

Universität Ulm
Abteilung Künstliche Intelligenz

Diplomarbeit

**Entwicklung einer Erklärungskomponente
für tableauxbasierte Subsumtionsbeweise**

vorgelegt von:
Michael Halfmann



1. Gutachter:
Professor Dr. Friedrich von Henke

2. Gutachter:
Dr. Thorsten Liebig

Zusammenfassung

Mit der Entwicklung des Semantic Web werden Ontologien und maschinelle Inferenzdienste nicht länger nur von Spezialisten verwendet werden. Den Umgang mit diesen zu vereinfachen stellt somit einen wichtigen Aspekt von Semantic Web spezifischen Applikationen dar. Die Semantic Web Sprache OWL basiert auf dem Sprachformalismus der Beschreibungslogiken. Ein typischer Inferenzdienst für Beschreibungslogiken ist die Subsumtion.

Maschinelle Inferenzverfahren basieren typischerweise auf äußerst komplizierten Algorithmen. Den Lösungsweg eines solchen Verfahrens nachzuvollziehen, gelingt in der Regel nur Experten, welche die zugrundeliegenden Algorithmen sehr genau studiert haben. Für viele Anwendungsszenarien kann es jedoch hilfreich sein, zu verstehen, warum das Verfahren zum vorliegenden Ergebnis gekommen ist. Dies kann unterstützt werden, indem der Inferenzdienst zusätzlich zum Ergebnis eine Erklärung für dieses liefert. Für diese zusätzliche Funktionalität hat sich der Begriff des *Explaining* etabliert.

Ein Standardverfahren, um Subsumtion zu prüfen, ist das Tableau-Verfahren. Auch im Kontext des Semantic Web spielt dieses Verfahren eine große Rolle. Das Tableau-Verfahren basiert auf einem Widerspruchsbeweis. Um eine Anfrage zu beweisen, wird sie negiert und daraufhin versucht, dieser negierten Form Unerfüllbarkeit nachzuweisen. Dazu wird nach vorgegebenen Regeln ein so genannter Tableau aufgebaut, welcher die in der negierten Anfrage geforderten Zusammenhänge repräsentiert. Der Nachweis der Unerfüllbarkeit erfolgt durch das Lokalisieren eines expliziten Widerspruchs innerhalb dieses Tableaus. Für unerfahrene Benutzer ist diese Vorgehensweise äußerst schwer nachzuvollziehen.

In dieser Diplomarbeit wird eine Form des Erklärens von tableaubasierten Subsumtionsbeweisen vorgestellt. Es wird ein theoretischer Ansatz für das Erklären von Subsumtionsbeziehungen im Sprachumfang der Beschreibungslogik *SHIF* geliefert. Der Algorithmus prüft eine Subsumtionsbeziehung unter Berücksichtigung der aktuellen T-Box und liefert, falls diese gültig ist, eine Erklärung. Diese besteht aus einer Liste von Erklärungsschritten, die während des Tableaubaufbaus erstellt werden. Im Rahmen der Arbeit wurden Erklärungsmöglichkeiten für eine Vielzahl von beschreibungslogischen Konstrukten entwickelt. Das Hauptaugenmerk lag auf der Generierung möglichst intuitiv verständlicher Erklärungen. Performanceaspekte des Tableau-Algorithmus wurden nur am Rande berücksichtigt.

Zusätzlich zu dem generellen Verfahren, einen Tableau-Beweis für eine Subsumtionsbeziehung zu erklären, werden einige Optimierungstechniken vorgestellt. Diese zielen darauf ab, die Qualität der Erklärungen zu erhöhen. Dazu werden verschiedene zusätzliche Operationen, wie zum Beispiel ein strukturel-

ler Vergleich von Subsumer und Subsumee benötigt.

Basierend auf dem erarbeiteten Ansatz wurde im Rahmen dieser Arbeit eine prototypische Implementierung für die Beschreibungslogik $\mathcal{ALSHF}_{\mathcal{R}^+}$ entwickelt. Der von diesem System behandelbare Sprachumfang wurde gegenüber dem theoretischen Ansatz leicht eingeschränkt. Das System arbeitet auf einer gegebenen T-Box und beweist die Gültigkeit von Subsumtionsbeziehungen mittels des Tableau-Verfahrens. Für gültige Subsumtionen wird zusätzlich eine quasi-natürlichsprachliche Erklärung generiert. Die Implementierung dieses Systems half das entwickelte Verfahren zu evaluieren und zusätzliche Erkenntnisse bezüglich verschiedener Optimierungen zu gewinnen.

Es hat sich gezeigt, dass das Erklären von tableaubasierten Subsumtionsbeweisen ein äußerst kompliziertes Unterfangen darstellt. Selbst für den relativ geringen Sprachumfang \mathcal{SHIF} müssen einige Aspekte beachtet werden, um intuitiv verständliche Erklärungen generieren zu können. In dieser Arbeit werden die erarbeiteten Ansätze, sowie gewonnene Erkenntnisse und Ideen zur weiteren Verbesserung diskutiert. Des Weiteren wird ein Überblick über bereits existierende sowie sich in Entwicklung befindende Projekte gegeben, welche sich mit ähnlichen Thematiken beschäftigen.

Zu dem im Rahmen der Arbeit erarbeiteten Verfahren wurden zwei Publikationen geschrieben und für den International Workshop on Description Logics (DL 2005) bzw. die International Conference TABLEAUX 2005 eingereicht.

Inhaltsverzeichnis

Zusammenfassung	iii
Abbildungsverzeichnis	vii
1 Einleitung	1
1.1 Semantic Web	1
1.2 Ontologien	2
1.3 Erklären	3
1.4 Überblick	3
2 Grundlagen	5
2.1 Beschreibungslogik	5
2.2 Web Ontology Language (OWL)	7
2.3 Inferenzdienste	8
2.3.1 Subsumtion	8
2.3.2 Unerfüllbarkeit	8
2.4 Schlussfolgerungsalgorithmen	8
2.4.1 Strukturelle Subsumtion	8
2.4.2 Tableau-Algorithmus	9
2.5 Inferenzsysteme	9
2.5.1 RACER	9
2.5.2 Pellet	10
3 Tableau-Verfahren für <i>SHIF</i>	11
3.1 Grundlagen des Tableau-Verfahrens	11
3.2 Blocking	14
3.3 Optimierungen	16
3.3.1 Lazy-Unfolding	16
4 Tableaubasiertes Erklären	19
4.1 Grundlegendes zum Erklären	19
4.2 Erklären von Subsumtionsbeziehungen	20
4.2.1 Umschalten des Erklärungsmodus	21
4.3 Einzelschritte und ihre Erklärungen	22
4.3.1 Auffaltung	22
4.3.2 Übergang zum Nachfolgerknoten	23
4.3.3 Clash durch \perp -Konzept	23
4.3.4 Clash durch Konjunktion eines Konzepts und dessen Ne- gation	24

4.3.5	Clash durch widersprüchliche Kardinalitätseinschränkungen	25
4.3.6	Disjunktionen	26
4.3.7	domain- und range-Restriktionen	26
4.3.8	Merging von Relationsfüllern	27
4.3.9	Funktionale Relationen	28
4.3.10	Subrelationen und äquivalente Relationen	28
4.3.11	Transitive Relationen	30
4.3.12	Inverse Relationen	31
4.3.13	Zusammenfassung	34
4.4	Erklärungsbeispiele zu komplexen Subsumtionsanfragen	37
4.5	Diskussion	44
4.5.1	Reine Subsumtionserklärung vs. verschiedene Erklärungs- Modi	44
4.5.2	Filtern von besonderen Situationen	47
4.5.3	Ausblenden nicht relevanter Konstrukte	47
4.5.4	Komplexes Merging	48
5	Umsetzung	49
5.1	Die Erklärungskomponente	49
5.2	Entwicklungsumgebung	49
5.3	Datenstruktur	49
5.3.1	T-Box	50
5.3.2	Rollendefinition	50
5.3.3	Constraint	50
5.3.4	Erklärungsschritt	52
5.4	Sprachumfang	53
5.5	Allgemeine Vorgehensweise	54
5.6	Optimierungen	55
5.6.1	lazy-unfolding	55
5.6.2	RACER Anbindung	55
5.6.3	Filtern von offensichtlichen Beziehungen	56
5.7	Visualisierung	56
5.8	OntoTrack Anbindung	57
5.9	Erkenntnisse	58
6	Bewertung und Ausblick	61
6.1	Allgemeine Vorgehensweise	61
6.2	Implementierung	62
6.3	Ähnliche Projekte	63
	Literaturverzeichnis	66

Abbildungsverzeichnis

1.1	Beispiel für eine Familienontologie	2
2.1	Syntax und Semantik der Beschreibungslogik <i>SHIF</i>	6
3.1	Beispiel für einen Tableau	11
3.2	Regeln zur Umformung eines Terms in NNF	12
3.3	Tableau-Erweiterungsregeln für die Sprache <i>SHIF</i> (OWL-Lite)	13
3.4	Übertragung der Unerfüllbarkeit auf den Elternknoten	14
3.5	Unendliche Expansion durch zyklische Konzeptdefinition	14
3.6	Beispiel für Paarweises Blocking	15
3.7	Nachträgliche Beeinflussung eines Knotens über eine inverse Rolle	16
3.8	Lazy-Unfolding	17
4.1	Trennung von linker und rechter Seite	21
4.2	Veränderungen im Tableau durch domain- und range-Restriktionen	27
4.3	Einfaches Merging	28
4.4	Erweiterung einer \forall -Restriktion für eine transitive Relation	30
4.5	Erweiterung einer \forall -Restriktion für eine inverse Relation	32
4.6	Merging mit dem Vorgängerknoten über eine inverse Rolle	33
4.7	Komplexes Beispiel 1	37
4.8	Komplexes Beispiel 2	38
4.9	Komplexes Beispiel 4	39
4.10	Komplexes Beispiel 3 - Tableau	40
4.11	Komplexes Beispiel 3 - Erklärung	41
4.12	Komplexes Beispiel 5 - Tableau	42
4.13	Komplexes Beispiel 5 - Erklärung	43
4.14	Subsumtionserklärung bei Unerfüllbarkeit auf einer Seite des Tableaus	45
4.15	Unerfüllbarkeitserklärung für eine Seite des Tableaus	46
5.1	Repräsentation der T-Box durch drei Hash-Tabellen	50
5.2	Repräsentation einer Rollendefinition	51
5.3	Repräsentation eines Constraints	51
5.4	Repräsentation eines Erklärungsschritts	52
5.5	Screenshot einer visualisierten Erklärung	57
5.6	Screenshot einer Erklärungsanfrage in OntoTrack	58

1 Einleitung

1.1 Semantic Web

Das *World Wide Web* zählt ohne Zweifel zu den wichtigsten und erfolgreichsten Entwicklungen der IT-Branche. Kein anderes Medium ermöglicht den gezielten Zugriff auf eine so umfangreiche Menge von Informationen. Diese, im Nachfolgenden auch als Wissen bezeichneten Informationen, sind in einer Form hinterlegt, die nur vom Menschen vollständig interpretiert werden kann. Die Daten können zwar auf syntaktischer Ebene von Maschinen, typischerweise Computerprogrammen, gelesen, durchsucht und modifiziert werden, ihre Semantik zu verstehen ist Maschinen jedoch nicht möglich.

Die Initiative *Semantic Web* versucht, genau diese Lücke zu schließen. Ziel ist es, eine neue Generation des Webs zu schaffen, bei welcher die Informationen in von Maschinen interpretierbarer Form hinterlegt werden. Die maschinelle Interpretation der in diesem Web hinterlegten Daten ermöglicht eine Vielzahl neuartiger Applikationen, die im heutigen WWW nicht realisierbar sind.

Eine Suchmaschine im Semantic Web kann nicht nur nach dem Vorkommen des gewünschten Begriffs suchen, sondern gleichzeitig semantische Informationen berücksichtigen. Würde beispielsweise nach einem „Kombi“ gesucht, so könnte aus einer Automobil-Ontologie¹ gelesen werden, dass Kombi ein Oberbegriff für die Ausdrücke T-Modell, Caravan, Variant, Avant usw. ist. Durch das Berücksichtigen dieser zusätzlichen Information wird die Suche genauer und findet ebenfalls relevante Informationen, in welchen der Begriff Kombi nicht explizit vorkommt.

Die Visionen darüber, was durch das Semantic Web erreicht werden kann, gehen jedoch viel weiter. Denkbar sind Agentensysteme, die selbstständig Wissen über benötigte Bereiche aus dem Netz akquirieren und dieses zur Lösung von komplexen Aufgaben verwenden. Ein beliebtes Beispiel ist die Vision eines Urlaubsplaners, der nach Angabe von groben Vorgaben selbstständig einen kompletten Urlaub planen kann.

Noch einen Schritt weiter geht die Überlegung, Systeme untereinander Wissen austauschen zu lassen. So könnten beobachtende Agenten ihre Beobachtungen im Netz hinterlegen und andere Systeme dieses Wissen auslesen und weiterverarbeiten. Ein Beispiel für eine solche Anwendung wäre ein System, bei welchem Fahrzeuge den Straßenzustand analysieren und im Netz hinterlegen und andere Fahrzeuge durch Auswertung dieses Wissens ihre Fahrer frühzeitig warnen können. Allerdings sind Anwendungen dieser Größenordnung reine Zukunftsmusik. Das Semantic Web steht gerade erst am Anfang seiner Entwicklung.

¹die Bedeutung des Begriffs Ontologie wird in Abschnitt 1.2 beschrieben

Wie das World Wide Web ist auch das Semantic Web als anarchisch organisierte Plattform des Datenaustauschs konzipiert. Das heißt, dass jeder das dort hinterlegte Wissen benutzen oder eigene Informationen hinzufügen kann. Die als Ontologien bezeichneten Wissensbereiche werden im Semantic Web in der vom W3C spezifizierten *Web Ontology Language*² repräsentiert.

1.2 Ontologien

Ursprünglich bezeichnet der Begriff *Ontologie* die Wissenschaft des Seins. Daraus hat sich der im Bereich der Wissensrepräsentation verwendete Begriff einer Ontologie entwickelt. Als Ontologie wird hier die Definition von Beschreibungen für einen Weltausschnitt bezeichnet. Typischerweise wird das Wissen über diesen Ausschnitt über eine Menge von Konzepten und deren Relationen zueinander beschrieben. Es wird das nötige Vokabular und dessen semantische Bedeutung definiert. Dadurch werden implizit auch Regeln für das repräsentierte Wissen definiert.

Eine Ontologie für Familienbeziehungen würde beispielsweise Konzepte wie Person, Mann, Frau, Vater, Mutter, Großmutter, Onkel oder Schwager definieren. Des Weiteren müssten Relationen wie hat-kind, verheiratet-mit, hat-bruder und hat-schwester enthalten sein.

Die in einer Ontologie definierten Konzepte bilden typischerweise eine Hierarchie. Abbildung 1.1 zeigt eine solche Konzepthierarchie am Beispiel der Familienontologie. Das Konzept Großmutter ist ein Unterkonzept von Mutter und Mutter ein Unterkonzept von Frau. Auch Relationen können hierarchisch angeordnet sein. So ist hat-sohn zum Beispiel eine Subrelation von hat-kind. Die Inklusionsbeziehungen zwischen Konzepten können entweder explizit definiert worden sein oder durch komplexe Konzeptdefinitionen entstehen.

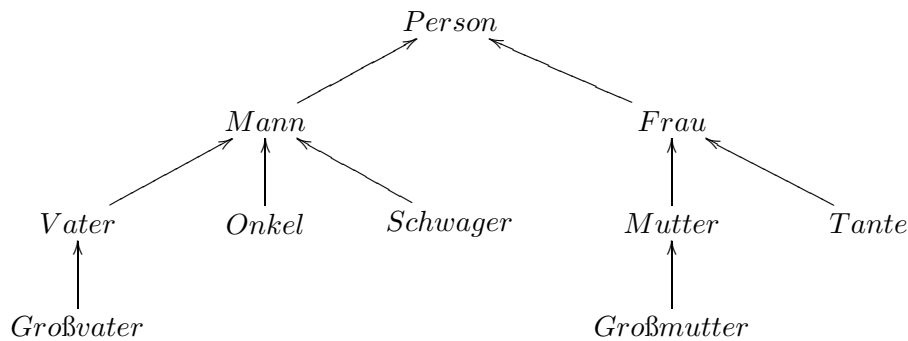


Abbildung 1.1: Beispiel für eine Familienontologie

Das in Abschnitt 1.1 beschriebene Semantic Web kann als eine Ansammlung von Ontologien betrachtet werden. Durch Schlussfolgerung über Wissen aus mehreren Ontologien kann implizites Wissen aufgedeckt werden. Eine

²die *Web Ontology Language* wird im Kapitel Grundlagen in Abschnitt 2.2 beschrieben

Schlussfolgerung ist eine im Rahmen eines Beweises hergeleitete Aussage. Der Prozess des Schlussfolgerns basiert auf logischen Grundsätzen. Diese können bei geeigneter Repräsentation des Wissens auch von Computern angewandt werden. Dadurch entsteht eine maschinelle Form des Beweises. Implizites Wissen kann aus großen Mengen von Wissen maschinell abgeleitet werden.

1.3 Erklären

Die maschinelle Ableitung von impliziten Informationen basiert auf komplizierten Techniken. In Abschnitt 2.4 werden einige Algorithmen dazu vorgestellt. Diese bekommen eine Anfrage vom Benutzer, kombinieren und analysieren das vorhandene Wissen und liefern daraufhin ein Ergebnis. Für viele Problemstellungen ist es jedoch nicht uninteressant für den Benutzer, zu erfahren, wie das Ergebnis zustande gekommen ist.

Beispielsweise liegt es nahe, beim Erstellen von Ontologien einen Inferenzdienst zu benutzen, der die neu eingefügten Konzepte direkt in die bisherige Hierarchie einordnet. Kleine Modellierungsfehler können dabei unerwartete Auswirkungen auf die Hierarchie haben. Dem Benutzer stellt sich nun die Frage, wo er einen solchen Fehler zu suchen hat. Bei großen Ontologien kann dies äußerst kompliziert sein. Eine falsche Klassifizierung kann zum Beispiel durch eine Modellierungsungenauigkeit in einem anderen Konzept hervorgehoben werden.

Um den Benutzer bei diesem und ähnlichen Problemen zu unterstützen, ist es hilfreich, den internen Vorgang, der zum vorliegenden Ergebnis geführt hat, transparent zu machen. Das Erklären (*Explaining*) von Inferenzdiensten verfolgt dieses Ziel. Ein Inferenzmechanismus wird so gestaltet, dass er nicht nur ein Ergebnis liefert, sondern gleichzeitig erklärt, wie er zu diesem gekommen ist. Für die Form der Erklärung sind verschiedene Ansätze denkbar. Zunächst muss die Erklärung auf die zu beantwortende Fragestellung zugeschnitten sein. Des Weiteren ist zu beachten, für welche Art von Benutzern die Erklärung generiert wird. Ein Experte wird eine technisch orientierte, knappe Erklärung vorziehen, wohingegen ein unerfahrener Benutzer eine sehr ausführliche, einfach formulierte Erklärung benötigt, um den Vorgang zu verstehen.

In dieser Diplomarbeit wird eine Möglichkeit entwickelt, die in Abschnitt 2.3 beschriebenen Unerfüllbarkeitsbeweise für Subsumtionsanfragen zu erklären. Dazu wird eine schrittweise Erklärung aufgebaut. Die einzelnen Schritte liefern natürlichsprachliche Erklärungen. Dadurch können auch unerfahrene Benutzer verstehen, wie der Algorithmus zu seinem Ergebnis gekommen ist.

1.4 Überblick

Diese Arbeit beschäftigt sich mit der Erklärung von tableaubasierten Subsumtionsbeweisen.

In Kapitel 2 werden die zum weiteren Verständnis notwendigen Grundlagen

beschrieben. Es wird ein Überblick über die Sprachfamilie der Beschreibungslogiken und die Semantic Web Sprache OWL gegeben. Des weiteren werden typische Inferenzdienste und die für diese verwendeten Schlussfolgerungsalgorithmen dargestellt. Abschließend wird auf existierende Inferenzsysteme eingegangen.

Kapitel 3 behandelt das Tableau-Verfahren für die Beschreibungslogik *SHIF*. Es wird das grundsätzliche Vorgehen des Algorithmus, sowie einige Verfahren zur Optimierung und Gewährleistung der Terminierung beschrieben. Das Tableau-Verfahren stellt die Grundlage für den in Kapitel 4 beschriebenen Erklärungsalgorithmus dar.

Ein Verfahren zur Generierung von Erklärungen für tableaubasierte Subsumtionsbeweise wird in Kapitel 4 beschrieben. Dieses wurde im Rahmen dieser Arbeit entwickelt. Es werden die generelle Vorgehensweise sowie die konkreten Erklärungsschritte dargestellt. Anschließend werden konkrete Beispiele für generierte Erklärungen geliefert. Das Kapitel schließt mit einer Diskussion über mögliche Verbesserungsansätze für das Generieren von Erklärungen.

In Kapitel 5 wird die Implementierung einer Erklärungskomponente für Subsumtionsbeweise beschrieben. Dabei werden zunächst einige technische Aspekte dargestellt. Des weiteren wird auf die durch das Programm erklärbaaren Konstrukte, sowie die dazu notwendigen Optimierungen eingegangen. Abschließend werden die durch die Implementierung gewonnenen Erkenntnisse diskutiert.

Die Arbeit schließt mit Kapitel 6. In diesem werden die erarbeiteten Erkenntnisse bewertet. Dies umfasst sowohl den prinzipiellen Ansatz, als auch die prototypische Implementierung. Abschließend wird auf einige Projekte eingegangen, welche sich ebenfalls mit der Generierung von Erklärungen im Bereich der Beschreibungslogik beschäftigen.

2 Grundlagen

Der in dieser Arbeit beschriebene Ansatz baut auf verschiedenen Techniken und Formalismen aus dem Bereich der künstlichen Intelligenz und deren Anwendung im Kontext des Semantic Web auf, welche in diesem Kapitel beschrieben werden. Es wird eine kurze Einführung zum Sprachformalismus der Beschreibungslogik und der Semantic Web Sprache OWL gegeben. Des Weiteren werden die wichtigsten Inferenzdienste, die für diese typischerweise eingesetzten Algorithmen sowie einige existierende Inferenzsysteme (Reasoner) behandelt.

