyanpur, Bijan Parsia, Evren Sirin, and Bernardo Cuenca-Grau. Repairing Unsatisfiable Concepts in OWL Ontologies. Accepted for Publication in ESWC2006.
- [LH05] Thorsten Liebig and Michael Halfmann. A tableau-based explainer for dl subsumption. In *TABLEAUX*, pages 323–327, 2005.

- [MvH04] Deborah L. McGuinness and Frank van Harmelen. OWL web ontology language overview. World Wide Web Consortium, Recommendation REC-owl-features-20040210, February 2004.
- [RDH<sup>+</sup>04] Alan L. Rector, Nick Drummond, Matthew Horridge, Jeremy Rogers, Holger Knublauch, Robert Stevens, Hai Wang, and Chris Wroe. Owl pizzas: Practical experience of teaching owl-dl: Common errors & common patterns. In *EKAW*, pages 63–81, 2004.
- [SC03] Stefan Schlobach and Ronald Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In Georg Gottlob and Toby Walsh, editors, *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, pages 355–362. Morgan Kaufmann, 2003.
- [SPG<sup>+</sup>06] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical owl-dl reasoner. *Journal of Web Semantics*, 2006.
- [vdBdJKO00] Mark van den Brand, H. A. de Jong, Paul Klint, and Pieter A. Olivier. Efficient annotated terms. *Softw, Pract. Exper*, 30(3):259–291, 2000.
- [WHR<sup>+</sup>05] Hai Wang, Matthew Horridge, Alan L. Rector, Nick Drummond, and Julian Seidenberg. Debugging owl-dl ontologies: A heuristic approach. In *International Semantic Web Conference*, pages 745–757, 2005.
- [WPH06] Taowei David Wang, Bijan Parsia, and James A. Hendler. A survey of the web ontology landscape. In Isabel F. Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Michael Uschold, and Lora Aroyo, editors, *The Semantic Web - ISWC 2006, 5th International Semantic Web Conference, ISWC 2006, Athens, GA, USA, November 5-9, 2006, Proceedings*, volume 4273 of *Lecture Notes in Computer Science*, pages 682–694. Springer, 2006.

Name: Julian Lambertz  
Matrikelnummer: 448021

## Erklärung

Ich erkläre, dass ich die Diplomarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den 5. Februar 2007 .....