2.1 Beschreibungslogik

Beschreibungslogiken (*Description Logics*) sind eine Familie von Sprachformalismen zur Repräsentation von semantischen Sachverhalten. Sie basieren auf der Definition von Konzepten, häufig auch Klassen genannt, und binären Relationen, so genannten Rollen. Eine Beschreibungslogik ist über eine Menge von Konstruktoren definiert. Diese ermöglichen die Definition von komplexen Konzepten und Rollen. Je mehr Konstruktoren eine Beschreibungslogik zulässt, desto umfangreicher ist der durch sie beschreibbare Sprachumfang. Mit dem Sprachumfang steigt auch die Komplexität der Inferenzdienste¹ für eine Beschreibungslogik.

In dieser Arbeit werden für beschreibungslogische Ausdrücke folgende Konventionen verwendet:

A, B	für atomare Konzepte
C, D, E	für komplexe Konzeptausdrücke
r, s, t	für Relationen
f	für funktionale Relationen
$subR$	für Subrelation der Relation r ($subR \sqsubseteq r$)
$invR$	für die Inverse der Relation r

Es hat sich die Notation etabliert, Beschreibungslogiken durch eine Buchstabenfolge zu benennen, bei welcher die einzelnen Buchstaben für erlaubte Konstruktoren stehen. Eine sehr bekannte Beschreibungslogik ist \mathcal{ALC} [SSS91]. \mathcal{ALC} erlaubt die Definition von Konzepten mittels qualifizierter Existenzquantoren ($\exists r.C$), qualifizierter Allquantoren ($\forall r.C$), Konjunktion ($C \sqcap D$), Disjunktion ($C \sqcup D$) und Negation ($\neg C$) sowie die Definition von einfachen

¹Inferenzdienste werden in Abschnitt 2.3 beschrieben

Rollen. Viele komplexere Beschreibungslogiken bauen auf \mathcal{ALC} auf. So wird in [HST99] die Sprache \mathcal{S} als eine Erweiterung von \mathcal{ALC} um *transitive Rollen* definiert. Werden zusätzlich *inverse Rollen* zugelassen, so wird die Sprache \mathcal{SHI} genannt. Die Möglichkeit, *Rollenhierarchien* der Form $r \sqsubseteq s$ zu definieren, wird durch ein zusätzliches \mathcal{H} gekennzeichnet. \mathcal{SHI} ist also die Sprache \mathcal{ALC} erweitert um transitive und inverse Rollen sowie Rollenhierarchien.

Wird \mathcal{SHI} um qualifizierte Kardinalitätseinschränkungen² erweitert, so wird ein \mathcal{Q} hinzugefügt. Diese Einschränkungen der Form $\leq n r.C$ oder $\geq n r.C$ führen zu einer hohen Komplexität der Inferenzdienste für \mathcal{SHIQ} . Weniger problematisch sind unqualifizierte Kardinalitätseinschränkungen ($\leq n r$ oder $\geq n r$). \mathcal{SHI} um diese erweitert wird \mathcal{SHIN} genannt. Eine noch stärkere Einschränkung der Sprache \mathcal{SHIQ} ist \mathcal{SHIF} . \mathcal{SHIF} erlaubt nur funktionale unqualifizierte Kardinalitätseinschränkungen. Dies sind unqualifizierte Einschränkungen auf die Kardinalitäten 0 oder 1. Die in dieser Arbeit entwickelten Erkenntnisse beziehen sich auf den Sprachumfang von \mathcal{SHIF} . Genauere Definitionen zur Familie der \mathcal{S} -Beschreibungslogiken sind in [HST99] zu finden.

Eine *Interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ besteht aus einer Menge $\Delta^{\mathcal{I}}$ und einer Funktion $\cdot^{\mathcal{I}}$, welche jedem Konzept eine Untermenge von $\Delta^{\mathcal{I}}$ und jeder Rolle eine Untermenge von $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ zuordnet. Eine Interpretation \mathcal{I} wird *Modell* für eine Terminologie \mathcal{T} genannt, genau dann wenn $C_i^{\mathcal{I}} \subseteq D_i^{\mathcal{I}}$ für alle $C_i \sqsubseteq D_i \in \mathcal{T}$ gilt.

Abbildung 2.1 zeigt die Syntax sowie die jeweilige semantische Bedeutung der in der Sprache \mathcal{SHIF} zugelassenen Sprachkonstrukte.

Konstrukt	Syntax	Semantik
allgemeinstes Konzept	\top	$\Delta^{\mathcal{I}}$
speziellstes Konzept	\perp	\emptyset
atomares Konzept	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
atomare Rolle	r	$r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
transitive Rolle	$r \in r_+$	$r^{\mathcal{I}} = (r^{\mathcal{I}})^+$
Konjunktion	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
Disjunktion	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
Negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
Existenzquantor	$\exists r.C$	$\{x \mid \exists y. \langle x, y \rangle \in r^{\mathcal{I}} \text{ und } y \in C^{\mathcal{I}}\}$
Allquantor	$\forall r.C$	$\{x \mid \forall y. \langle x, y \rangle \in r^{\mathcal{I}} \implies y \in C^{\mathcal{I}}\}$
Rollenhierarchie	$r \sqsubseteq s$	$r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$
inverse Rolle	r^-	$\{\langle x, y \rangle \mid \langle y, x \rangle \in r^{\mathcal{I}}\}$
funktionale unqualifizierte Kardinalitäts- einschränkungen	$(\geq n r)$ $(\leq n r)$ mit $n \in [0, 1]$	$\{x \mid \#\{y. \langle x, y \rangle \in r^{\mathcal{I}}\} \geq n\}$ $\{x \mid \#\{y. \langle x, y \rangle \in r^{\mathcal{I}}\} \leq n\}$ mit $n \in [0, 1]$

Abbildung 2.1: Syntax und Semantik der Beschreibungslogik \mathcal{SHIF}

²der Effekt qualifizierter Kardinalitätseinschränkungen kann teilweise durch die Kombination von Sub-Relationen und range-Einschränkungen simuliert werden

Weitere Informationen zu Beschreibungslogiken im Allgemeinen finden sich in [BCM⁺03].

2.2 Web Ontology Language (OWL)

Die vom W3C entwickelte *Web Ontology Language* (kurz OWL) ist eine neue Sprache für das Semantic Web. OWL stellt eine Weiterentwicklung der Sprache DAML+OIL³ dar. Sie baut auf RDF⁴ und RDFS⁵ auf, basiert jedoch auf einem formalen Hintergrund. OWL basiert syntaktisch auf XML und RDF, semantisch auf Beschreibungslogiken. Letzteres ermöglicht die maschinelle Verarbeitung der in OWL repräsentierten Informationen durch die Verwendung existierender Algorithmen für Beschreibungslogiken. Insbesondere kann der in 2.4.2 beschriebene Tableau-Algorithmus für das automatische Schlussfolgern über OWL Ontologien benutzt werden. OWL wurde in drei unterschiedlich umfangreiche Sprachebenen unterteilt.

OWL Lite stellt eine sehr eingeschränkte Untermenge des kompletten Sprachumfangs dar. OWL Lite wurde entwickelt, um einen vereinfachten Einstieg in das Erstellen von Ontologien und die Entwicklung von Inferenzdiensten⁶ zu ermöglichen. Semantisch repräsentiert OWL Lite in etwa den Sprachumfang der Beschreibungslogik *SHIF*.

OWL DL (*DL* steht für *Description Logic*) entspricht in etwa dem Sprachumfang der Beschreibungslogik *SHOIN*. OWL DL stellt somit die maximale Ausdrucksmächtigkeit, für welche alle Schlussfolgerungen noch entscheidbar sind, zur Verfügung. Inferenzdienste auf OWL DL haben jedoch nicht-deterministische exponentielle Komplexität. Bisher gibt es keinen vollständigen Reasoner⁷ für OWL DL.

OWL Full ermöglicht eine über Beschreibungslogik hinausgehende Ausdrucksmächtigkeit, verliert dadurch jedoch die volle Entscheidbarkeit für Schlussfolgerungen. OWL Full wurde entwickelt, um volle Kompatibilität zu RDF und RDFS zu gewährleisten.

Jede dieser Sprachebenen stellt eine Untermenge der nächst höheren Ebene dar. Eine ausführliche Beschreibung der Entwicklung von OWL und den dabei berücksichtigten Kriterien findet sich in [HPSvH03]. Für diese Arbeit ist besonders OWL Lite relevant, da die entwickelten Techniken sich auf den Sprachumfang von OWL Lite beschränken.

³<http://www.daml.org/>

⁴<http://www.w3.org/RDF/>

⁵<http://www.w3.org/TR/rdf-schema/>

⁶typische Inferenzdienste werden in 2.3 beschrieben

⁷einige aktuelle Reasoner werden in Abschnitt 2.5 vorgestellt

2.3 Inferenzdienste

Systeme, welche maschinelle Ableitungsverfahren implementieren werden Inferenz- oder Schlussfolgerungsdienste genannt. Die meisten komplexen Inferenzdienste lassen sich auf systematische Anwendung einiger Basis-Inferenzdienste zurückführen. Im Folgenden werden zwei dieser Basis-Inferenzdienste beschrieben. Diese stellen die Grundlage für die meisten Inferenzdienste dar.

2.3.1 Subsumtion

Subsumtion ist die hierarchische Einordnung eines Konzepts unter ein anderes. Ein Konzept A (*Subsumee*) wird also von einem Konzept B (*Subsumer*) subsumiert (geschrieben $A \sqsubseteq B$), wenn A eine Untermenge von B darstellt. Vater ist beispielsweise eine Untermenge von Mann, da jeder Vater auch ein Mann ist. Das Konzept Vater wird also vom Konzept Mann subsumiert. Eine Subsumtion stellt in Gegenrichtung eine Implikation dar. Ist ein Individuum vom Typ Vater, so muss es zwingenderweise auch vom Typ Mann sein. Subsumtionsbeziehungen können implizit durch die Definition der Konzepte folgen. Ein typischer Inferenzdienst ist es, solche Subsumtionsbeziehungen zu finden. Dies geschieht, indem potentielle Subsumtionsbeziehungen auf ihre Gültigkeit geprüft werden.

Subsumtionstests werden unter anderem dazu benutzt, eine Konzepthierarchie (siehe Abb. 1.1) zu erstellen. Dazu werden alle definierten Konzepte auf Subsumtionsbeziehungen untereinander überprüft.

2.3.2 Unerfüllbarkeit

Ein weiterer wichtiger Inferenzdienst ist die Überprüfung eines Konzepts oder Terms auf Unerfüllbarkeit. Dies geschieht, indem in der Konzeptdefinition bzw. dem Term nach einem Widerspruch gesucht wird. Wird ein solcher gefunden, so ist das Konzept unerfüllbar. Ein Subsumtionstest kann zu einem Unerfüllbarkeitstest umgewandelt werden. Dabei gilt folgende Äquivalenz:

$$A \sqsubseteq B \iff A \sqcap \neg B \text{ ist unerfüllbar}$$

Dies ermöglicht es, einen Unerfüllbarkeitsbeweis, zum Beispiel basierend auf dem in Abschnitt 2.4.2 beschriebenen Tableau-Verfahren, für Subsumtionstests zu verwenden.

2.4 Schlussfolgerungsalgorithmen

2.4.1 Strukturelle Subsumtion

Die Methode des strukturellen Subsumtionstest beruht auf einem strukturellen Vergleich von Subsumer und Subsumee. Zunächst werden die beiden Konzeptdefinitionen vollständig aufgefaltet, das heißt, alle komplex definier-

ten Konzepte werden rekursiv durch ihre Definitionen ersetzt. Die daraus resultierenden Terme werden in eine Normalform transformiert und redundante Information wird entfernt. Nun folgt der eigentliche Vergleich. Für jedes Konstrukt im Subsumee wird ein Konstrukt im Subsumer gesucht, welches dieses subsumiert. Ist dies erfolgreich, so gilt die Beziehung $Subsumee \sqsubseteq Subsumer$.

Für Sprachen, die kompliziertere Konstruktoren, wie zum Beispiel inverse Rollen zulassen, sind strukturelle Beweisverfahren nicht definiert. Für solche ausdrucksmächtigere Sprachen verlieren die strukturellen Beweisverfahren die Eigenschaft der Vollständigkeit. Aus diesem Grund werden sie in heutigen Systemen kaum noch eingesetzt. Vielmehr hat sich der im nächsten Abschnitt beschriebene Tableau-Algorithmus zum Standard für Inferenzdienste entwickelt.

2.4.2 Tableau-Algorithmus

Tableaubasierte Algorithmen versuchen, ein Modell für einen logischen Ausdruck zu konstruieren. Wird dabei ein Widerspruch, ein so genannter *Clash*, gefunden, ist der Ausdruck unerfüllbar. Wie in Abschnitt 2.3.2 beschrieben, kann dieses Verfahren auch für Subsumtionstests benutzt werden. Die Subsumtionsbeziehung wird dazu negiert und in eine Normalform umgewandelt. Kann dieser Normalform Unerfüllbarkeit nachgewiesen werden, so gilt die ursprüngliche Subsumtionsbeziehung. Der konkrete Ablauf eines Tableau-Beweises wird in Abschnitt 3 für die Beschreibungslogik *SHIF* beschrieben.

Das Tableau-Verfahren hat sich zum Standardverfahren für beschreibungslogische Inferenzdienste entwickelt.

2.5 Inferenzsysteme

Seit fast 20 Jahren werden Inferenzsysteme für Beschreibungslogiken entwickelt. Die älteren Systeme basieren meist auf strukturellen Subsumtionsalgorithmen. Das wohl bekannteste dieser Systeme ist LOOM [Gre88], welches an der University of Southern California entwickelt wurde. Mit der Erweiterung beschreibungslogischer Sprachen um komplexere Konstrukte, wie zum Beispiel inverse Rollen, können auf strukturellen Algorithmen basierende Systeme jedoch nicht umgehen. Aus diesem Grund basieren neuere Systeme auf dem in 2.4.2 beschriebenen Tableau-Algorithmus. Das erste relevante tableaubasierte System für Beschreibungslogiken war das von Ian Horrocks entwickelte FaCT [Hor98]. Lange Zeit galt FaCT als das Standard Inferenzsystem für Beschreibungslogik. In den letzten Jahren haben sich jedoch aufgrund besserer Performance und umfangreicherer Funktionalität die in den nächsten Abschnitten beschriebenen Systeme RACER und Pellet durchgesetzt.

2.5.1 RACER

Der zur Zeit erfolgreichste Reasoner für Beschreibungslogiken ist RACER. RACER wird seit 1999 von Ralf Möller und Volker Haarslev an der TU Hamburg-

Harburg entwickelt. Das System ist in Lisp implementiert und basiert auf dem in 2.4.2 beschriebenen Tableau-Verfahren. Durch die Anwendung verschiedener Optimierungsstrategien erreicht RACER eine sehr hohe Performance. Weitere Informationen zu RACER und dem von RACER behandelbaren Sprachumfang finden sich in [HM01].

2.5.2 Pellet

Pellet ist ein relativ neuer Reasoner für OWL DL. Pellet wird an der University of Maryland von der MINDSWAP Research Group entwickelt. Pellet ist als ein open-source Projekt in Java implementiert. In aktuellen Tests hat sich gezeigt, dass Pellet bereits fast so schnell und zuverlässig wie RACER ist. Pellet wurde speziell für Inferenzdienste im Semantic Web entwickelt. Das System ist also auf Reasoning für OWL DL zugeschnitten. Das Pellet System wird auf der Webseite der MINDSWAP Research Group [Min] genauer beschrieben.

3 Tableau-Verfahren für $SHIF$

Dieses Kapitel beschreibt das Tableau-Verfahren für die Beschreibungslogik $SHIF$. Es wird zunächst die allgemeine Vorgehensweise dargestellt. Dies umfasst die Tableau-Erweiterungsregeln und die Suche nach einem Clash. Abschnitt 3.2 beschreibt die Technik des Blockings, welche notwendig ist, um zu garantieren, dass der Algorithmus terminiert. Abschließend wird in Kapitel 3.3 die Optimierungsstrategie des Lazy-Unfolding vorgestellt.

3.1 Grundlagen des Tableau-Verfahrens

Das heutzutage bevorzugte Inferenzverfahren, um Subsumtionen zu prüfen, ist das Tableau-Verfahren. Bei diesem wird die Anfrage über die zu beweisende Subsumtionsbeziehung in ein äquivalentes Erfüllbarkeitsproblem umgewandelt.

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

Der Tableau-Algorithmus versucht nun, ein Modell für die negierte Anfrage zu finden. Gelingt dies, wurde somit ein Gegenbeispiel für die Anfrage gefunden, die Subsumtionsbeziehung gilt also nicht. Kann jedoch kein Modell gefunden werden, so ist der Ausdruck unerfüllbar und dessen Negation, die ursprünglich gesuchte Subsumtionsbeziehung, somit gültig.

Die Suche nach einem Modell für den beschreibungslogischen Ausdruck verläuft über den Aufbau eines so genannten Tableaus. Ein solcher Tableau wird zur Veranschaulichung zumeist als Baum dargestellt. Er besteht aus Knoten, welche beschreibungslogische Ausdrücke enthalten und Kanten die Relationsbeziehungen zu weiteren Knoten repräsentieren (Abb. 3.1).

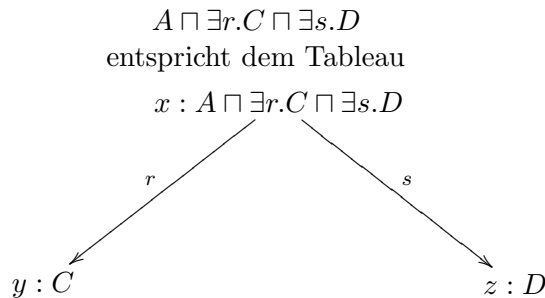


Abbildung 3.1: Beispiel für einen Tableau

Die negierte Anfrage wird zunächst in Negationsnormalform (kurz NNF) umgewandelt, das heißt Negationen stehen nur noch direkt vor Konzepten. Dies geschieht durch Anwendung der in Abbildung 3.2 dargestellten Äquivalenzen.

$$\begin{aligned}\neg(C \sqcup D) &\equiv \neg C \sqcap \neg D \\ \neg(C \sqcap D) &\equiv \neg C \sqcup \neg D \\ \neg(\exists r.C) &\equiv \forall r.\neg C \\ \neg(\forall r.C) &\equiv \exists r.\neg C \\ \neg(\geq n C) &\equiv (\leq (n-1) C) \\ \neg(\leq n C) &\equiv (\geq (n+1) C)\end{aligned}$$

Abbildung 3.2: Regeln zur Umformung eines Terms in NNF

Der Tableau wird, ausgehend vom initialen Ausdruck in NNF, durch die Anwendung bestimmter Erweiterungsregeln erstellt. Für jedes zugelassene Sprachkonstrukt muss es eine Regel geben, die beschreibt wie der Algorithmus dieses Konstrukt behandelt. Wichtig ist, dass alle Regeln konsistenzhaltend sind, das heißt, dass der aus einer Erweiterung resultierende Ausdruck genau dann unerfüllbar ist, wenn der erweiterte Ausdruck unerfüllbar ist. Abbildung 3.3 zeigt die Tableau-Erweiterungsregeln für die Beschreibungslogik \mathcal{SHIF} . Dabei bezeichnet $\mathcal{L}(x)$ die Menge der dem Knoten x zugeordneten Konzepte und $\langle x, y \rangle$ eine die Knoten x und y verbindende Kante. Ein Knoten y wird r -Nachfolger des Knotens x genannt, wenn y ein Nachfolger von x ist und $s \in \mathcal{L}(\langle x, y \rangle)$ für ein $s \sqsubseteq r$. Ein Knoten y wird r -Nachbar des Knotens x genannt, wenn y ein r -Nachfolger von x ist oder x ein $\text{Inv}(r)$ -Nachfolger von y ist.

Konstrukte, die die Existenz eines einzigen oder einer Anzahl von bestimmten Rollenfüllern fordern, wie zum Beispiel $\exists r.C$, fügen dem Tableau neue, über Kanten referenzierte Knoten hinzu. Des weiteren können Qualifikationen für die in Relation stehenden Knoten gefordert werden, zum Beispiel durch ein Konstrukt der Form $\forall r.C$. Diese beeinflussen den Tableau nur, falls bereits eine Kante für die entsprechende Relation existiert. Grundsätzlich wird zwischen generierenden und nicht-generierenden Regeln unterschieden. Generierend werden Regeln genannt, welche dem Tableau neue Kanten hinzufügen. Dazu gehören die \exists -Regel und die \geq -Regel. Die Reihenfolge, in welcher die Erweiterungsregeln angewendet werden, ist nicht beliebig. Generierende Regeln müssen immer vor den nicht-generierenden angewendet werden, da sie eventuell Knoten erzeugen, welche von den nicht-generierenden Regeln nachfolgend verändert werden.

Die Unerfüllbarkeit eines Knotens lässt sich durch Widersprüche, so genannte Clashes, erkennen. Ein Knoten enthält einen Clash, falls ein Konzept und dessen Negation in ihm enthalten sind. Ein Vorkommen des speziellsten Konzepts Bottom wird ebenfalls als Clash interpretiert.

$$\{C, \neg C\} \subseteq \mathcal{L}(x) \quad \text{oder} \quad \{\perp\} \subseteq \mathcal{L}(x)$$

Eine weitere Möglichkeit für einen Clash ist die, dass zum einen mehr als n und zum anderen weniger als m ($m, n \in \mathbb{N}$) Füller für eine bestimmte Relation

Regel	Bedingung	Behandlung
\sqcap -Regel	$C_1 \sqcap C_2 \in \mathcal{L}(x)$ und $\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$	$\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{C_1, C_2\}$
\sqcup -Regel	$C_1 \sqcup C_2 \in \mathcal{L}(x)$ und $\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$	für ein $C \in \{C_1, C_2\}$, $\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{C\}$
\exists -Regel	$\exists s.C$ und x hat keinen s-Nachbar y mit $C \in \mathcal{L}(y)$	füge neuen Knoten y mit $\mathcal{L}(\langle x, y \rangle) = \{s\}$ und $\mathcal{L}(y) = \{C\}$ hinzu
\forall -Regel	$\forall s.C$ und x hat einen s-Nachbar y mit $C \notin \mathcal{L}(y)$	$\mathcal{L}(y) \longrightarrow \mathcal{L}(y) \cup \{C\}$
\forall_+ -Regel	$\forall s.C$, es gibt ein r mit $\text{Trans}(r)$ und $r \sqsubseteq s$ und x hat einen r -Nachbar y mit $\forall r.C \notin \mathcal{L}(y)$	$\mathcal{L}(y) \longrightarrow \mathcal{L}(y) \cup \{\forall r.C\}$
\geq -Regel	$(\geq 2 r) \in \mathcal{L}(x)$ und x hat kei- nen r -Nachbar y mit $A \in \mathcal{L}(y)$	füge 2 neue Knoten y_1, y_2 mit $\mathcal{L}(\langle x, y_1 \rangle) = \mathcal{L}(\langle x, y_2 \rangle) = \{r\}$, $\mathcal{L}(y_1) = \{A\}$ und $\mathcal{L}(y_2) = \{A\}$ hinzu
\leq -regel	$(\leq 1 r) \in \mathcal{L}(x)$ und x hat zwei r -Nachbarn y und z , wobei y kein Vorfahre von z ist	1. $\mathcal{L}(z) \longrightarrow \mathcal{L}(z) \cup \mathcal{L}(y)$ 2. falls z ein Vorfahre von y ist $\mathcal{L}(\langle z, x \rangle) \rightarrow \mathcal{L}(\langle z, x \rangle) \cup$ $\text{Inv}(\mathcal{L}(\langle x, y \rangle))$ andernfalls $\mathcal{L}(\langle x, z \rangle) \rightarrow \mathcal{L}(\langle x, z \rangle) \cup \mathcal{L}(\langle x, y \rangle)$ 3. $\mathcal{L}(\langle x, y \rangle) \longrightarrow \emptyset$

 Abbildung 3.3: Tableau-Erweiterungsregeln für die Sprache *SHIF* (OWL-Lite)

gefordert wurden, wie z.B. in

$$\{(\geq 1 r), (\leq 0 r)\} \subseteq \mathcal{L}(x)$$

Ein Knoten ist ebenfalls unerfüllbar, falls ein konjunktiv referenzierter oder *alle* disjunktiv referenzierten Nachfolger unerfüllbar sind (Abb. 3.4). Somit kann sich die Unerfüllbarkeit eines Knotens auf dessen Vorgänger übertragen. Kann sie bis zum Knoten der ursprünglichen Anfrage propagiert werden, so wurde die Unerfüllbarkeit der Anfrage gezeigt.

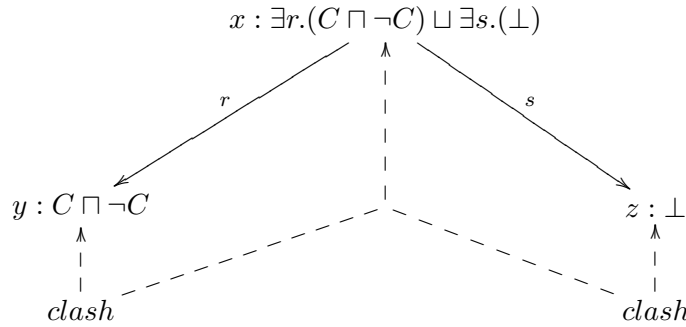


Abbildung 3.4: Übertragung der Unerfüllbarkeit auf den Elternknoten

3.2 Blocking

Einige Sprachkonstrukte führen dazu, dass ein primitiver Tableau-Algorithmus nicht zwingend terminiert. Wird zum Beispiel in der Definition eines Konzeptes ein Relationsfüller von der Art des gleichen Konzeptes gefordert, fügt der Algorithmus in jedem Schritt einen weiteren Knoten hinzu und generiert somit eine endlose Kette gleicher Knoten und Kanten (siehe Abb. 3.5).

$$C \doteq \exists r.C$$

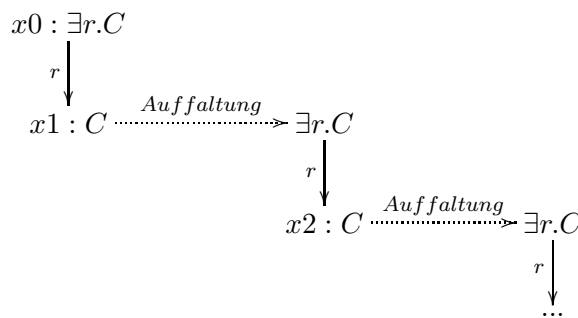


Abbildung 3.5: Unendliche Expansion durch zyklische Konzeptdefinition

Ein ähnlicher Effekt kann durch die Verarbeitung transitiver Rollen entstehen. Um dennoch zu gewährleisten, dass der Algorithmus terminiert, wendet man die Technik des so genannten Blockings an. Dabei werden Knoten, von denen

man weiß, dass ihre Expansion keine zusätzliche Information liefern kann, nicht weiter expandiert. Man sagt der Knoten wird geblockt. Man muss jedoch davon ausgehen, dass der geblockte Knoten erfüllbar ist. Welche Knoten geblockt werden dürfen, hängt wiederum vom Umfang der Sprache ab.

Für *SHIF* muss ein so genanntes paarweises Blocking angewandt werden, um zu gewährleisten, dass keine Knoten geblockt werden, die eventuell noch zu einem Clash führen könnten. Dieses blockt einen Knoten x , der zu seinem direkten Vorgängerknoten y über eine Kante der Relation r verbunden ist, falls er einen beliebigen Vorgängerknoten x' mit $x = x'$ hat, der zu seinem direkten Vorgängerknoten y' mit $y = y'$ ebenfalls über die Relation r verbunden ist (Abb. 3.6). Mit anderen Worten, der Knoten x des Paares $y \xrightarrow{r} x$ wird geblockt, falls bereits ein äquivalentes Vorgängerpaar im Tableau existiert.

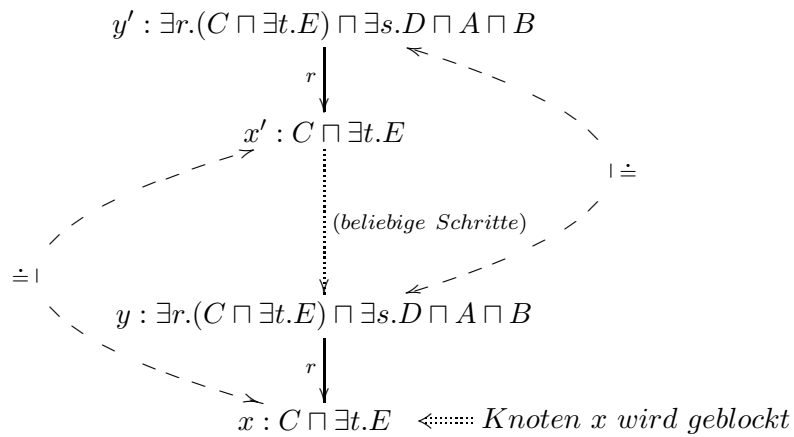


Abbildung 3.6: Beispiel für Paarweises Blocking

Durch die Expansion von inversen Rollen können sich bereits bearbeitete Vorgängerknoten ändern. Dies kann wiederum Einflüsse auf einen bereits geblockten Knoten haben. Insbesondere kann es dazu führen, dass der ursprünglich geblockte Knoten nachträglich unerfüllbar wird. Um diesen Clash jedoch erkennen zu können, muss der Block aufgehoben werden.

Abbildung 3.7 zeigt wie ein Vorgängerknoten x über eine inverse Rolle, deren Eigenschaft in z gefordert ist, verändert wird. Der Knoten x muss also nach dieser Veränderung neu evaluiert werden. Das neue Konstrukt $\forall r. \neg A$ beeinflusst wiederum einen bereits evaluierten Nachfolgerknoten y . Somit muss auch dieser neu evaluiert werden. In diesem Beispiel führt der nachträgliche Eintrag sogar zu einem Clash. Wäre y vor der Änderung geblockt gewesen, hätte diese bewirken müssen, dass der Block aufgehoben wird und der Knoten somit neu evaluiert werden kann.

Da *SHIF* auch inverse Rollen zulässt, muss das Blocking dynamisch gestaltet werden. Das bedeutet, dass ein einmal geblockter Knoten nicht zwingend für immer geblockt bleiben muss. Wie im vorhergehenden Beispiel demonstriert, kann es durch nachträgliche Veränderungen von Vorgängerknoten notwendig werden, einen Block aufzuheben.

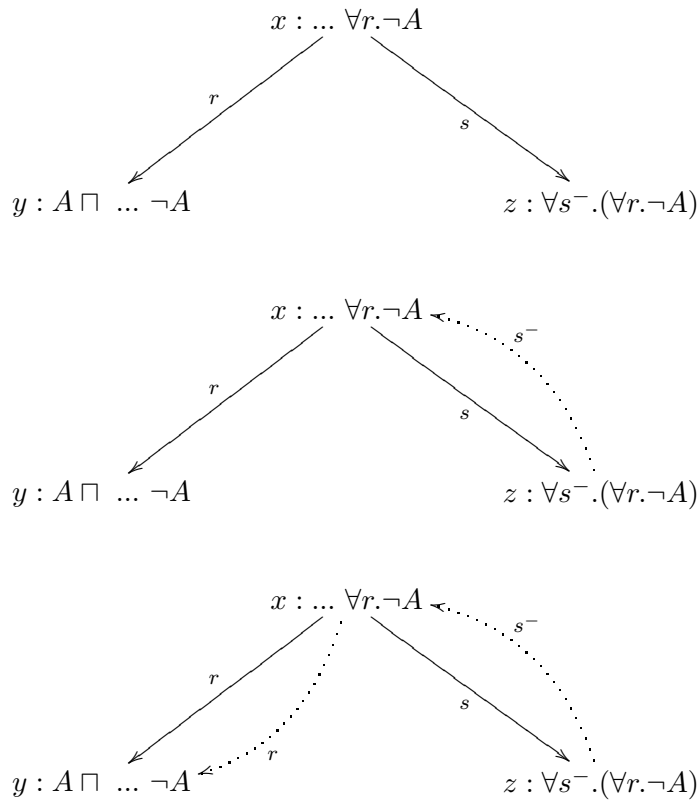


Abbildung 3.7: Nachträgliche Beeinflussung eines Knotens über eine inverse Rolle

3.3 Optimierungen

Der Tableau-Algorithmus kann unter Umständen eine sehr hohe Komplexität haben. Vor allem schwierig zu behandelnde Sprachkonstrukte, wie inverse oder transitive Rollen, erhöhen diese in hohem Maße. Selbst bei relativ einfach erscheinenden Problemstellungen kann es zu enormen Laufzeiten kommen. Um diese dennoch in akzeptablen Grenzen zu halten, empfiehlt es sich, verschiedene Optimierungsstrategien anzuwenden. Im Folgenden wird die Optimierungstechnik des Lazy-Unfolding vorgestellt. Um die Performance des Tableau-Algorithmus zu verbessern existieren viele weitere Optimierungen. In Bezug auf das Erklären von Tableau-Beweisen stellen sich diese jedoch als eher hinderlich dar, da sie zumeist komplizierte, und somit schwer nachvollziehbare, interne Veränderungen vornehmen. Aus diesem Grund wird, abgesehen von Lazy-Unfolding, in dieser Arbeit auf keine weiteren Performanceoptimierungen eingegangen.

3.3.1 Lazy-Unfolding

Die einem Subsumtionsbeweis zugrunde liegenden Konzept- und Relationsdefinitionen werden in realen Anwendungen meist in einer sogenannten T-Box gespeichert. Die initiale Subsumtionsfragstellung enthält also nicht zwin-

gend alle Informationen, die zur Lösungsfindung beitragen können. Vielmehr müssen Konzepte durch ihre Definitionen ersetzt werden, um Widersprüche aufzudecken. Der einfachste Weg, dies zu erreichen, wäre von Anfang an die komplette T-Box aufzufalten, das heißt, alle komplex definierten Konzepte sukzessiv durch ihre Definitionen zu ersetzen. Dieser Vorgang ist jedoch bei großen T-Boxen äußerst aufwändig. Die Technik des *Lazy-Unfolding* umgeht diesen Aufwand, indem jeder Knoten vor seiner Evaluation einzeln aufgefaltet wird. Dabei werden nur Konzepte aufgefaltet, die direkt im Knoten enthalten sind. Über Relationen referenzierte Konzepte werden erst im nächsten Knoten aufgefaltet (siehe Abb. 3.8). Wird ein Clash gefunden, so ist es nicht nötig, alle weiteren, über Relationen referenzierten Knoten zu betrachten. Der Aufwand, diese aufzufalten wurde also eingespart. Vollständig definierte Konzepte werden beim Auffalten durch ihre Definitionen ersetzt. Bei partiell definierten Konzepten muss zusätzlich ein generisches Konstrukt gefordert werden. Dieses repräsentiert den Unterschied zu der durch die Definition definierten Menge. Ein partiell definiertes Konstrukt $C \sqsubseteq A$ würde beispielsweise zu $A \sqcap C'$ aufgefaltet werden. Der durch die Auffaltung entstandene Term muss vor der weiteren Bearbeitung wieder in NNF umgeformt werden.

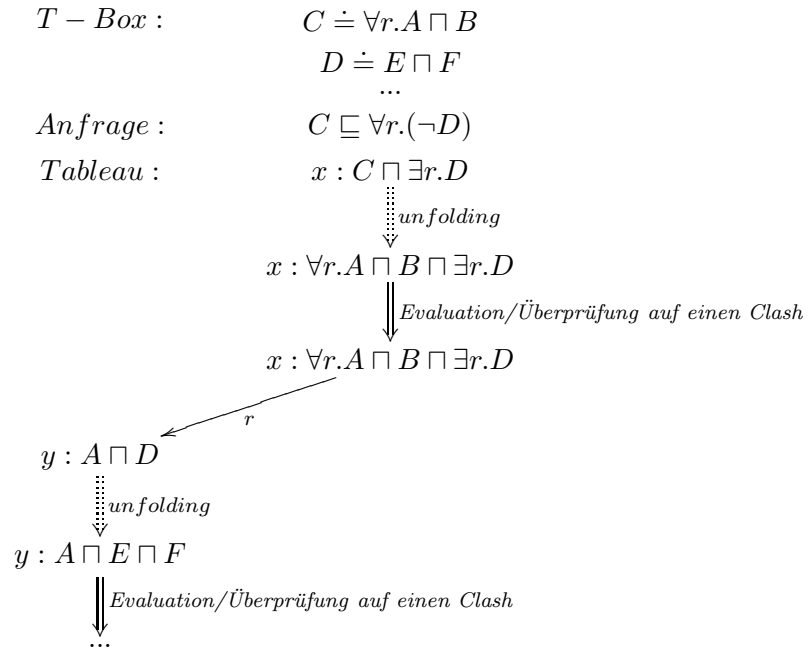


Abbildung 3.8: Lazy-Unfolding

4 Tableaubasiertes Erklären

Dieser Abschnitt befasst sich mit der Generierung von Erklärungen für tableaubasierte Subsumtionsbeweise. Zunächst wird beschrieben, was generell zu beachten ist, um Erklärungen zu Tableau-Beweisen generieren zu können. Anschließend wird ein konkreter Ansatz vorgestellt, um Subsumtionsbeweise für die OWL Lite zugrunde liegende Beschreibungslogik *SHIF* zu erklären. Das Kapitel schließt mit einer Diskussion darüber, wie das dargestellte Erklärungsverfahren verbessert und erweitert werden kann.

4.1 Grundlegendes zum Erklären

Tableaubasiertes Erklären versucht, einen beschreibungslogischen Tableau-Beweis zu erklären. Es soll ein gewisser Grad an Transparenz erreicht werden, der dem Benutzer verdeutlicht, wie das Ergebnis zustande gekommen ist. Dabei wird davon ausgegangen, dass der Benutzer zwar über grundlegendes Wissen über Beschreibungslogiken verfügt, jedoch nicht zwingend versteht, wie ein Tableau-Beweis funktioniert.

Grundsätzlich sind verschiedene Formen des Erklärens denkbar. In erster Linie muss es auf die Fragestellung, welche durch den Tableau-Beweis beantwortet wird, zugeschnitten sein. Ein Unerfüllbarkeitstest erfordert zum Beispiel eine völlig andere Erklärung als ein Subsumtionstest. Die einfache Subsumtionsanfrage

$$(\geq 2 r) \sqsubseteq (\geq 1 r) ?$$

führt zum Beispiel zu einem Tableau mit dem Startknoten

$$x : (\geq 2 r) \sqcap (\leq 0 r)$$

welcher offensichtlich unerfüllbar ist. Als Erklärung für die Unerfüllbarkeit des Tableaus wäre „*Es können nicht gleichzeitig mehr als 2 und weniger als 0 Füller für die Relation r existieren*“ denkbar. Soll aber die Subsumtionsfrage beantwortet werden, ist eine solche Erklärung relativ unbrauchbar. Der Benutzer muss nicht wissen, dass die Anfrage intern in ein Erfüllbarkeitsproblem umgeformt wurde und dessen Unerfüllbarkeit dann mittels Tableau-Verfahren bewiesen wurde. Vielmehr soll er eine Erklärung für die Gültigkeit der Subsumtionsbeziehung geliefert bekommen. Denkbar wäre eine Aussage wie „*Wenn mehr als 2 Füller für eine Relation r existieren, muss auch mehr als 1 Füller für diese Relation existieren*“.

Um einen komplexen Tableau-Beweis zu erklären, muss man die Information strukturieren. In diesem Ansatz wird jeder Einzelschritt, der zur Lösungsfindung (intern zum Widerspruch) beiträgt, in einer Liste abgelegt. Zusätzlich wird die Baumtiefe, auf welcher der Schritt erfolgte, gespeichert. Als Erweiterung wäre es denkbar, nachträglich zusammengehörende Schritte zusammenzufassen, um schlüssigere Erklärungen zu bekommen. Für die endgültige Darstellung der einzelnen Erklärungsschritte sind verschiedene Formen denkbar. Für die im Rahmen dieser Arbeit entwickelte Visualisierungskomponente (5.7) wurde eine baumartige Darstellung gewählt. Diese ermöglicht es, Teilbäume ein- und auszublenden, und bietet dadurch die nötige Übersicht, um umfangreiche Erklärungen darzustellen.

Diese Arbeit beschränkt sich auf das Erklären von Subsumtionsbeweisen. Es lassen sich jedoch durch die erarbeitete Methode auch Tableau-Beweise, welche andere Fragestellungen beantworten, erklären. Dazu müssen die Erklärungen der einzelnen Schritte umformuliert werden. Je nach Art der Fragestellung sind eventuell auch einige Nachbearbeitungsschritte notwendig.

4.2 Erklären von Subsumtionsbeziehungen

Um Subsumtionsbeziehungen mittels Tableau-Verfahren zu beweisen, wird die Anfrage zunächst in ein Erfüllbarkeitsproblem umgewandelt. Dazu wird die zu beweisende Aussage der Form $C \sqsubseteq D$ negiert. Dies führt zu einer Aussage der Form $C \sqcap \neg D$. Der Tableau-Beweis versucht nun ein Modell für diese negierte Aussage zu finden.

Wir Menschen denken jedoch normalerweise nicht in einer solchen negierten Form. Um eine Aussage zu begründen suchen wir zusammenhängende positive Aussagen. So erscheint uns die Aussage „*Jemand mit mindestens 3 Kindern muss auch mindestens 2 Kinder haben*“ logisch, wir kämen jedoch vermutlich nicht auf die Idee zu sagen „*Dies gilt, da es niemanden gibt, der mehr als 3 Kinder und gleichzeitig weniger als 2 Kinder hat*“.

Um also menschenverständliche Erklärungen zu Subsumtionsbeweisen, die mittels Tableau-Verfahren geführt wurden, liefern zu können, muss man die Bedeutung der einzelnen Schritte in Bezug auf die ursprüngliche Anfrage ausgeben. Wird in einem Knoten zum Beispiel das speziellste Konzept \perp gefunden und kann dessen Herkunft auf die rechte Seite, also den Subsumer der ursprünglichen Subsumtionsanfrage, zurückgeführt werden, so lautet die Begründung „*Alles wird vom allgemeinsten Konzept \top subsumiert*“. Dies begründet sich dadurch, dass das gefundene Konzept \perp einem negierten \top entspricht.

Um aus den einzelnen Tableau-Schritten deren ursprüngliche Bedeutung extrahieren zu können, ist es zwingend notwendig zu wissen, welche Teile aus dem Subsumee und welche aus dem Subsumer abgeleitet wurden. Dies wird auch in dem in [BFH⁺99] beschriebenen Ansatz deutlich. Tableaubasierte Subsumtionsbeweiser ohne Erklärungsfunktion arbeiten typischerweise aus Effizienzgründen auf nur einem zusammenhängenden, beschreibungslogischen

Term, der aus der negierten Subsumtionsanfrage resultiert. Aus der Anfrage $A \sqcap B \sqsubseteq \neg C \sqcup \neg D$ entsteht zum Beispiel der Term $A \sqcap B \sqcap C \sqcap D$. Die Anfrage $A \sqcap B \sqcap C \sqsubseteq \neg D$ führt ebenfalls zum Term $A \sqcap B \sqcap C \sqcap D$. Der Tableau-Beweis für beide Anfragen sieht also genau gleich aus. Um jedoch zu erklären, wie der Beweis zu seinem Ergebnis gekommen ist, ist es notwendig, aus dem Term im Tableau wieder die ursprüngliche Subsumtionsanfrage ableiten zu können. Um dies zu ermöglichen, muss in der Datenstruktur des Tableau die Herkunft der einzelnen Termkonzepte gespeichert werden. In dem in dieser Arbeit dargestellten Ansatz werden immer zwei Terme, einer für die linke Seite (Subsumee) und einer für die rechte Seite (Subsumer), erstellt und auf diesen gearbeitet. Abbildung 4.1 zeigt, in welcher Form die beiden oben genannten Anfragen nach Umwandlung zum Erfüllbarkeitsproblem repräsentiert werden. Semantisch haben sie nach wie vor die gleiche Bedeutung, da linke (lhs) und rechte Seite (rhs) eine Konjunktion bilden.



Abbildung 4.1: Trennung von linker und rechter Seite

Das eigentliche Tableau-Verfahren ändert sich durch diese erweiterte Form der Datenrepräsentation nicht. Bei der Untersuchung auf Clashes muss man jedoch immer beide Terme in Kombination betrachten. Für die Erklärung eines gefundenen Clashes ist die Herkunft der zum Clash führenden Konzepte wieder relevant. Welche Bedeutung die einzelnen Tableau-Schritte in Bezug auf die ursprüngliche Subsumtionsanfrage haben und welche Erklärungen aus ihnen resultieren, wird im Abschnitt 4.3 genauer diskutiert.

4.2.1 Umschalten des Erklärungsmodus

Per Definition wird das speziellste Konzept \perp von allem subsumiert. Ebenso subsumiert das allgemeinste Konzept \top alles andere. Im Tableau-Beweis äußert sich dies dadurch, dass eine Seite äquivalent zu \perp ist. Die Erklärung für einen solchen Fall lautet entweder

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

oder

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

Eine solche Äquivalenz zu erkennen, erfordert unter Umständen mehrere Schritte, gegebenenfalls über die Expansion verschiedener Nachfolgerknoten. Kann die Äquivalenz jedoch frühzeitig erkannt werden, so empfiehlt es sich, in den nachfolgenden Schritten nur diese zu erklären. Statt einer Subsumtionserklärung muss nun eine Erklärung für die Äquivalenz der linken Seite mit \perp oder die der rechten Seite mit \top generiert werden. Oft bietet sich dieses Verfahren an um einen Teilbaum des gesamten Tableau zu erklären. Die resultierende Erklärung ist intuitiv verständlicher als eine, die weiterhin die Subsumtionsbeziehung erklärt. Um jedoch möglichst früh in einen anderen Erklärungsmodus wechseln zu können, ist eine Form von vorausschauendem Reasoning notwendig. Vor der Weiterverarbeitung eines Knotens des Tableaus muss jede Seite auf Unerfüllbarkeit überprüft werden. Sollte eine der beiden Seiten unerfüllbar sein, so muss der Erklärungsmodus gewechselt werden. Die Vorteile und Nachteile dieser Methode werden in 4.5 genauer erläutert.

4.3 Einzelschritte und ihre Erklärungen

Um komplexe Probleme zu verstehen, teilen wir Menschen diese typischerweise in viele kleinere Probleme auf. Dies wird sukzessive fortgesetzt bis die Teilprobleme klein genug sind, um von uns direkt erfasst zu werden. Der hier vorgestellte Ansatz geht ebenfalls nach diesem Prinzip vor. Während der Algorithmus den Tableau aufbaut und versucht die Unerfüllbarkeit des Wurzelknotens zu beweisen, werden alle für die Erklärung relevanten Schritte in eine Liste geschrieben. Diese bildet später die Erklärung für den gesamten Beweis. Die Erklärungsliste ist gewissermaßen ein Logbuch des Algorithmus. Da die einzelnen Einträge unter Umständen nur in Zusammenhang mit dem aktuellen Status des Algorithmus erklärbar sind, wird die Einzelerklärung zu jedem Schritt direkt generiert und mit diesem gespeichert. Im Folgenden wird die Bedeutung der einzelnen Schritte für die Erklärung des Subsumtionsbeweises erläutert.

4.3.1 Auffaltung

Wie im Abschnitt 3.3.1 beschrieben werden vor der Evaluation eines Knotens alle direkt referenzierten Konzepte aufgefaltet. Semantisch bleibt der Ausdruck äquivalent. Damit der Benutzer diese Äquivalenz erkennt und versteht, warum der Algorithmus auf einem veränderten Term weiterarbeitet, wird ein Erklärungsschritt für die Auffaltung eingefügt. Dabei wird sowohl der ursprüngliche als auch der resultierende Term, dargestellt als Subsumtionsbeziehung, ausgegeben. Der intern als rechte Seite abgespeicherte Term wird negiert, um eine Ausgabe der Form „ $lhs \sqsubseteq rhs$ “ zu erzeugen.

Im Beispiel aus Abbildung 3.8 wird der Liste für den ersten Auffaltungsschritt folgende Erklärung hinzugefügt:

Es wird geprüft ob $C \sqsubseteq \forall r.D$ gilt.
Dies ist äquivalent zu $\forall r.A \sqcap B \sqsubseteq \forall r.D$

Erklärungen zu Auffaltungsschritten werden nur eingefügt, falls auch wirklich etwas aufgefaltet wurde. Andernfalls genügt es die zu prüfende Subsumtionsbeziehung anzugeben.

4.3.2 Übergang zum Nachfolgerknoten

Wird in einem Knoten kein lokaler Clash gefunden, werden seine Nachfolger untersucht. Diese müssen zunächst aufgefaltet werden (siehe Abschnitt 3.3.1). Ein solcher Auffaltungsschritt wird in Kombination mit der betreffenden Kante erklärt.

Der zweite Auffaltungsschritt in Abbildung 3.8 wird demnach wie folgt erklärt:

Für die Relation r muss geprüft werden ob $A \sqsubseteq \neg D$ gilt.
Dies ist äquivalent zu $A \sqsubseteq \neg E \sqcup \neg F$

Diese Erklärung stellt auch die Verbindung zum nachfolgenden Unterbeweis dar. Der Algorithmus setzt seine Suche nach einem Clash in den Nachfolgerknoten fort. Dem Benutzer wird dies wie ein eigener Subsumtionsbeweis erklärt. Es wird gezeigt, dass die Subsumtionsbeziehung für die Konzepte die über eine Relation referenziert wurden gilt. Eine wie in Abschnitt 5.7 beschriebene hierarchische Darstellung der Erklärungsschritte bietet sich an, um dies zu verdeutlichen.

4.3.3 Clash durch \perp -Konzept

Wird in einer Konjunktion das speziellste Konzept \perp gefunden, so ist diese unerfüllbar. Um einen solchen Clash im Tableau-Beweis zu erklären, muss man beachten, auf welcher Seite er aufgetreten ist. Wurde \perp auf der linken Seite gefunden, ist diese unerfüllbar und somit als Ganzes äquivalent zu \perp . Dies wird durch folgende Aussage erklärt:

Das speziellste Konzept \perp wird per Definition von allem subsumiert.

Tritt \perp auf der rechten Seite auf, so entspricht dies einem negierten \top Konzept. Der Benutzer kennt dieses Konzept nur als \top , da er immer nur die zu Subsumtionsbeziehungen umgewandelten Terme zu Gesicht bekommt. Die Konjunktion, in welcher \perp gefunden wurde, entspricht einer negierten Disjunktion im ursprünglichen Subsumer. Als eine Disjunktion mit \top als einem Disjunkt, ist dieser äquivalent zu \top . Die Erklärung lautet also:

Das allgemeinste Konzept \top subsumiert per Definition alles andere.

Wie alle Clash-Erklärungen, stellen diese beiden abschließende Aussagen für umfangreiche Listen von Erklärungsschritten dar. Der Benutzer wird durch die vorhergehenden Schritte zu dem Punkt geführt, an dem das Auftreten von \perp (bzw. \top) direkt erkennbar ist.

4.3.4 Clash durch Konjunktion eines Konzepts und dessen Negation

Eine Unerfüllbarkeit, die durch Konjunktion eines Konzepts und dessen Negation (z.B. $C \sqcap \neg C$) hervorgerufen wird, lässt sich auf vier verschiedene Arten erklären.

Stehen beide auf der gleichen Seite, so entsprechen sie einem dort aufgetretenen \perp -Konzept. Die Erklärungen sind also analog zu denen in 4.3.3. Zusätzlich wird jedoch erklärt, wie es zum Auftreten von \perp bzw. \top kam, da dies nicht explizit im Term enthalten ist.

Für die linke Seite ergibt sich somit folgende Erklärung:

$C \sqcap \neg C$ ist äquivalent zum speziellsten Konzept \perp .
Dieses wird per Definition von allem subsumiert.

Ein Ausdruck der Form $C \sqcap \neg C$ auf der rechten Seite entspricht in der ursprünglichen Subsumtion dem Ausdruck $\neg C \sqcup C$. Dieser ist immer erfüllt und somit äquivalent zu \top . Die Erklärung lautet:

$C \sqcup \neg C$ ist äquivalent zum allgemeinsten Konzept \top .
Dieses subsumiert per Definition alles andere.

Des Weiteren kann ein Konzept auf der einen und dessen Negation auf der anderen Seite der Subsumtionsbeziehung stehen. Auch dies führt im Tableau-Beweis zu einem Clash, da die beiden Seiten als Konjunktion betrachtet werden müssen. Semantisch bedeutet es, dass entweder das Konzept oder dessen Negation auf beiden Seiten der Subsumtionsbeziehung steht. Auf der rechten Seite, in der Beschreibung des Subsumers, kann es im Tableau nur in einer Konjunktion vorkommen, falls es in der ursprünglichen Form in einer Disjunktion stand. Trat es dort bereits in einer Konjunktion auf, so wird diese im Tableau zu einer Disjunktion, was bedeutet, dass alle Disjunkte überprüft werden müssen. Es würde also eine Teilerklärung zustande kommen, die den Clash mit diesem Disjunkt erklärt (siehe Abschnitt 4.3.6). Aus Sicht des Benutzers kann das Konzept auf der rechten Seite also nur in einer Disjunktion oder einzeln vorkommen. Der Einfachheit halber wird im Folgenden angenommen, dass das Konzept auf der rechten Seite einzeln auftritt. Dies entspricht den Erklärungen, welche durch die in 5 beschriebene Erklärungskomponente generiert werden können, da explizite Disjunktion bei der Implementierung nicht berücksichtigt wurde.

Ein verteiltes Auftreten eines Konzepts und seiner Negation hat demnach eine der beiden Bedeutungen:

$$\boxed{(C \sqcap \dots) \sqsubseteq C} \text{ oder } \boxed{(\neg C \sqcap \dots) \sqsubseteq \neg C}.$$

Eine solche Aussage genügt auch als Erklärung für den Clash, denn es ist offensichtlich, dass eine Subsumtionsbeziehung dieser Art gilt.

4.3.5 Clash durch widersprüchliche Kardinalitätseinschränkungen

Wie in 3.1 beschrieben, können Kardinalitätseinschränkungen zu einem Clash führen. Dieser tritt auf, wenn zum einen $(\geq m r)$ und zum anderen $(\leq n r)$ (mit $m > n$ und $m, n \in \mathbb{N}$) Füller für eine Relation r gefordert werden. Semantisch ist dies einleuchtend, da nicht gleichzeitig mindestens m und höchstens n Füller existieren können, wenn $m > n$ gilt. Ein Term der Form $(\geq 1 r) \sqcap (\leq 0 r)$ ist zum Beispiel unerfüllbar. Da bei der Generierung einer Erklärung immer auf die Herkunft der einzelnen Konstrukte geachtet werden muss, gibt es vier verschiedene Arten einen solchen Clash zu erklären.

Stehen beide Einschränkungen auf der linken Seite, so lautet die Erklärung:

Es können nicht gleichzeitig mehr als m und weniger als n Füller für die Relation r existieren. Daher ist die linke Seite äquivalent zum speziellsten Konzept \perp und dieses wird per Definition von allem anderen subsumiert.

Stehen beide Restriktionen als Konjunktion auf der rechten Seite, so entsprechen sie einer Disjunktion im ursprünglichen Subsumtionsterm. Der Ausdruck $\dots \sqcap (\geq 1 r) \sqcap (\leq 0 r)$ entsteht zum Beispiel aus der Subsumtionsanfrage $\dots \sqsubseteq (\leq 0 r) \sqcup (\geq 1 r)$. Diese Disjunktion ist offensichtlich immer erfüllt. Allgemein gilt, dass $(\geq m r) \sqcap (\leq n r)$ auf der rechten Seite einem negierten $(\leq (m - 1) r) \sqcup (\geq (n + 1) r)$ entspricht. Dabei ist $(m - 1)$ maximal um 1 kleiner als $(n + 1)$, da $m > n$ und $m, n \in \mathbb{N}$ gilt. Durch die Disjunktion der beiden Kardinalitätseinschränkungen werden demnach alle möglichen Mengen von Relationsfüllern für r zugelassen. Somit ist der Subsumer äquivalent zu \top .

Die Erklärung für diesen Fall lautet:

Es existieren immer mehr als $(n + 1)$ oder weniger als $(m - 1)$ Füller für die Relation r . Somit ist die rechte Seite äquivalent zu \top , welches per Definition alles andere subsumiert.

Zwei weitere Fälle werden dadurch beschrieben, dass eine Einschränkung auf der rechten und die andere auf der linken Seite steht. Diese Fälle sind selbsterklärend, wenn man die rechte Seite in ihre ursprünglichen Form bringt. Als Erklärung entsteht eine der beiden Aussagen:

$(\geq m r)$ wird von $(\geq (n + 1) r)$ subsumiert.

oder

$(\leq n r)$ wird von $(\leq (m - 1) r)$ subsumiert.

Diese Aussagen sind offensichtlich verständlich, da für m und n die tatsächlichen Werte eingesetzt werden und $m > n$ sowie $m, n \in \mathbb{N}$ gilt.

4.3.6 Disjunktionen

Wie im Abschnitt 4.3.4 bereits angedeutet, bewirkt eine Disjunktion ein Aufsplitten des Beweises zu mehreren Teilbeweisen. Im Tableau muss jedes Disjunkt zu einem Clash führen, um den Knoten unerfüllbar zu machen. OWL Lite erlaubt keine Disjunktion. Im Tableau kann sie demnach nur auf der rechten Seite auftreten, wo sie durch eine negierte Konjunktion entsteht (siehe Abb. 3.2). Ein solche Disjunktion auf der rechten Seite wird als Aufsplittung des Beweises in mehrere Unterbeweise erklärt. Der dazu einleitende Erklärungsschritt sieht wie folgt aus:

Da alle folgenden Subsumtionen gelten gilt auch die gesuchte Subsumtion $C \sqsubseteq D \sqcap E$.

Beweis für $C \sqsubseteq D \dots$

Beweis für $C \sqsubseteq E \dots$

Daraufhin folgen die Erklärungen für die Teilbeweise. Diese sollten in einer übersichtlichen Form dargestellt werden. Diese Methode, Disjunktionen zu erklären, ist nicht in allen Fällen die Beste. Oft wäre es wünschenswert, Erklärungen über mehrere Schritte zu generieren. Hierzu könnten nachverarbeitende Schritte eingeführt werden. Einige Ideen dazu werden im Abschnitt 4.5 diskutiert. Das im Rahmen dieser Arbeit implementierte System MEX beschränkt sich jedoch auf eine einfache Auflistung der Erklärungen für die durch die Disjunktion geforderten Subsumtionsbeziehungen.

4.3.7 domain- und range-Restriktionen

OWL Lite erlaubt es, globale domain- und range-Restriktionen für Relationen anzugeben. Diese werden im Tableau-Beweis relevant, sobald mindestens ein Füller für eine auf diese Art eingeschränkte Relation gefordert wird. Eine domain-Einschränkung muss dann dem Elternknoten, eine range-Einschränkung dem Nachfolgeknoten hinzugefügt werden. Eine solche Restriktion kann durch einen beliebigen Ausdruck der zugrunde liegenden Beschreibungslogik (wiederum OWL Lite) definiert werden. Die Restriktionen können somit auch zu einem Clash im Tableau führen.

Dem Benutzer muss die Herkunft des durch domain- oder range-Restriktionen hinzugefügten Ausdrucks erklärt werden. Andernfalls wäre es schwer nachzuvollziehen, woher die zusätzlichen Konstrukte kommen.

Eine aufgrund einer domain-Restriktion hinzugefügte Einschränkung wird wie folgt erklärt:

Da die Relation r eine domain-Einschränkung auf A hat, wird aus $\exists r.C \sqsubseteq \exists r.D$ die Beziehung $\exists r.C \sqcap A \sqsubseteq \exists r.D$

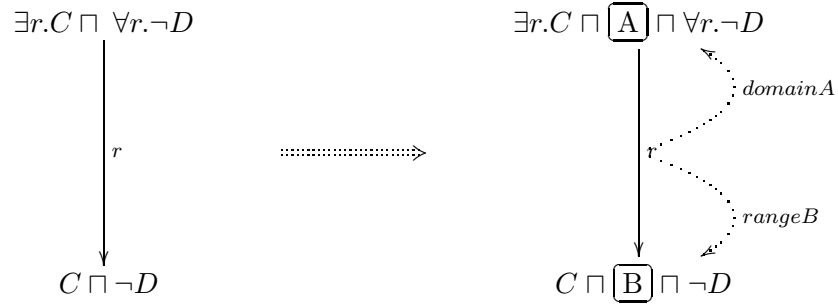


Abbildung 4.2: Veränderungen im Tableau durch domain- und range-Restriktionen

Bei einer range-Restriktion muss vorher ein Übergang (e.g. Auffaltungsschritt) über eine Kante erfolgt sein. Das Zusammenspiel der beiden zugehörigen Erklärungen sieht dann wie folgt aus:

Für die Relation r muss geprüft werden ob $C \sqsubseteq D$ gilt.

Da die Relation r eine range-Einschränkung auf B hat wird aus $C \sqsubseteq D$ die Beziehung $C \sqcap B \sqsubseteq D$

Diese Form der Erklärung ist nicht ideal. Die Terme können durch viele domain- und range-Restriktionen unübersichtlich groß werden. Ein Großteil der hinzugefügten Konstrukte hat jedoch in den vielen Fällen keine Bedeutung für die Gültigkeit der Subsumtionsbeziehung. Einige Ideen zur Verbesserung dieser Erklärungen werden in 4.5 diskutiert.

4.3.8 Merging von Relationsfüllern

Die Kombination von Existenzquantoren und \leq -Einschränkungen kann im Tableau-Beweis Merging von mehreren Nachfolgerknoten bewirken. Beim Bearbeiten des Ausdrucks

$$\exists r.A \sqcap \exists r.B \sqcap (\leq 1 r)$$

werden beispielsweise zunächst zwei Nachfolgerknoten A und B über die Relation r erstellt. Die danach evaluierte Kardinalitätseinschränkung ($\leq 1 r$) fordert jedoch, dass es nur einen Nachfolgerknoten geben darf. Semantisch bedeutet dies, dass der Nachfolger über r beide Eigenschaften, also A und B erfüllen muss. Die beiden Knoten müssen also zusammengefasst (merged) werden. Abbildung 4.3 zeigt, wie ein solcher Merging-Vorgang den Tableau verändert.

Für Kardinalitätseinschränkungen größer 1 wird das Merging erheblich komplizierter. Es müssen dann verschiedene Möglichkeiten ausprobiert werden, die alle zu einem Clash führen müssen. Da OWL Lite nur Kardinalitätseinschränkungen von 0 oder 1 zulässt, beschränkt sich das in dieser Arbeit entwickelte

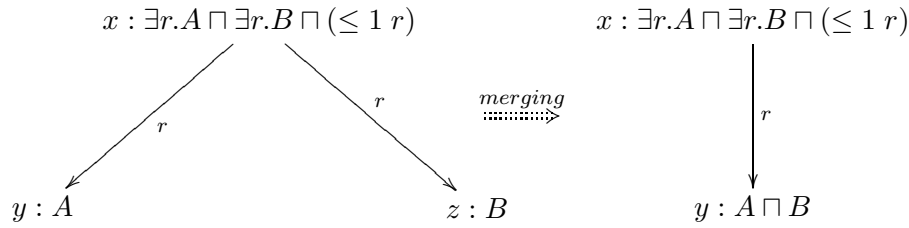


Abbildung 4.3: Einfaches Merging

Erklären auf einfaches Merging. Auch kann das Merging im Sprachumfang von OWL Lite nur auf der linken Seite auftreten, da explizite Disjunktion nicht erlaubt ist. Somit kann rechts keine Konjunktion im Tableau auftreten. Die Erklärung für den Merging-Schritt aus Abbildung 4.3 lautet:

Da es nur einen Füller für die Relation r auf der linken Seite geben darf muss dieser $A \sqcap B$ erfüllen.

Danach folgt die Erklärung für den gemergten Nachfolgerknoten. Komplexere Merging-Vorgänge zu erklären ist weitaus schwieriger. Einige Ideen dazu werden in Abschnitt 4.5 dargestellt.

4.3.9 Funktionale Relationen

OWL Lite erlaubt das Definieren von funktionalen Relationen. Diese haben die Eigenschaft, jeweils nur einen Füller zu erlauben. Ist f als funktionale Relation definiert, sind die Ausdrücke

$$\exists f.A \sqcap \exists f.B \text{ und } \exists f.A \sqcap \exists f.B \sqcap (\leq 1 f)$$

äquivalent. Es kann also zu Merging kommen, obwohl keine explizite Kardinalitätseinschränkung für f gefordert wurde. Um den Benutzer darauf hinzuweisen, wird folgender Erklärungsschritt eingefügt:

Da die Relation f funktional definiert ist, kann sie nur einen Füller haben.
Für diesen muss demnach $A \sqcap B$ gelten.

4.3.10 Subrelationen und equivalente Relationen

In OWL Lite kann man Relationshierarchien festlegen. Man könnte beispielsweise eine Relation *hat-sohn* als Subrelation von *hat-kind* definieren. Semantisch bedeutet dies, dass jeder Füller von *hat-sohn* auch ein Füller von *hat-kind* ist. Der Umkehrschluss daraus ist, dass jede Einschränkung auf die in der Hierarchie höher stehende Relation *hat-kind* auch für die Subrelation *hat-sohn* gelten muss. Wird zum Beispiel gefordert, dass jedes Kind einer Person blaue Augen haben muss, so muss selbstverständlich auch jeder Sohn dieser Person

blaue Augen haben. Im Tableau-Beweis werden Restriktionen auf Relationen auch auf deren Subrelationen übertragen. Dies gilt sowohl für qualitative als auch für quantitative Einschränkungen. Für qualitative Einschränkungen wird eine Erklärung der folgenden Form geliefert:

Für alle Füller der Relation *hat-kind* gilt die Einschränkung *blaueAugen*. Diese muss auch für alle Füller von *hat-Sohn* gelten, da dies eine Subrelation von *hat-kind* ist.

Diese Erklärung wird zum Beispiel durch den Term $\exists \text{hat-sohn}.A \sqcap \forall \text{hat-kind}.\text{blaueAugen}$ hervorgerufen.

Kardinalitätseinschränkungen, die eine maximale Anzahl von Füllern für eine Relation r fordern, gelten auch für Subrelationen von r . Wird beispielsweise durch $(\leq 0 \text{ hat-kind})$ gefordert, dass kein Kind existiert, so darf selbstverständlich auch kein Sohn existieren. Der Ausdruck $(\leq 0 \text{ hat-kind}) \sqcap (\geq 1 \text{ hat-sohn})$ führt demnach zu einem Clash. Allgemein gilt, dass $(\leq n r) \sqcap (\geq m \text{ sub}R)$ mit $\text{sub}R \sqsubseteq r$ und $m > n$ mit $m, n \in \mathbb{N}$ einen Clash darstellt. Auch hier muss bei der Erklärung darauf geachtet werden, auf welcher Seite die Restriktionen gefordert wurden. Die Erklärungen sind analog zu denen aus Abschnitt 4.3.5 und sehen wie folgt aus.

Beide Kardinalitätseinschränkungen stehen auf der linken Seite:

Es können nicht gleichzeitig mehr als m Füller für die Subrelation $\text{sub}R$ und weniger als n Füller für die Relation r existieren. Daher ist die linke Seite äquivalent zum speziellsten Konzept \perp und dieses wird per Definition von allem anderen subsumiert.

Beide Kardinalitätseinschränkungen stehen auf der rechten Seite:

Es existieren immer mehr als $(n+1)$ Füller für die Relation $\text{sub}R$ oder weniger als $(m-1)$ Füller für die Subrelation r . Somit ist die rechte Seite äquivalent zu \top welches per Definition alles andere subsumiert.

Falls die beiden Kardinalitätseinschränkungen auf verschiedenen Seiten stehen, ist die Subsumtionsbeziehung sehr einleuchtend.

$(\geq m \text{ sub}R)$ wird von $(\geq (n+1) r)$ subsumiert.

$(\leq n r)$ wird von $(\leq (m-1) \text{ sub}R)$ subsumiert.

Durch eine Kardinalitätseinschränkung auf eine Relation kann merging von Füllern ihrer Subrelationen notwendig werden. Existieren beispielsweise zwei Nachfolgerknoten für die Subrelation *hat-sohn* und ein Konstrukt $(\leq 1 \text{ hat-kind})$ fordert, dass es nur einen Füller für die Relation *hat-kind* geben darf, so müssen die Beiden Füller gemerged werden.

Allgemein gilt, dass es nicht mehr Füller einer Subrelation geben darf als für

deren Superrelation erlaubt sind. Die durch merging entstehenden Relationskanten entsprechen immer der in der Relationshierarchie niedrigsten Relation, welche in das merging eingeflossen ist.

Die Erklärung für einen solchen merge-Vorgang sieht wie folgt aus

Da es nur einen Füller für die Relation r geben darf, darf es auch nur einen Füller für deren Subrelation $subR$ geben. Demnach muss dieser folgende Gestalt haben: (*neuer Knoten*)

Das in dieser Arbeit entwickelte Erklären ist auf die Beschreibungslogik \mathcal{SHIF} beschränkt. Es kommen also nur die Kardinalitäten 0 und 1 in Frage. Werden 0 Füller gefordert, führt jede bestehende Kante zu einem Clash. Daraus folgt, dass Merging immer nur zu einem Knoten erfolgt. Für Sprachen, die Kardinalitätseinschränkungen > 1 erlauben, wird das Merging weitaus komplizierter. Erklärungen für einen solchen komplexen Mergingvorgang zu generieren, gestaltet sich weitaus schwerer.

Für äquivalente Relationen sind die Erklärungen analog, da

$$r \doteq s \iff (r \sqsubseteq s) \sqcap (s \sqsubseteq r)$$

gilt.

4.3.11 Transitiv Relationen

Wird eine Relation r transitiv definiert, so gelten \forall -Einschränkungen für alle Nachfolgerknoten, welche über ein Kette von r -Kanten erreicht werden können. Der Tableau-Algorithmus löst dies typischerweise, indem dem direkten Nachfolgerknoten sowohl die geforderte Eigenschaft als auch die gesamte \forall -Einschränkung hinzugefügt wird (siehe Abb. 4.4). Um bei dieser Vorgehensweise die Terminierung des Algorithmus sicherzustellen, muss paarweises Blocking angewendet werden (siehe 3.2).

$$\begin{array}{c}
 r \text{ transitiv} \\
 x : (\geq 1 r) \sqcap \forall r.B \\
 \downarrow r \\
 y : B \sqcap \forall r.B
 \end{array}$$

Abbildung 4.4: Erweiterung einer \forall -Restriktion für eine transitive Relation

Wird eine *forall*-Restriktion über eine transitive Relation auf der linken Seite expandiert, lässt sich dies relativ einfach erklären:

Da r transitiv definiert ist, muss B auf der linken Seite von jedem Nachfolger über r erfüllt werden. Dazu muss der direkte Nachfolger $\dots \sqcap \forall r.B$ auf der linken Seite erfüllen.

Steht die zu expandierende \forall -Restriktion jedoch auf der rechten Seite, so entspricht sie einer negierten \exists -Restriktion im ursprünglichen Term. Die durch die Transitivität der entsprechenden Relation dem Nachfolgerknoten hinzugefügte \forall -Restriktion führt dort zu einer Konjunktion auf der rechten Seite. Aus Sicht des Benutzers entsteht also eine Disjunktion auf der rechten Seite. Angenommen $\exists r.B$ wurde auf der rechten Seite der Subsumtionsbeziehung gefordert, im Tableau steht demnach $\forall r.\neg B$ auf der rechten Seite, wird dies wie folgt erklärt:

Da r transitiv definiert ist, kann B auf der rechten Seite direkt oder von einem beliebigen Nachfolger über r erfüllt werden. Dazu muss der direkte Nachfolger $\dots \sqcup \exists r.B$ auf der rechten Seite erfüllen.

Ein unerfahrener Benutzer wird mit diesen Erklärungen jedoch überfordert sein. Besser wäre es, die Information, dass in jedem Nachfolgeknoten ein B gefordert werden muss, separat, also nicht im Tableau-Knoten direkt zu speichern und als Erklärung für eventuelle Nachfolger folgenden Schritt hinzuzufügen:

Da in einem vorhergehenden Knoten die Eigenschaft B für alle Nachfolger über die transitive Relation r gefordert wurde, muss diese auch in diesem Nachfolger gelten.

Zusätzlich könnte diese Erklärung nur bei Bedarf hinzugefügt werden, das heißt, nur wenn die Information nötig ist, um den aufgetretenen Clash zu erklären. Die im Rahmen dieser Arbeit entwickelte Erklärungskomponente (siehe 5) verwendet jedoch die erste Variante. Verbesserungen werden in Abschnitt 4.5 diskutiert.

4.3.12 Inverse Relationen

Inverse Relationen werden prinzipiell wie einfache Relationen behandelt. Der Unterschied ist, dass Tableau-Erweiterungen über inverse Relationen auch Auswirkungen auf den Vorgängerknoten haben können. Eigenschaften, die über ein \forall -Konstrukt gefordert werden, müssen beispielsweise dem Vorgängerknoten hinzugefügt werden, falls dieser über die inverse Relation mit dem jetzigen in Verbindung steht. Abbildung 4.5 zeigt einen solchen Fall. Der über die inverse Relation veränderte Vorgängerknoten muss nun neu evaluiert werden. Um diesen Schritt zurück im Tableau zu erklären, kann folgender Erklärungsschritt geliefert werden:

Da für die inverse Relation $invR$ der Relation r A gefordert wurde, muss dies auch für die vorhergehende Subsumtionsbeziehung gelten. Diese ändert sich also zu $lhs \sqsubseteq \neg rhs$ und muss demnach neu überprüft werden.

Nachfolgend wird der Vorgängerknoten neu evaluiert und erzeugt dabei die entsprechenden Erklärungsschritte.

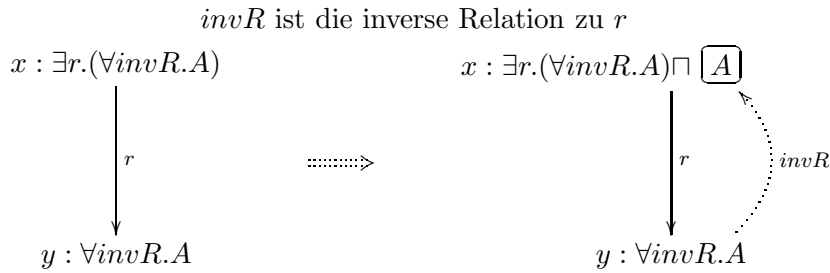


Abbildung 4.5: Erweiterung einer \forall -Restriktion für eine inverse Relation

Des weiteren kann eine Kardinalitätseinschränkung auf eine inverse Relation zu einem Clash führen. Der Vorgängerknoten muss immer als ein Füller über die Inverse der Relation, über die er seinen Nachfolger referenziert, betrachtet werden. Wird nun im Nachfolgerknoten eine Kardinalitätseinschränkung auf diese inverse Relation gefordert, kann dies in Konflikt zu dem vorhandenen Vorgänger stehen. Wird beispielsweise in einem über die Relation *hat-kind* referenzierten Nachfolgerknoten (≤ 0 *hat-eltern*) gefordert, ist dies offensichtlich unerfüllbar.

Um einen solchen Kardinalitätsclash zu erklären, muss beachtet werden, auf welcher Seite die Kardinalitätseinschränkung gefordert wurde. Werden beide im aktuellen Knoten gefordert, so ist die Erklärung analog zu der in Abschnitt 4.3.5 beschriebenen für einfache Relationen. Erfolgt der Clash jedoch aufgrund einer einzelnen \leq -Einschränkung, die widersprüchlich zum existierenden Vorgängerknoten ist, muss er anders erklärt werden.

Steht die Einschränkung ($\leq 0 r$) auf der rechten Seite, so lautet die Erklärung

($\geq 1 r$) ist erfüllt, da dieser Term über die Rolle $invR$ referenziert wurde und somit der Vorgänger einen Füller für deren inverse Rolle r darstellt. Dadurch ist die rechte Seite äquivalent zu \top und subsumiert somit alles andere.

Wird ($\leq 0 r$) auf der linken Seite gefordert, muss die Subsumtionsbeziehung wie folgt erklärt werden

Die Forderung ($\leq 0 r$) widerspricht der Tatsache, dass der Term für einen Füller der Rolle $invR$ gelten muss. Der Vorgänger ist demnach bereits ein Füller für deren inverse Rolle r . Dadurch ist die linke Seite äquivalent zu \perp , welches von allem anderen subsumiert wird.

Ein weiteres Erklärungsproblem stellt das Merging von Nachfolgern über inverse Rollen mit dem Vorgängerknoten dar. Dies ist notwendig, wenn bereits neue Nachfolgerknoten über eine inverse Rolle $invR$ erstellt wurden, der Vorgängerknoten ebenfalls über $invR$ referenziert ist und eine Einschränkung der Form ($\leq 1 invR$) gefordert wird. Daraufhin werden die Nachfolgerknoten gelöscht und deren Eigenschaften dem Vorgängerknoten hinzugefügt. Ein solcher Vorgang wird in Abbildung 4.6 skizziert.

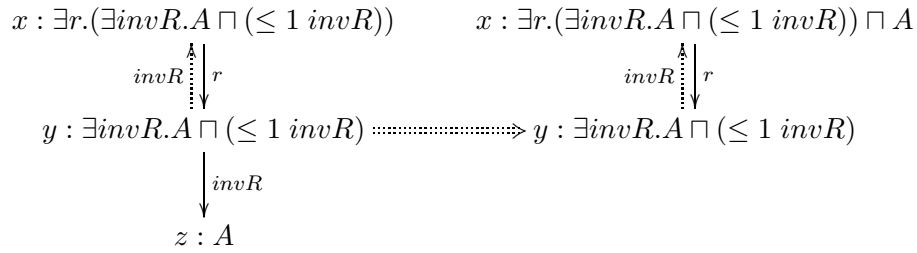


Abbildung 4.6: Merging mit dem Vorgängerknoten über eine inverse Rolle

Ein solcher Merging Vorgang führt zur erneuten Evaluation des Vorgängerknotens x . Gleichzeitig stellt dieser den geforderten Nachfolger über $invR$ dar. Dieser Zusammenhang muss dem Benutzer erklärt werden. Eine solche Erklärung kann wie folgt aussehen:

Da es nur einen Füller für $invR$ geben darf und der Vorgänger $\exists r.(\exists invR.A \sqcap (\leq 1 invR))$ bereits über $invR$ referenziert ist, muss A auch für diesen gelten. Es muss also geprüft werden, ob $\exists r.(\exists invR.A \sqcap (\leq 1 invR)) \sqcap A$ für den Vorgängerknoten gilt.

Dieser Abschnitt hat verdeutlicht, dass inverse Rollen den Erklärungsprozess erheblich erschweren. Die Erklärungen sind weniger leicht verständlich als die der übrigen Fälle. Bei der im Rahmen dieser Arbeit implementierten Erklärungskomponente wurde aus diesen Gründen auf die Behandlung inverser Rollen verzichtet.

4.3.13 Zusammenfassung

Folgende Tabelle fasst die einzelnen Tableau-Schritte und ihre Erklärungen zusammen. Dabei werden folgende Abkürzungen verwendet:

- C, D, E = komplex definierte Konzepte
 $l1$ = linke Seite vor Ausführung des Tableau-Schritts
 $r1$ = negierte rechte Seite vor Ausführung des Tableau-Schritts
 $l2$ = linke Seite nach Ausführung des Tableau-Schritts
 $r2$ = negierte rechte Seite nach Ausführung des Tableau-Schritts
 $n1$ = erster Nachfolger über die jeweilige Relation
 $n2$ = erster Nachfolger über die jeweilige Relation

Tableau-Schritt	Seite	Erklärung
Auffaltung	beide	Es wird geprüft ob $l1 \sqsubseteq r1$ gilt. Dies ist äquivalent zu $l2 \sqsubseteq r2$
Auffaltung über eine Relation	beide	Für die Relation r muss geprüft werden ob $l1 \sqsubseteq r1$ gilt. Dies ist äquivalent zu $l2 \sqsubseteq r2$
Clash: \perp	links	Das speziellste Konzept \perp wird per Definition von allem subsumiert.
Clash: \top	rechts	Das allgemeinste Konzept \top subsumiert per Definition alles andere.
Clash: $C \sqcap \neg C$	links	$C \sqcap \neg C$ ist äquivalent zum speziellsten Konzept \perp . Dieses wird per Definition von allem subsumiert.
Clash: $C \sqcup \neg C$	rechts	$C \sqcup \neg C$ ist äquivalent zum allgemeinsten Konzept \top . Dieses subsumiert per Definition alles andere.
Clash: $C \sqcap \neg C$	li/re	$(C \sqcap \dots) \sqsubseteq C$ gilt per Definition.
Clash: $\neg C \sqcap C$	re/li	$(\neg C \sqcap \dots) \sqsubseteq \neg C$ gilt per Definition.
Clash: $(\geq m r) \sqcap (\leq n r)$	links	Es können nicht gleichzeitig mehr als m und weniger als n Füller für die Relation r existieren. Daher ist die linke Seite äquivalent zum speziellsten Konzept \perp und dieses wird per Definition von allem anderen subsumiert.
Clash: $(\geq m r) \sqcap (\leq n r)$	rechts	Es existieren immer mehr als $(n - 1)$ oder weniger als Füller für die Relation r . Somit ist die rechte Seite äquivalent zu \top , welches per Definition alles andere subsumiert.
Clash: $(\geq m r) \sqcap (\leq n r)$	li/re	$(\geq m r)$ wird von $(\geq (n + 1) r)$ subsumiert.
Clash: $(\leq n r) \sqcap (\geq m r)$	re/li	$(\leq n r)$ wird von $(\leq (m - 1) r)$ subsumiert.

$C \sqcap D \sqcup E$	beide	Da alle folgenden Subsumtionen gelten gilt auch die gesuchte Subsumtion $C \sqsubseteq D \sqcap E$.
domain-Restriktion	beide	Da die Relation r eine domain-Einschränkung auf A hat wird aus $l1 \sqsubseteq r1$ die Beziehung $l2 \sqsubseteq r2$.
range-Restriktion	beide	Da die Relation r eine range-Einschränkung auf B hat wird aus $n1 : l1 \sqsubseteq r1$ die Beziehung $n1 : l2 \sqsubseteq r2$.
merging	beide	Da es nur einen Füller für die Relation r auf der ... Seite geben darf muss dieser $n1 \sqcap n2$ erfüllen.
merging funktionale Relation	beide	Da die Relation f funktional definiert ist, kann sie nur einen Füller haben. Für diesen muss demnach $n1 \sqcap n2$ gelten.
\forall Subrelation	beide	Für alle Füller der Relation r gilt die Einschränkung A . Diese muss auch für alle Füller von $subR$ gelten, da dies eine Subrelation von r ist.
Clash: ($\geq m \ subR$) \sqcap ($\leq n \ r$)	links	Es können nicht gleichzeitig mehr als m Füller für die Subrelation $subR$ und weniger als n Füller für die Relation r existieren. Daher ist die linke Seite äquivalent zum speziellsten Konzept \perp und dieses wird per Definition von allem anderen subsumiert.
Clash: ($\geq m \ subR$) \sqcap ($\leq n \ r$)	rechts	Es existieren immer mehr als $(n - 1)$ Füller für die Subrelation $subR$ oder weniger als $(m + 1)$ Füller für die Relation r . Somit ist die rechte Seite äquivalent zu \top , welches per Definition alles andere subsumiert.
Clash: ($\geq m \ subR$) \sqcap ($\leq n \ r$)	li/re	($\geq m \ subR$) wird von ($\geq (n + 1) \ r$) subsumiert.
Clash: ($\geq m \ subR$) \sqcap ($\leq n \ r$)	re/li	($\leq n \ r$) wird von ($\leq (m - 1) \ subR$) subsumiert.
merging Subrelation	beide	Da es nur einen Füller für die Relation r geben darf, darf es auch nur einen Füller für deren Subrelation $subR$ geben. Demnach muss dieser folgende Gestalt haben: $n1 \sqcap n2$
\forall transitive Relation	beide	Da in einem vorhergehenden Knoten die Eigenschaft B für alle Nachfolger über die transitive Relation r gefordert wurde, muss diese auch in diesem Nachfolger gelten.
inverse Relation	beide	Da für die inverse Relation $invR$ der Relation r A gefordert wurde, muss dies auch für die vorhergehende Subsumtionsbeziehung gelten. Diese ändert sich also zu $lhs \sqsubseteq \neg rhs$ und muss demnach neu überprüft werden.

Clash: inverse Relation	rechts	$(\geq 1r)$ ist erfüllt, da dieser Term über die Rolle $invR$ referenziert wurde und somit der Vorgänger einen Füller für deren inverse Rolle r darstellt. Dadurch ist die rechte Seite äquivalent zu \top und subsumiert somit alles andere.
Clash: inverse Relation	links	Die Forderung $(\leq 0r)$ widerspricht der Tatsache, dass der Term für einen Füller der Rolle $invR$ gelten muss. Der Vorgänger ist demnach bereits ein Füller für deren inverse Rolle r . Dadurch ist die linke Seite äquivalent zu \perp , welches von allem anderen subsumiert wird.
merging inverse Relation	beide	Da es nur einen Füller für $invR$ geben darf und der Vorgänger $\exists r.(\exists invR.A \sqcap (\leq 1 invR))$ bereits über $invR$ referenziert ist, muss A auch für diesen gelten. Es muss also geprüft werden, ob $\exists r.(\exists invR.A \sqcap (\leq 1 invR)) \sqcap A$ für den Vorgängerknoten gilt.

4.4 Erklärungsbeispiele zu komplexen Subsumtionsanfragen

In diesem Teil werden einige umfangreiche Beispiele dargestellt. Sie illustrieren das Zusammenwirken der einzelnen Erklärungsschritte zu einer Gesamterklärung. Es werden sowohl der aufgebaute Tableau als auch die entsprechenden Erklärungsschritte veranschaulicht.

Das erste Beispiel ist sehr einfach gehalten. Der zugehörige Tableau-Beweis sowie die dazu generierte Erklärung werden in Abbildung 4.7 dargestellt. Es basiert auf folgender T-Box:

$A \equiv \exists r.E \sqcap \forall r.C$
$B \equiv \exists r.D$
$C \equiv \exists q.E$
$D \equiv (\geq 1 q)$
$E \sqsubseteq \top$

Die zu beweisende Subsumtionsanfrage lautet

$A \stackrel{?}{\sqsubseteq} B$

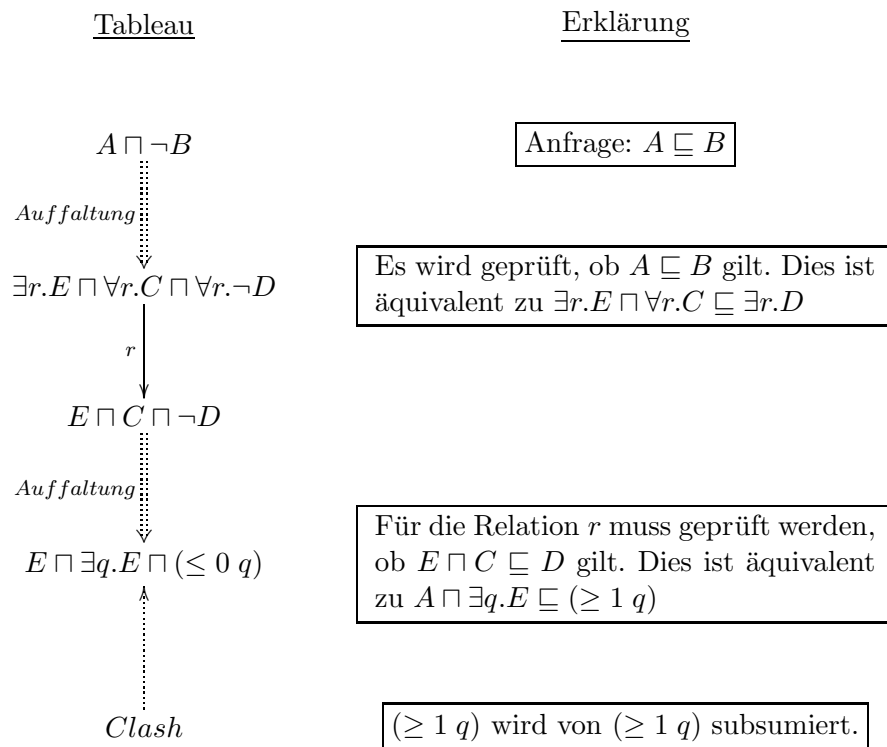


Abbildung 4.7: Komplexes Beispiel 1

Das Beispiel in Abbildung 4.8 zeigt, wie Merging von zwei Rollenfüllern erklärt wird. Des weiteren wird die in 4.5.2 beschriebene Optimierung durch eine Filterfunktion dargestellt.

Folgende T-Box liegt vor:

$$\begin{array}{l} A^* \equiv \exists r.A \sqcap \exists r.B \sqcap (\leq 1 r) \\ B^* \equiv \exists r.C \\ C \equiv A \sqcap B \quad A \sqsubseteq \top \quad B \sqsubseteq \top \end{array}$$

Geprüft wird die Subsumtionsbeziehung

$$A^* \stackrel{?}{\sqsubseteq} B^*$$

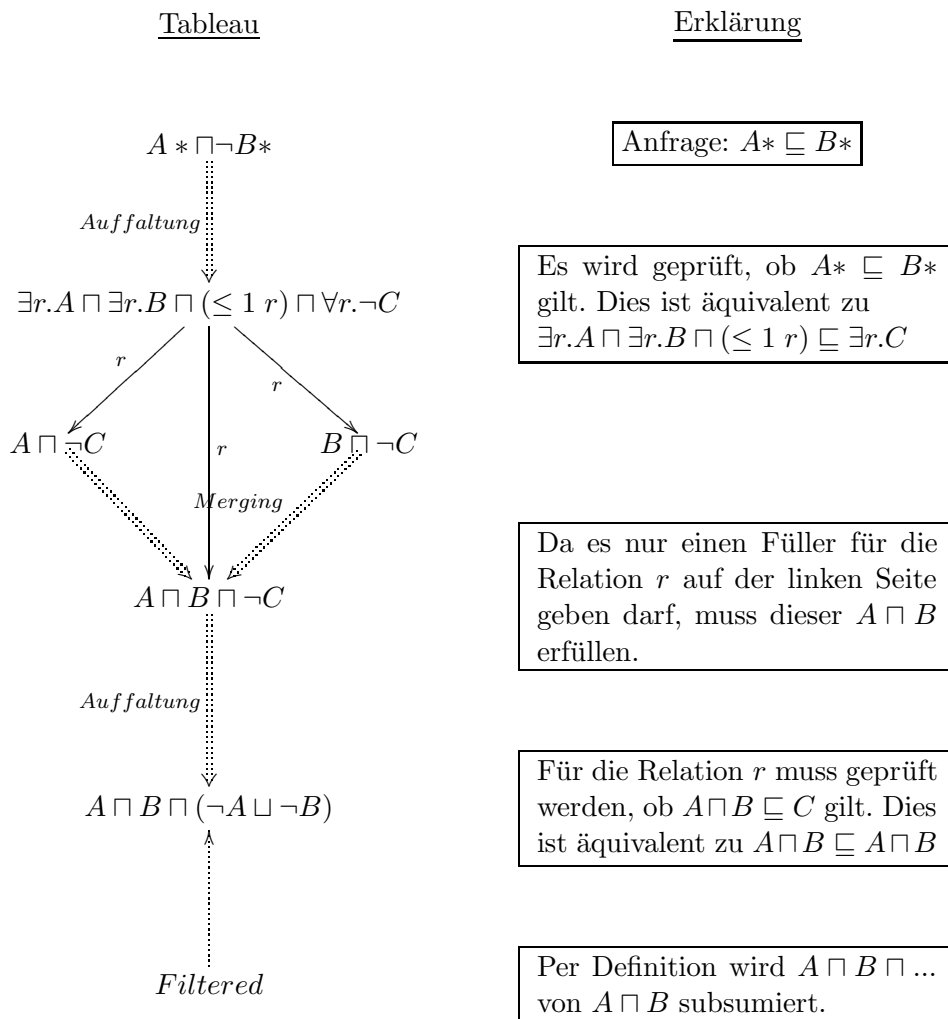


Abbildung 4.8: Komplexes Beispiel 2

Das nächste Beispiel in Abbildung 4.9 illustriert den Beweis und die zugehörige Erklärung für die Subsumtionsbeziehung $Kinderlos \sqcap VaterMitSohn \sqsubseteq Vater$. Diese ist gültig, da der Subsumee äquivalent zu \perp ist und somit von allem anderen subsumiert wird. Der Clash im Tableau kommt durch widersprüchliche Kardinalitätseinschränkungen auf eine Relation und deren Subrelation zustande.

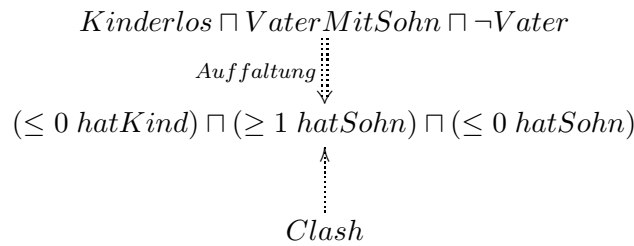
Das Beispiel basiert auf folgender TBox:

hatKind ist eine Relation
hatSohn ist eine Sub-Relation von *hatKind*
Kinderlos $\equiv (\leq 0 \text{ hatKind})$
VaterMitSohn $\equiv (\geq 1 \text{ hatSohn})$
Vater $\equiv (\geq 1 \text{ hatKind})$

Geprüft wird die Subsumtionsbeziehung

$Kinderlos \sqcap VaterMitSohn \stackrel{?}{\sqsubseteq} Vater$

Tableau



Erklärung

Es wird geprüft, ob $Kinderlos \sqcap VaterMitSohn \sqsubseteq Vater$ gilt. Dies ist äquivalent zu $(\leq 0 \text{ hatKind}) \sqcap (\geq 1 \text{ hatSohn}) \sqsubseteq (\geq 1 \text{ hatSohn})$

Es können nicht gleichzeitig mehr als 1 Füller für die Subrelation *hatSohn* und weniger als 0 Füller für die Relation *hatKind* existieren. Daher ist die linke Seite äquivalent zum speziellsten Konzept \perp und dieses wird per Definition von allem anderen subsumiert.

Abbildung 4.9: Komplexes Beispiel 4

Im folgenden Beispiel spielen die Behandlung einer Disjunktion und der Einfluss einer range-Restriktion eine Rolle. Da sowohl der Tableau als auch die Erklärung durch die längeren Konzeptnamen sehr viel voluminöser sind als bei den vorangehenden Beispielen, wurden diese in zwei Abbildungen aufgeteilt. Abbildung 4.10 zeigt den Tableau, Abbildung 4.11 die zugehörige Erklärung.

Das Beispiel basiert auf folgender TBox:

hatKind ist eine Relation mit *range* *Mensch*
Wohlhabende \equiv *Frau* \sqcap *Reich*
Mensch \sqsubseteq \top
Frau \sqsubseteq \top
Genie \sqsubseteq \top
Reich \sqsubseteq \top

Geprüft wird die Subsumtionsbeziehung

Wohlhabende \sqcap \exists *hatKind.Genie* $\stackrel{?}{\sqsubseteq}$ *Frau* \sqcap \exists *hatKind.Mensch*

Tableau

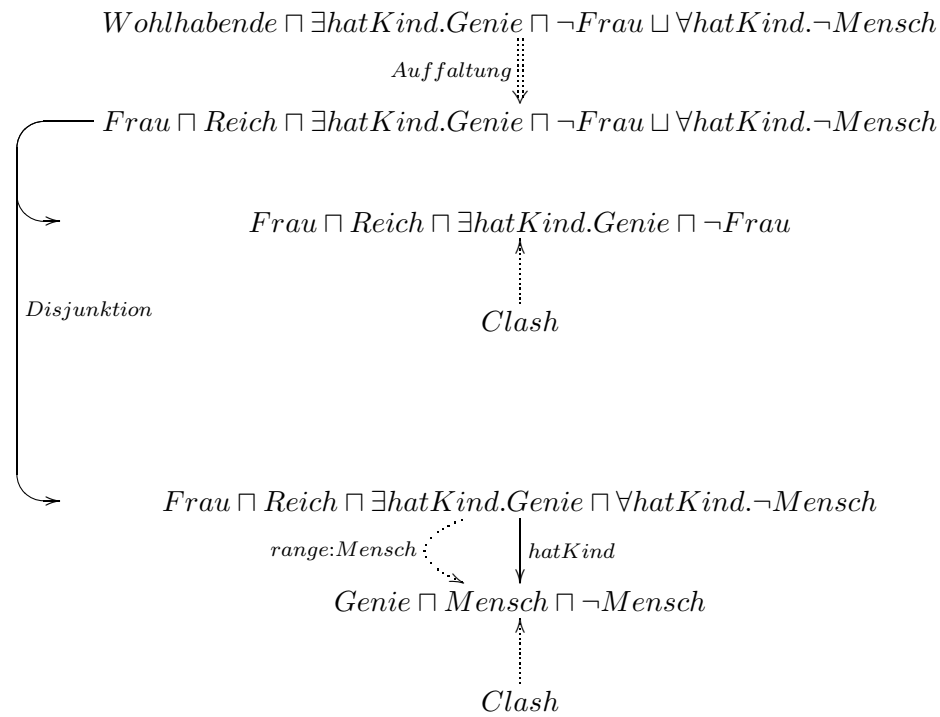


Abbildung 4.10: Komplexes Beispiel 3 - Tableau

Erklärung

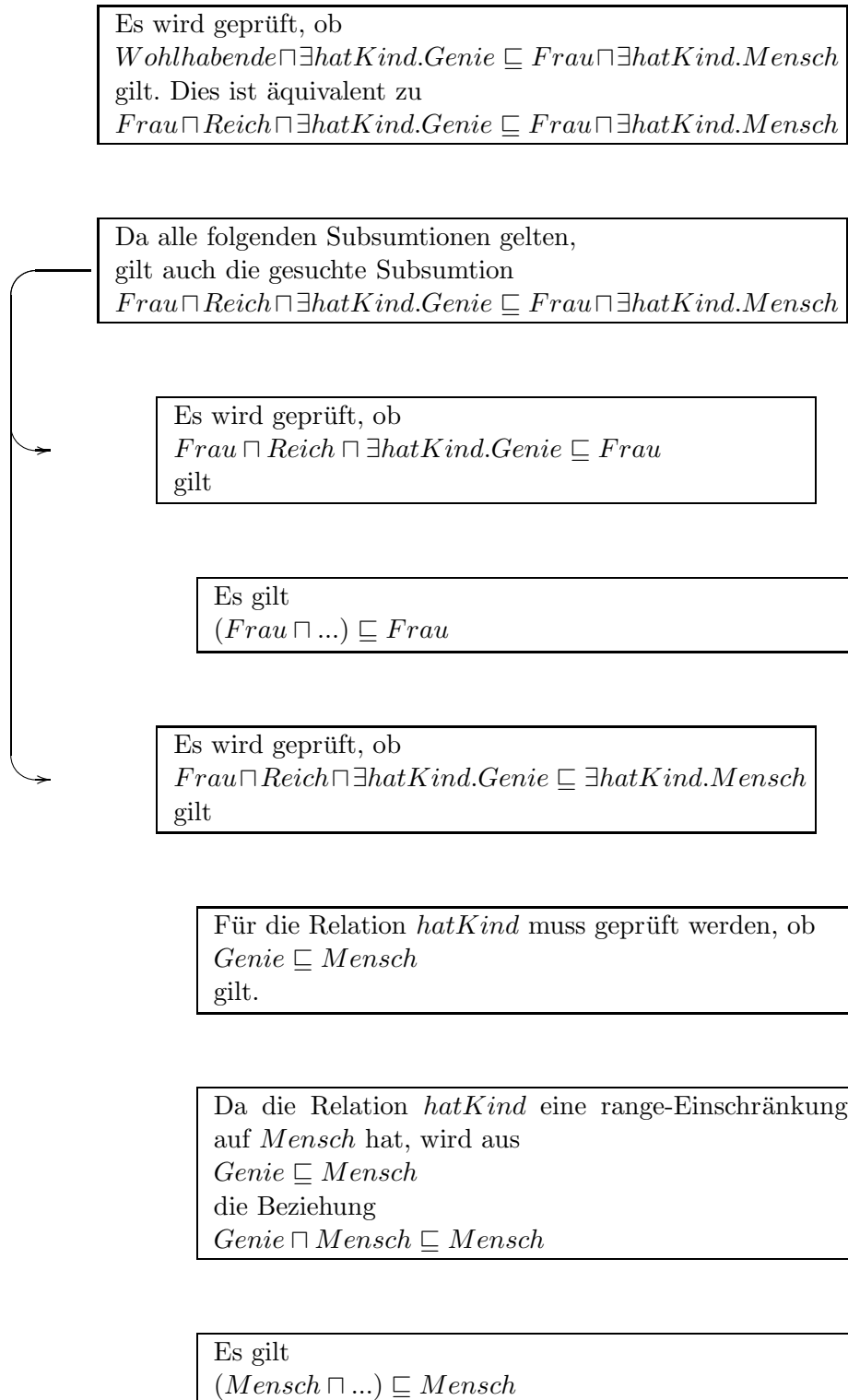


Abbildung 4.11: Komplexes Beispiel 3 - Erklärung

Das Beispiel in Abbildung 4.12 und 4.13 zeigt die Behandlung von transitiven Relationen im Tableau und die eine entsprechende Erklärung.

Der fünfte Erklärungsschritt ist für den Clash, und somit auch für die Erklärung, nicht relevant. Die in Abschnitt 4.5.3 diskutierte Optimierung würde diesen Schritt nicht einfügen und das im Nachfolgerknoten hinzugefügte Konstrukt $\exists \text{vorfahre}.\text{Adliger}$ ebenfalls ausblenden.

Es wird von folgender TBox ausgegangen:

vorfahre ist eine transitive Relation $\text{Koenig} \sqsubseteq \exists \text{vorfahre}.\text{Adliger}$ $\text{Adliger} \sqsubseteq \top$

Gepüft wird die Subsumtionsbeziehung

$\exists \text{vorfahre}.\text{Koenig} \stackrel{?}{\sqsubseteq} \exists \text{vorfahre}.\text{Adliger}$
--

Tableau

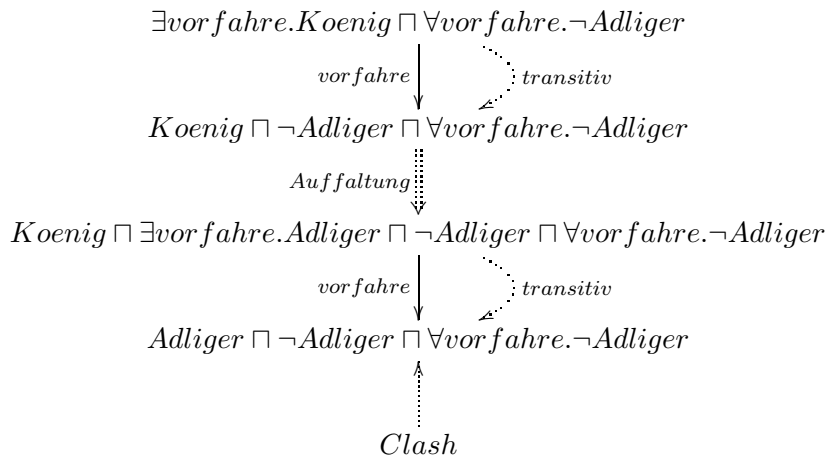


Abbildung 4.12: Komplexes Beispiel 5 - Tableau

Erklärung

Es wird geprüft, ob
 $\exists \text{vorfahre.Koenig} \sqsubseteq \exists \text{vorfahre.Adliger}$
 gilt.

Da *vorfahre* transitiv definiert ist, kann *Adliger* auf der rechten Seite direkt oder von einem beliebigen Nachfolger über *vorfahre* erfüllt werden. Dazu muss der direkte Nachfolger $\text{Adliger} \sqcup \exists \text{vorfahre.Adliger}$ auf der rechten Seite erfüllen.

Für die Relation *vorfahre* muss geprüft werden, ob
 $\text{Koenig} \sqsubseteq \text{Adliger} \sqcup \exists \text{vorfahre.Adliger}$
 gilt. Dies ist äquivalent zu $\text{Koenig} \sqcap \exists \text{vorfahre.Adliger} \sqsubseteq \text{Adliger} \sqcup \exists \text{vorfahre.Adliger}$

Da *vorfahre* transitiv definiert ist, kann *Adliger* auf der rechten Seite direkt oder von einem beliebigen Nachfolger über *vorfahre* erfüllt werden. Dazu muss der direkte Nachfolger $\text{Adliger} \sqcup \exists \text{vorfahre.Adliger}$ auf der rechten Seite erfüllen.

Für die Relation *vorfahre* muss geprüft werden, ob
 $\text{Adliger} \sqsubseteq \text{Adliger} \sqcup \exists \text{vorfahre.Adliger}$
 gilt.

Es gilt
 $(\text{Adliger} \sqcap \dots) \sqsubseteq \text{Adliger}$

Abbildung 4.13: Komplexes Beispiel 5 - Erklärung

4.5 Diskussion

Die vorgestellten Methoden, einen Tableau-Beweis zu erklären, sind nicht immer ideal. Für viele Situationen ließen sich bessere Wege finden, diese zu erklären. In diesem Abschnitt werden einige Ansätze diskutiert, die das Erklären von tableaubasierten Subsumtionsbeweisen verbessern können.

4.5.1 Reine Subsumtionserklärung vs. verschiedene Erklärungs-Modi

Die beschriebenen Erklärungsschritte beziehen sich alle auf reine Subsumtionsbeweise. Dem Benutzer werden bei der Erklärung nur Subsumtionsbeziehungen gezeigt. Dies kann unter Umständen zu verwirrenden Ergebnissen führen. Ist eine der beiden Seiten einzeln betrachtet unerfüllbar, so gilt die Subsumtionsbeziehung. Für die linke Seite bedeutet dies eine Äquivalenz mit \perp , welches von allem subsumiert wird. Ist die rechte Seite im Tableau äquivalent zu \perp , bedeutet dies eine Äquivalenz mit \top im ursprünglichen Term. \top subsumiert per Definition alles andere, die Subsumtionsbeziehung gilt demnach ebenfalls. Die Äquivalenz einer Seite mit \perp ist jedoch nicht zwingend offensichtlich. Unter Umständen benötigt es viele weitere Erklärungsschritte, um diese zu begründen. Das folgende Beispiel (Abbildung 4.14) zeigt einen solchen Fall und die entsprechende Subsumtionserklärung.

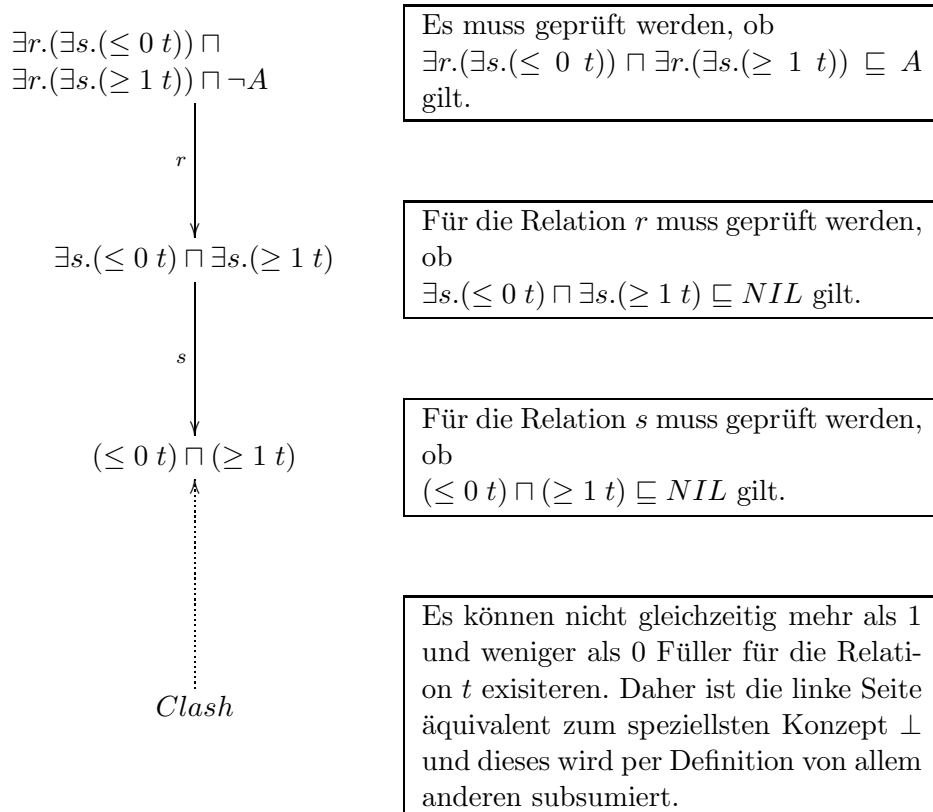


Abbildung 4.14: Subsumtionserklärung bei Unerfüllbarkeit auf einer Seite des Tableaus

Es ist deutlich zu erkennen, dass die Erklärung für den Beweis nicht optimal ist. Ab dem zweiten Schritt fehlen auf der rechten Seite jegliche Konstrukte. Dies wird durch das Schlüsselwort *NIL* zum Ausdruck gebracht. Dennoch gilt die ursprüngliche Subsumtionsbeziehung, da bereits im ersten Schritt der Subsumtee äquivalent zu \perp ist. Eine sehr viel bessere Erklärung wird erreicht, wenn dies bereits im ersten Schritt angemerkt und daraufhin nur noch diese Äquivalenz der linken Seite mit \perp erklärt wird. Der gleiche Tableau mit entsprechender Erklärung wird in Abbildung 4.15 dargestellt.

Um eine derartige Erklärung generieren zu können, werden einige Erweiterungen benötigt. Zunächst muss zu jedem Knoten gespeichert werden, auf welche Art dessen Unerfüllbarkeit erklärt werden soll. Um dies zu entscheiden, müssen vor der weiteren Evaluierung des Knotens beide Seiten unabhängig voneinander auf Unerfüllbarkeit überprüft werden. Dies erhöht die Komplexität des Algorithmus in hohem Maße. Ist eine Seite einzeln betrachtet bereits unerfüllbar, so muss der Erklärungsmodus für diese geändert werden. Für die linke Seite muss im Nachfolgenden deren Unerfüllbarkeit, für die rechte deren Äquivalenz mit \top erklärt werden. Zusätzlich sollte ein Erklärungsschritt eingefügt werden, der darauf hinweist, dass die Subsumtionsbeziehung gilt, da die entsprechende Seite äquivalent zu \perp bzw. \top ist, und dass im nachfolgenden

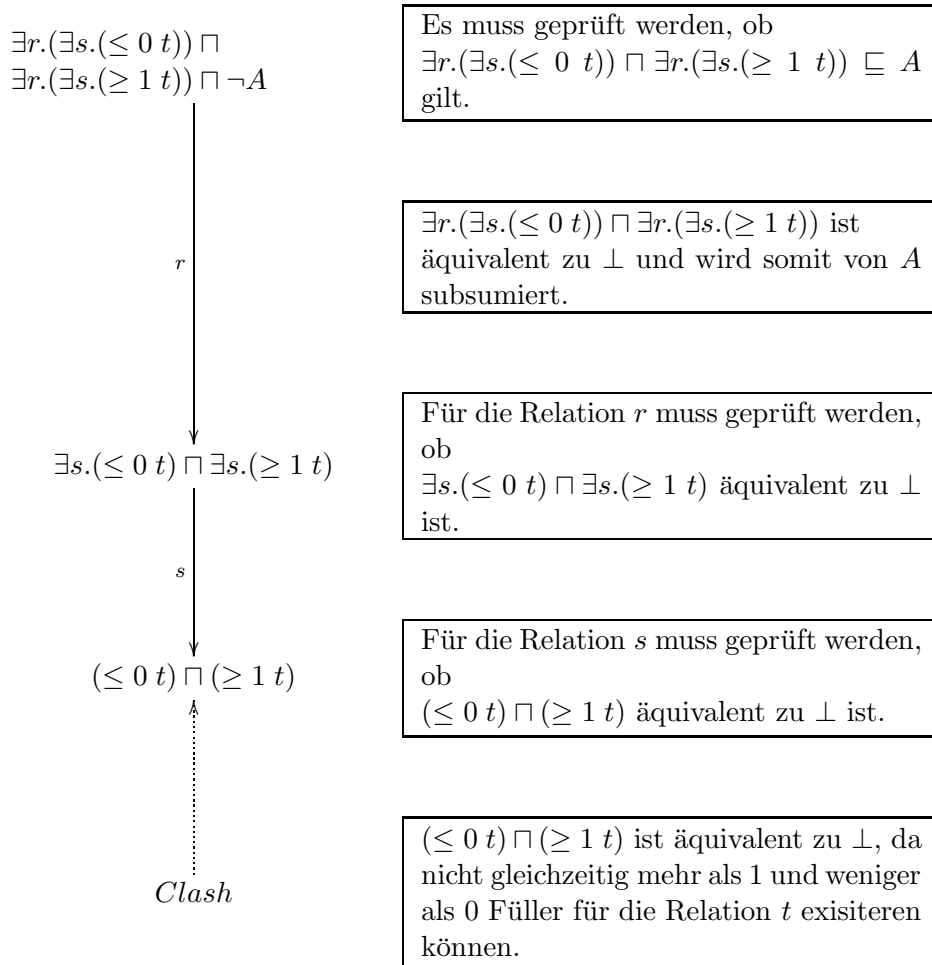


Abbildung 4.15: Unerfüllbarkeitserklärung für eine Seite des Tableaus

diese Äquivalenz erklärt wird.

Bei der Erstellung der einzelnen Erklärungsschritte muss nun berücksichtigt werden, in welcher Form der aktuelle Knoten zu erklären ist. Die Erklärung muss dem Erklärungsmodus entsprechend formuliert werden.

4.5.2 Filtern von besonderen Situationen

Oft lassen sich spezielle Situationen besser erklären als es eine allgemeine Erklärung für den entsprechenden Tableau-Beweis kann. Um solche Situationen zu erkennen, benötigt man Filterfunktionen. Diese untersuchen den aktuellen Term auf Situationen, für welche intuitivere Erklärungen bekannt sind.

Ein gutes Beispiel für eine solche Situation ist die Subsumtionsbeziehung

$$A \sqcap B \sqcap C \sqsubseteq A \sqcap B$$

Diese ist offensichtlich gültig, da der Subsumee dem Subsumer plus weiteren Einschränkungen entspricht. Der bisherige Erklärungsalgorithmus würde dafür einen Tableau mit einer Disjunktion auf der rechten Seite aufbauen. Die Erklärung würde dementsprechend umständlich ausfallen. Besser ist es in diesem Fall, die Gültigkeit der Subsumtionsbeziehung direkt zu begründen und auf weitere Erklärungsschritte zu verzichten. Ein diese Situation erkennender Filter kann durch einen strukturellen Vergleich von Subsumee und Subsumer realisiert werden. Werden dabei alle Konstrukte, welche im Subsumer enthalten sind, auch im Subsumee gefunden, so ist eine Erklärung der Form

Per Definition wird $A \sqcap B \sqcap \dots$ von $A \sqcap B$ subsumiert.

ausreichend. Vor einem solchen Vergleich muss der Subsumer in seine ursprüngliche Form umgeformt, das heißt, negiert werden. Um Situationen dieser Art im gesamten Tableau-Beweis erkennen zu können, muss der Vergleich für jeden Knoten vor dessen Evaluation durchgeführt werden. Dieser Vorgang erhöht die Komplexität des gesamten Algorithmus.

Weitere, die Erklärungen deutlich verbessernde Filter, können realisiert werden, wenn die Erklärung erst nach dem vollständigen Aufbau des Tableau generiert wird. Beispielweise lassen sich dabei mehrere Schritte über inverse Rollen, die nachfolgend zu einem Clash führen, zusammenfassen. Insbesondere beim Erklären von Subsumtionsbeziehungen in ausdrucksmächtigeren Sprachen sind Optimierungen durch Filtern bestimmter Situationen wichtig, um die Erklärungen verständlich zu halten.

4.5.3 Ausblenden nicht relevanter Konstrukte

Beim Aufbau eines Tableau wird jegliche verfügbare Information berücksichtigt. Die Terme in den einzelnen Knoten können dadurch sehr groß werden. Dies führt zwangsläufig zu weniger Übersicht und schlechterem Verständnis des Benutzers. Ein Großteil der Konstrukte ist jedoch für die Gültigkeit der Subsumtionsbeziehung nicht relevant.

Die Qualität der Erklärung kann verbessert werden indem nur relevante Konstrukte dargestellt werden. In [BFH⁺99] wird eine Funktion *Relevant* beschrieben, die aus einem fertigen Tableau die relevanten Beweisschritte und Terme für jede Seite extrahiert. Dies setzt allerdings voraus, dass der Tableau bereits vollständig aufgebaut wurde.

Doch auch wenn die Erklärung bereits während des Tableau-Aufbaus generiert wird, können einige irrelevante Teile ausgeblendet werden. Erfüllbare Teilbäume des Tableaus können beim Zusammensetzen der einzelnen Erklärungsschritte während des rekursiven Aufstiegs abgeschnitten werden. Diese Maßnahme ist unumgänglich, da andernfalls Erklärungen für ungültige Subsumtionsbeziehungen geliefert würden.

Bessere Erklärungen zu ausgeblendeten Konstrukten können erreicht werden, indem zu jedem Konstrukt dessen Ursprung gespeichert wird. Mit Ursprung ist in diesem Zusammenhang das Sprachkonstrukt gemeint, welches dazu geführt hat, dass das Konstrukt im aktuellen Knoten berücksichtigt werden muss. So können Konstrukte, die durch GCI's, domain- oder range-Restriktionen zum Knoten hinzugefügt wurden, speziell behandelt werden. Zunächst werden sie nicht dargestellt. Sind sie jedoch für die nachfolgende Erklärung relevant, so werden sie eingeblendet und ihre Existenz über den jeweiligen Ursprung erklärt.

4.5.4 Komplexes Merging

Bei Kardinalitätseinschränkungen, die Werte größer 1 zulassen, kann komplexes Merging notwendig werden, um den Tableau-Beweis zu führen. Wird beispielsweise in einem Knoten

$$\exists r.A1 \sqcap \exists r.A2 \sqcap \exists r.A3 \sqcap \exists r.A4 \sqcap (\geq 2r) \sqcap (\leq 3r)$$

gefordert, so müssen alle Kombinationen von 2 bis 3 möglichen Nachfolgerknoten geprüft werden, wobei jedes Konzept $A1$, $A2$, $A3$ und $A4$ von mindestens einem Nachfolger erfüllt werden muss.

Jede Möglichkeit diese Nachfolger zu kombinieren, muss zu einem Clash führen, um die Gültigkeit der ursprünglichen Subsumtionsanfrage zu beweisen. Eine Erklärung für jeden dieser vielen Teilbeweise zu generieren, wird sich als wenig praktikabel erweisen. Vielmehr sollte versucht werden, eine Begründungen für die Gültigkeit der ursprünglichen Subsumtionsbeziehung zu finden. Dies könnte ähnlich wie in 4.5.2 beschrieben durch Filtern des vorhergehenden Terms erreicht werden. Im Rahmen dieser Arbeit wurde das Erklären von komplexen Merging Vorgängen nicht weiter untersucht. Es zeigt sich jedoch durch die vorangehenden Überlegungen, dass dies eins der Hauptprobleme beim Erklären von umfangreicheren Beschreibungslogiken, etwa der OWL DL entsprechenden Logik *SHOIN*, darstellen wird.

5 Umsetzung

Dieses Kapitel beschreibt die Implementierung einer Erklärungskomponente. Es werden die verwendeten Datenstrukturen sowie der zugrunde gelegte Sprachumfang dargestellt. Des weiteren werden einige implementierungsspezifische Optimierungen beschrieben. Abschließend werden die bei der Implementierung gewonnenen Erkenntnisse diskutiert.

5.1 Die Erklärungskomponente

Im Rahmen dieser Diplomarbeit wurde eine Erklärungskomponente für tableaubasierte Subsumtionsbeweise entwickelt. Das Programm, im Nachfolgenden MEX¹ genannt, arbeitet auf einer vorgegebenen T-Box. Eine Subsumtionsanfrage wird mittels Tableau-Verfahren geprüft. Falls die Subsumtionsbeziehung gilt, wird eine Erklärung für deren Gültigkeit ausgegeben. Die Erklärung besteht aus einer Liste von Erklärungsschritten, welche nach den in Abschnitt 4 dargestellten Prinzipien generiert werden. Es handelt sich bei MEX um einen Prototypen. Das Hauptaugenmerk wurde auf die automatische Generierung von möglichst verständlichen Erklärungen gelegt. Die Performance wurde nur am Rande berücksichtigt. So wurden ausser Lazy-Unfolding (siehe Abschnitt 3.3.1) keine Tableau-Optimierungsstrategien eingesetzt, da diese den Erklärungsprozess erheblich erschwert hätten. Aufgrund des relativ geringen Sprachumfangs (siehe Abschnitt 5.4), für welchen MEX beweisen und Erklärungen liefern kann, ist der Performanceaspekt ohnehin kaum relevant.

5.2 Entwicklungsumgebung

MEX wurde in Lisp, genauer unter Allegro Common Lisp 6.2², entwickelt. Als IDE wurde Emacs unter Linux verwendet. Die Visualisierungskomponente (siehe Abschnitt 5.7) wurde in Java³ entwickelt.

5.3 Datenstruktur

In diesem Abschnitt werden die bei der Implementierung von MEX verwendeten Datenstrukturen beschrieben. Um Speicherproblemen vorzubeugen, wurde

¹Der Name MEX entstand in einer sehr frühen Entwicklungsphase durch die Abkürzung des Dateinamens “my-explaining.lisp”

²<http://www.franz.com>

³<http://java.sun.com>

versucht, die zu bearbeitende Datenmenge möglichst gering zu halten. Dabei wurde auf die Speicherung von zusätzlicher Information, wie zum Beispiel der Herkunft der einzelnen Konstrukte innerhalb eines Tableau, verzichtet.

5.3.1 T-Box

Konzept- und Rollendefinitionen werden über deren Namen referenziert in Hash-Tabellen gespeichert. Für vollständig definierte Konzepte, partiell definierte Konzepte und Rollen existiert je eine dieser Hash-Tabellen. Zusammen repräsentieren sie die T-Box, auf welcher der Algorithmus arbeitet (Abbildung 5.1). Konzeptdefinitionen sind Listen, bei welchen das erste Element ein Konstruktor ist, zum Beispiel `some` für \exists , die restlichen Elemente liefern die Argumente für diesen Konstruktor. Die Definition $C \doteq D \sqcap \exists r.E$ wird beispielsweise in der Hash-Tabelle für vollständig definierte Konzepte an der Stelle C als ein Eintrag der Form $(and D (some r E))$ gespeichert.

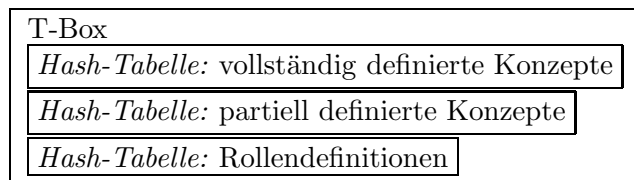


Abbildung 5.1: Repräsentation der T-Box durch drei Hash-Tabellen

5.3.2 Rollendefinition

Die Definition einer Rolle wird durch ein lisp *struct* wie in Abbildung 5.2 repräsentiert. Die Möglichkeit inverse Rollen zu definieren ist im aktuellen Entwicklungsstand nicht gegeben, da MEX nicht mit inversen Rollen umgehen kann. Es wäre jedoch durchaus denkbar, MEX um diese Funktionalität zu erweitern (siehe 4.3.12). Dazu wurde bereits ein slot für die Inverse einer Rolle in der Definition reserviert. Die Listen für die Speicherung der Subrollen (sub-relations) bzw. Superrollen (parent-relations) stellen redundante Informationen dar, da man die Liste der Superrollen auch über eine Suche in allen Rollendefinitionen nach der entsprechenden Subrolle generieren könnte. Dennoch werden die Beziehung zu anderen Rollen in beide Richtungen gespeichert. Dies dient dazu, die Generierung der einzelnen Erklärungsschritte während des Aufbaus des Tableaus zu beschleunigen.

5.3.3 Constraint

Ein Knoten im Tableau wird durch ein Constraint repräsentiert. Der Aufbau eines solchen, ebenfalls in einem struct gespeicherten Constraints, ist in Abbildung 5.3 dargestellt.

Die rechte und linke Seite des zu bearbeitenden Terms werden wie in 4.1

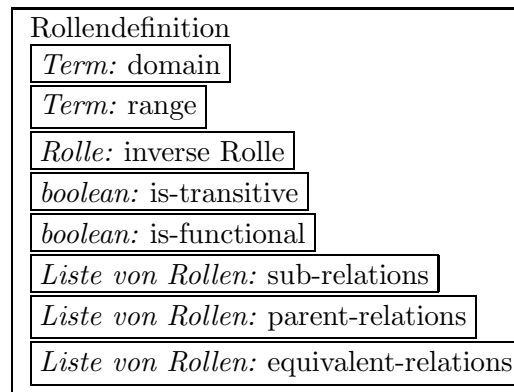


Abbildung 5.2: Repräsentation einer Rollendefinition

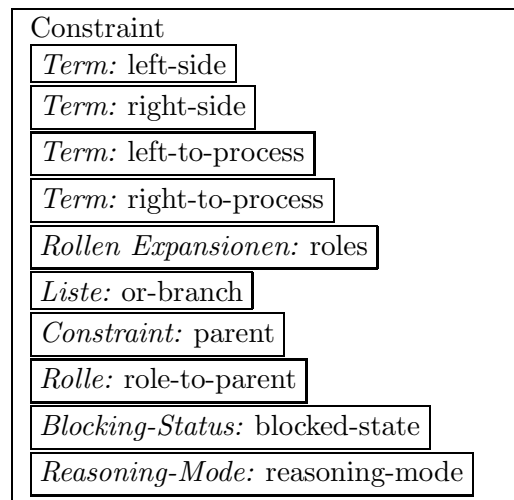


Abbildung 5.3: Repräsentation eines Constraints

beschrieben getrennt gespeichert. Die *to-process*-slots werden zur Weiterverarbeitung verwendet.

Rollenexpansionen werden in einer Liste gespeichert. Diese enthält für jede Rolle einen minimalen und einen maximalen Kardinalitätswert sowie eine Liste von Constraints, welche die Nachfolgerknoten über diese Rolle repräsentieren. Widersprüchliche Kardinalitätseinschränkungen für eine Rolle lassen sich durch diese Repräsentation schnell und effektiv erkennen.

Wird in einem Knoten eine Disjunktion erkannt, muss für jedes Disjunkt ein Clash gefunden werden. In der constraintbasierten Implementierung wird dies erreicht, indem für jedes Disjunkt ein neues Constraint erzeugt wird. Die disjunktiv referenzierten Constraints werden im slot *or-branch* in einer Liste gespeichert. Der Algorithmus versucht nun einen Clash in jedem dieser Nachfolgerknoten zu finden.

Für jedes Constraint wird zusätzlich das Vorgängerconstraint sowie die Rolle, über welche die beiden Constraints in Verbindung stehen, gespeichert. Diese

Information ist für das Erstellen einiger Erklärungsschritte notwendig. Um MEX um die Fähigkeit, inverse Rollen behandeln zu können, zu erweitern, wird ebenfalls ein Verweis auf das Vorgängerconstraint benötigt.

Der slot *blocking-state* gibt an, ob der Knoten geblockt ist. Für dynamisches Blocking (siehe 3.2) kann zusätzlich gespeichert werden, ob der Knoten bereits einmal geblocked war.

reasoning-mode gibt an, in welcher Form die Erklärung für den Knoten generiert werden soll. Dies dient der in 4.2.1 beschriebenen Vorgehensweise.

5.3.4 Erklärungsschritt

Die von MEX generierte Erklärung für eine gültige Subsumtionsbeziehung ist eine Liste von einzelnen Erklärungsschritten. Diese werden nach den in 4 dargestellten Prinzipien zusammengestellt. Ein Erklärungsschritt wird durch die in Abbildung 5.4 gezeigte Struktur repräsentiert.

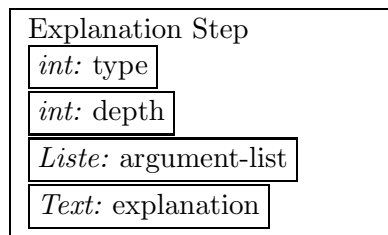


Abbildung 5.4: Repräsentation eines Erklärungsschritts

Jeder Art von Erklärungsschritt wurde eine eindeutige Typ-id zugeordnet. Diese kann dazu verwendet werden, bestimmte Schritte zu filtern oder individuell zu markieren. In Verbindung mit der Liste von Argumenten können externe Programme, die eine Erklärung von MEX geliefert bekommen, eigene Formulierungen für die Erklärungsschritte entwickeln. In der Argumenteliste werden relevante Daten zum jeweiligen Erklärungsschritt gespeichert. Welche dies sind, hängt vom Typ des Erklärungsschrittes ab. Die Daten wurden so gewählt, dass die textuelle Erklärung aus ihnen generiert werden kann.

Zu jedem Schritt wird zusätzlich die Erklärungstiefe gespeichert. Diese kann verwendet werden, um die gesamte Erklärung in einer baumartigen Struktur darzustellen.

Im *explanation*-slot wird eine natürlichsprachliche Erklärung für den aktuellen Schritt abgelegt. Diese entspricht den Formulierungen aus Abschnitt 4. Ziel bei der Entwicklung von MEX war es, durch Aneinanderreihung der natürlichsprachlichen Einzelerklärungen eine möglichst intuitiv verständliche Gesamterklärung für die Gültigkeit der Subsumtionsbeziehung zu erreichen.

5.4 Sprachumfang

Der von MEX verarbeitbare Sprachumfang entspricht einer Untermenge der durch OWL Lite definierten Beschreibungslogik. Für den Aufbau der T-Box werden Konstruktoren für

- partielle Konzeptdefinition
- vollständige Konzeptdefinition
- Definition von Rollen
- Definition von Subrollen
- Definition von equivalenten Rollen
- Definition von transitiven Rollen
- Definition von funktionalen Rollen
- Definition von domain- und range-Restriktionen für Rollen

zur Verfügung gestellt. Für Konzeptdefinitionen können die folgende Sprachkonstrukte verwendet werden

- Konjunktion ($C \sqcap D$)
- qualifizierte Existenzquantoren ($\exists r.C$)
- qualifizierte Allquantoren ($\forall r.C$)
- Kardinalitätseinschränkungen ($\leq n r$), ($\geq n r$), ($= n r$) mit $n \in [0, 1]$

Auf die Behandlung von inversen Rollen und General Concept Inclusions (GCIs) wurde verzichtet.

Der resultierende Sprachumfang lässt sich durch die Sprache $\mathcal{AL}\mathcal{E}\mathcal{H}\mathcal{F}_{R^+}$ erweitern um globale domain- und range-Restriktionen beschreiben.

Reasoning über inverse Rollen erfordert dynamisches Blocking und steigert die Komplexität des Tableau-Verfahrens. Erklärungen für die durch inverse Rollen entstehenden internen Prozesse zu generieren, gestaltet sich dementsprechend schwierig. In Abschnitt 4.3.12 werden einige theoretische Ansätze dazu beschrieben. Die Umsetzung dieser hätte jedoch den zeitlichen Rahmen dieser Arbeit gesprengt.

Um GCIs im Tableau-Beweis zu berücksichtigen, werden die durch sie definierten Axiome typischerweise jedem Knoten hinzugefügt. Dies führt zu sehr großen Termen in den einzelnen Tableau-Knoten. Daraus resultiert ein dramatischer Performanceverlust für den Reasoningprozess. Für das Erklären des Beweises wäre eine solche Behandlung von GCIs fatal. Die einzelnen Erklärungsschritte würden durch die zusätzliche Information der durch GCIs

hinzugefügten Konstrukte unübersichtlich lang. Auch wäre nicht intuitiv verständlich, woher die einzelnen Konstrukte kommen. Eigene Erklärungsschritte für die Berücksichtigung der GCIs wären nötig. Bei großen T-Boxen werden unter Umständen sehr viele GCIs definiert. Die Auswirkungen eines jeden GCIs auf die einzelnen Tableau-Schritte zu erklären, würde die Erklärung völlig unübersichtlich und somit unbrauchbar machen.

Denkbar wäre es, die durch GCIs hinzugefügten Konstrukte zwar im Tableau zu berücksichtigen, sie in der Erklärung jedoch nur darzustellen, wenn sie zu einem Clash etwas beigetragen haben. Die Idee, die im Tableau vorkommenden Terme nur teilweise für die Erklärung zu verwenden, wird in 4.5 genauer diskutiert. Mit der in MEX verwendeten Datenstruktur ist dieser Ansatz jedoch nicht realisierbar.

Eine effektivere Möglichkeit, GCIs im Tableau-Beweis zu berücksichtigen, bietet die in [HT00b] beschriebene Optimierungstechnik der Absorption. Dabei werden GCIs in einem vorverarbeitenden Prozess zu einfachen Subsumtionsbeziehungen umgeformt, bei welchen auf der linken Seite nur primitive Konzepte stehen dürfen. Dadurch müssen nicht mehr alle GCIs jedem Tableau-Knoten hinzugefügt werden. Dies bewirkt eine geringere Komplexität des Algorithmus und somit eine höhere Performance. Das Generieren von Erklärungen wird dadurch jedoch nicht einfacher. Die Menge der zusätzlichen Konstrukte wird zwar kleiner, diese zu erklären aber komplizierter, da sie nicht mehr den ursprünglichen GCIs entsprechen.

5.5 Allgemeine Vorgehensweise

Um eine Subsumtionsbeziehung zu beweisen, baut MEX nach den in Kapitel 3 beschriebenen Prinzipien einen Tableau auf. Parallel dazu wird on-the-fly eine Erklärung in Form einer Liste von Erklärungsschritten erstellt. Zu jedem relevanten Tableau-Schritt wird ein Erklärungsschritt generiert und der Liste hinzugefügt. Die Generierung der einzelnen Erklärungsschritte erfolgt nach dem in Kapitel 4 beschriebenen Verfahren.

Die Gültigkeit der Subsumtionsbeziehung ist bewiesen, sobald dem initialen Constraint Unerfüllbarkeit nachgewiesen werden konnte. Dies ist der Fall, wenn ein Widerspruch (Clash) im Knoten selbst gefunden wurde. Ist dies nicht möglich, so werden die Nachfolgerconstraints sukzessiv auf Clashes durchsucht. Nachfolger, welche komplett expandiert wurden, ohne dass ein Clash gefunden werden konnte, demnach also erfüllbar sind, werden für die Erklärung nicht weiter berücksichtigt. Beim rekursiven Aufstieg wird die Teilerklärung zu erfüllbaren Nachfolgern schicht abgeschnitten. Der erste Relations-Nachfolger, dem Unerfüllbarkeit nachgewiesen werden konnte, wird zur Erklärung herangezogen. Da sich diese Unerfüllbarkeit auf den Vorgängerknoten überträgt, müssen keine weiteren Nachfolger untersucht werden. Dennoch sind Fälle denkbar, in welchen mehrere Nachfolger unerfüllbar, die Erklärungen für diese jedoch unterschiedlich umfangreich sind. In einem solchen Fall wäre es optimal, den Nachfolger, dessen Unerfüllbarkeit (bzw. die daraus resultierende Gültigkeit der Subsumtion) am intuitivsten zu erklären ist, für die Gesamter-

klärung zu verwenden. Die aktuelle Implementierung von MEX wendet diese Optimierung nicht an. Dazu wäre eine Funktion nötig, welche alternative Erklärungen in Bezug auf deren Verständlichkeit bewertet.

5.6 Optimierungen

Um die Performance von tableaubasierten Algorithmen zu verbessern, existieren einige Optimierungsverfahren. Diese werden in [HT00a] ausführlich beschrieben. Bei der Entwicklung von MEX lag das Hauptaugenmerk auf der Generierung möglichst verständlicher Erklärungen. Die meisten Optimierungsverfahren hätten dies erheblich erschwert, da sie die ursprünglichen Strukturen verändern. Aus diesem Grund wurde auf die Anwendung solcher Verfahren verzichtet. Die in MEX verwendeten Optimierungen werden in den folgenden Abschnitten beschrieben.

5.6.1 lazy-unfolding

MEX verwendet die in 3.3.1 beschriebene Technik des lazy-unfolding. Vor der weiteren Verarbeitung eines Constraints werden also nur direkt referenzierte Konzepte aufgefaltet. Konzepte, die erst in einem Nachfolgerknoten zu einem Clash führen können, bleiben vorerst unaufgefaltet.

5.6.2 RACER Anbindung

Optional kann MEX mit dem in 2.5.1 erwähnten Reasoner RACER kommunizieren. Dies führt sowohl zu einer höheren Performance als auch zu verbesserten Erklärungen. Um die Verbingung zwischen MEX und RACER herzustellen, werden beide in die selbe Lisp Umgebung geladen. Somit kann MEX RACER direkt über den Aufruf von dessen Funktionen ansprechen.

Vor der Evaluation eines Constraints wird dessen Erfüllbarkeit mittels RACER überprüft. Wird das Constraint als erfüllbar erkannt, so kann dieses Ergebnis direkt zurückgegeben werden. MEX generiert nur für unerfüllbare Constraints, welche gültigen Subsumtionsbeziehungen entsprechen, Erklärungen. Es ist also nicht nötig, erfüllbare Constraints erneut nach einem Clash zu untersuchen. Durch Anerwendung zahlreicher Optimierungsverfahren ist RACER sehr viel schneller als MEX. Bei der Durchführung eines Tableau-Beweises entstehen typischerweise Teilbäume, die erfüllbar sind. Diese Erfüllbarkeit zu erkennen, kann je nach Größe und Komplexität des Teilbaums sehr aufwändig sein. In solchen Fällen führt die RACER Anbindung zu einer signifikanten Performancesteigerung. Die Erfüllbarkeit eines Teilbaums (evtl. auch des gesamten Tableau) kann von RACER sehr viel schneller erkannt werden als von MEX. Da solche erfüllbaren Teilbäume für die Erklärung nicht relevant sind, müssen sie von MEX nicht weiter bearbeitet werden. Durch die direkte Anbindung fällt die Tatsache, dass zunächst die T-Box an RACER übermittelt werden muss, kaum ins Gewicht.

Zusätzlich werden vor der Evaluation eines Constraints sowohl die linke als auch die rechte Seite mittels RACER auf Unerfüllbarkeit überprüft. Ist die linke Seite unerfüllbar, so gilt die ursprüngliche Subsumtionsbeziehung, da \perp von allem subsumiert wird. Ist die rechte Seite unerfüllbar so gilt sie ebenfalls, da \top alles andere subsumiert. Für diesen Teilbaum muss also eine Erklärung für die Äquivalenz der betroffenen Seite zu \perp bzw. \top geliefert werden. Die andere Seite kann in einem solchen Fall vernachlässigt werden. Die Erklärung für einen solchen Fall wird in 4.2.1 genauer beschrieben. Die zur Erkennung der Unerfüllbarkeit notwendigen zusätzlichen Anfragen an RACER führen zwar zu einem leichten Performanceverlust, erhöhen jedoch die Qualität der Erklärungen.

5.6.3 Filtern von offensichtlichen Beziehungen

Vor der Bearbeitung einer Subsumtionsanfrage prüft MEX, ob eine äquivalente Konzeptdefinition in der T-Box enthalten ist. Ist dies der Fall, so gilt die Subsumtionsbeziehung per Definition. Eine weitere Erklärung wird nicht benötigt.

In einigen Fällen ist die Gültigkeit der Subsumtionsbeziehung bereits an Stellen offensichtlich, an denen das in Kapitel 4 beschriebene Verfahren mit weiteren Erklärungen fortfahren wurde. Ein Beispiel für einen solchen Fall ist die Subsumtionsbeziehung

$$A \sqcap B \sqcap C \sqsubseteq A \sqcap B$$

Es kann davon ausgegangen werden, dass der Benutzer erkennt, dass diese Beziehung gilt. Im Tableau wird der Subsumer als $\neg A \sqcup \neg B$ repräsentiert. Dies führt zu einer Erklärung der gesamten Beziehung über zwei Teilerklärungen, eine für $A \sqcap B \sqcap C \sqsubseteq A$ und eine für $A \sqcap B \sqcap C \sqsubseteq B$. Diese Form der Erklärung ist zwar ebenfalls gültig, in diesem Fall jedoch überflüssig und wenig intuitiv. Sehr viel einfacher ist es, die Gesamtbeziehung als offensichtlich gültig zu erklären.

MEX filtert diesen und ähnliche Fälle, indem vor der Bearbeitung eines Constraints geprüft wird, ob der Subsumee alle Konstrukte des Subsumers enthält. Dazu wird zunächst die rechte Seite negiert, um die ursprüngliche Form zu erhalten. Daraufhin wird für jedes Konstrukt auf rechten Seite ein exakt gleiches auf der linken Seite gesucht. Gelingt dies, so wird frühzeitig abgebrochen und eine intuitivere Erklärung geliefert. Diese Vorgehensweise entspricht einem strukturellen Subsumtionstest, allerdings in vereinfachter Form. So wird in diesem Ansatz nur auf tatsächliche Äquivalenz der Konstrukte geprüft. Die Subsumtionsbeziehung $A \sqcap (\leq 0 r) \sqcap C \sqsubseteq A \sqcap (\leq 0 r)$ wird daher gefiltert, wohingegen $A \sqcap (\leq 0 r) \sqcap C \sqsubseteq A \sqcap (\leq 1 r)$ auf herkömmliche Weise erklärt wird. Konkrete Erklärungsansätze zu gefilterten Situationen werden in 4.5 genauer beschrieben.

5.7 Visualisierung

Als ein Teil dieser Arbeit wurde eine Visualisierungskomponente für die von MEX generierten Erklärungen implementiert. Diese wurde in der Programmier-

sprache Java unter Verwendung des JDK 1.4.2 entwickelt. Die GUI basiert auf den in java.awt und javax.swing zur Verfügung gestellten Klassen.

Die von MEX generierte Erklärung für eine Subsumtionsanfrage wird im XML Format über TCP an die Visualisierungskomponente geschickt. Diese erstellt auf der Basis dieser Daten einen Erklärungsbaum. Die Verzweigungen dieses Baums sind von den in den einzelnen Erklärungsschritten gespeicherten Erklärungstiefen abhängig. Nachdem der aus Java-Objekten bestehende Baum vollständig aufgebaut ist wird er mit Hilfe der Klasse JTree dargestellt. Abbildung 5.5 zeigt einen Screenshot einer solchen Darstellung.

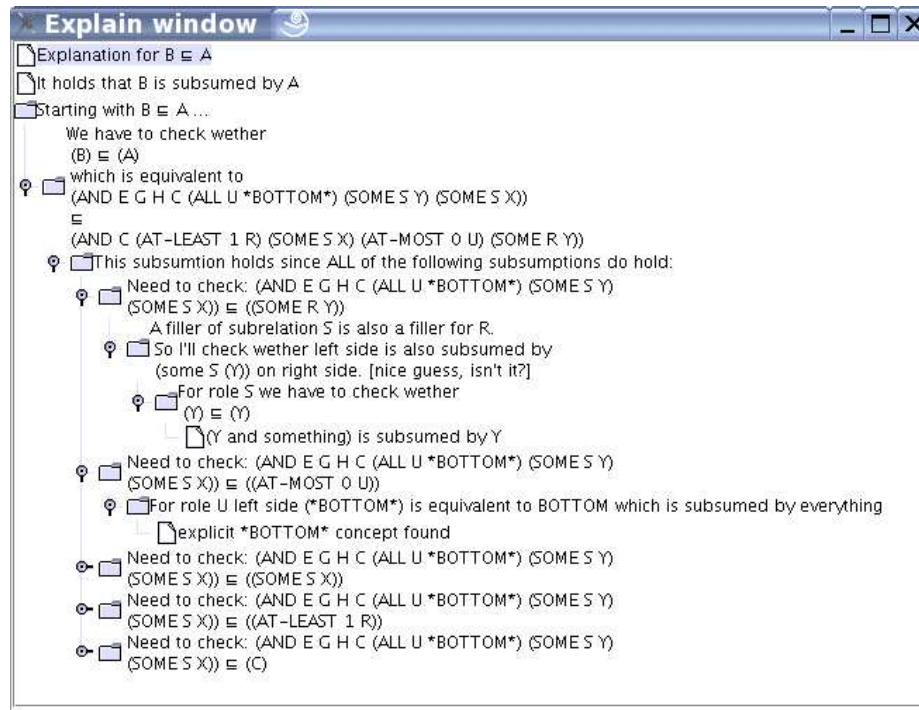


Abbildung 5.5: Screenshot einer visualisierten Erklärung

Diese Form der hierarchischen Darstellung bietet eine klare Strukturierung der Gesamterklärung und unterstützt somit zusätzlich das Verständnis des Benutzers. Des weiteren ist die Möglichkeit gegeben, Teile der Erklärung interaktiv ein- und auszublenden. Dadurch kann der Benutzer genau so viel Information einblenden wie er benötigt. Dies ist insbesondere bei sehr umfangreichen Beweisen hilfreich.

5.8 OntoTrack Anbindung

Die hier beschriebene Visualisierungskomponente wurde in den, an der Universität Ulm entwickelten, Ontologie Editor OntoTrack⁴ integriert. Dies zeigt

⁴Mehr Informationen zu OntoTrack finden sich in [LN03] oder unter <http://www.informatik.uni-ulm.de/ki/ontotrack/>

eine Anwendungsmöglichkeit für das Erklären von Subsumtionsbeweisen. In diesem Fall dient es dem besseren Verständnis von bestehenden Subsumtionsbeziehungen in einer Ontologie. Des Weiteren kann es unterstützend beim Debugging von erstellten Ontologien genutzt werden. In [LN04] werden die ersten Schritte der Anbindung von MEX an OntoTrack beschrieben. Inzwischen wurde sowohl die Funktionalität der Erklärungskomponente MEX als auch die der genannten Visualisierungskomponente stark verbessert. Abbildung 5.6 zeigt, wie ein Benutzer von OntoTrack eine Erklärung für eine bestehende Subsumtionsbeziehung anfordern kann.

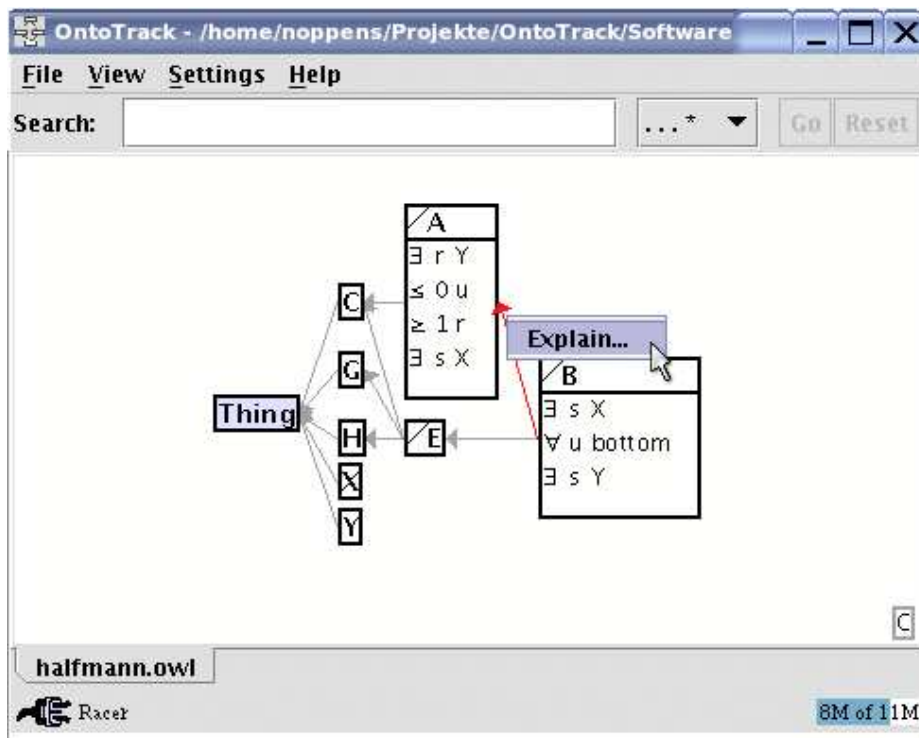


Abbildung 5.6: Screenshot einer Erklärungsanfrage in OntoTrack

Die Kommunikation zwischen OntoTrack und MEX erfolgt über eine TCP-Verbindung. Vor einer Subsumtionsanfrage wird zunächst die in OntoTrack geladene TBox an MEX geschickt. Eine PlugIn-Komponente in OntoTrack generiert die dazu notwendigen Befehle und schickt sie an MEX. Die anschließende Subsumtionsanfrage beantwortet MEX mit einer Erklärung für deren Gültigkeit. Die in OntoTrack integrierte Visualisierungskomponente öffnet daraufhin ein neues Fenster und präsentiert dem Benutzer die Erklärung.

5.9 Erkenntnisse

Aus der prototypischen Implementierung der Erklärungskomponente MEX sind einige Erkenntnisse gewonnen worden. Diese sollten in nachfolgenden Implementierungen ähnlicher Systeme berücksichtigt werden.

Die Entwicklung von MEX hat gezeigt, dass das Erklären von tableaubasierenden Beweisen sehr kompliziert ist. Für das Generieren von Erklärungen werden zusätzliche Informationen zu den einzelnen Konstrukten benötigt. Wichtig ist demnach eine umfangreiche Datenstruktur, die das Speichern solcher Informationen ermöglicht. Um die Erklärungen übersichtlich zu halten, sollten nicht relevante Beweisteile keinen Einfluss auf die Erklärung haben. Dies umfasst sowohl erfüllbare Teilbäume des Tableaus als auch Konstrukte in den einzelnen Knoten, die für deren Unerfüllbarkeit nicht relevant sind.

Des Weiteren hat sich gezeigt, dass eine einfache Liste von Erklärungsschritten nicht die nötige Übersicht bietet. Es empfiehlt sich eine graphbasierte Darstellung der Gesamterklärung. Dabei ist es von Vorteil, wenn der Benutzer die Möglichkeit hat, Teilbeweise interaktiv aus- und einzublenden. Somit können umfangreiche Erklärungen Schritt für Schritt betrachtet und verstanden werden. Für die im Rahmen dieser Arbeit entstandene Visualisierungskomponente wurde eine baumartige Darstellung gewählt. Denkbar sind jedoch auch andere Formen der graphbasierten Darstellung.

6 Bewertung und Ausblick

In diesem Kapitel werden der erarbeitete Ansatz und die im Rahmen dieser Arbeit entstandene Implementierung bewertet. Abschließend werden aktuelle Projekte, die sich ebenfalls mit dem Erklären von beschreibungslogischen Inferenzdiensten beschäftigen, genannt.

6.1 Allgemeine Vorgehensweise

Diese Arbeit hat gezeigt, dass tableaubasierte Subsumtionsbeweise um Explaining-Funktionalität erweitert werden können. Dazu muss das übliche Tableau-Verfahren modifiziert werden. Die Ausprägung dieser Modifikationen hängt in erster Linie von der zu erklärenden Fragestellung ab. Diese Arbeit beschäftigt sich ausschließlich mit der Erklärung von Subsumtionsbeziehungen. Das dargestellte Verfahren basiert auf den in [BFH⁺99] beschriebenen theoretischen Vorarbeiten.

Um tableaubasierte Subsumtionsbeweise erklären zu können, ist es besonders wichtig, die ursprüngliche Bedeutung der im Tableau zum Teil negiert auftretenden Konstrukte zu speichern. Es hat sich als praktikabel erwiesen, die linke (Subsumee) und die rechte Seite (Subsumer) der Terme getrennt zu halten. Alternativ kann zu jedem einzelnen Konstrukt dessen Herkunft gespeichert werden. Dies ist zwar aufwändiger, erlaubt jedoch auch zusätzliche Informationen zu den einzelnen Konstrukten zu speichern. Dadurch können übersichtliche Erklärungen trotz Berücksichtigung von Sprachkonstrukten wie GCI's oder domain- und range-Restriktionen generiert werden. Dies wird in Abschnitt 4.5 genauer beschrieben. Die Entscheidung, in welcher Form die ursprüngliche Bedeutung der Terme gespeichert wird, sollte vom angestrebten Sprachumfang abhängig gemacht werden. Für weniger ausdrucksmächtige Sprachen genügt die einfache Trennung von linker und rechter Seite. Werden jedoch komplexere Konstrukte, wie zum Beispiel GCI's zugelassen, so empfiehlt sich die umständlichere Variante, die Herkunft eines jeden Konstrukts separat zu speichern.

Eine weitere grundsätzliche Entscheidung fordert die Frage, wann die Erklärung generiert wird. Eine Möglichkeit ist es, dies während des Tableau-Beweises zu tun. Dadurch können die angewendeten Regeln direkt erklärt werden. Ebenso wird keine weitere Struktur zum Abspeichern des aufgebauten Tableaus benötigt. Allerdings birgt dieser Ansatz auch einige Nachteile. So können rückwirkend keine Änderungen an der bereits generierten Erklärung vorgenommen werden. Auch können verschiedene Teile des Tableaus nicht gemeinsam analysiert werden. Durch Generieren der Erklärung nach dem Aufbau

des vollständigen Tableau können diese Techniken angewandt werden. Konkrete Ansätze für die Verbesserung der Erklärungen durch eine nachträgliche Bearbeitung werden in Abschnitt 4.5 beschrieben. Die nachträgliche Generierung einer Erklärung für einen bereits existierenden Tableau ist zwar aufwändiger, ermöglicht jedoch auch bessere Erklärungen.

Zusammenfassend lässt sich erkennen, dass sich durch komplexere und schwieriger zu realisierende Verfahren weitaus bessere Erklärungen generieren lassen. Dies betrifft insbesondere die Behandlung von komplizierteren Sprachkonstrukten wie GCI's oder inversen Rollen. Um beispielsweise Erklärungen für den Sprachumfang der Semantic Web Sprache OWL DL zu realisieren, sind erweiterte Vorgehensweisen notwendig. Als größtes Problem wurde das Erklären von komplexem Merging identifiziert.

Ob die mittels des erarbeiteten Verfahrens generierten Erklärungen tatsächlich intuitiv verständlich sind, wurde im Rahmen dieser Arbeit nicht explizit untersucht. Das heißt, es wurden keine Testpersonen mit den Erklärungen konfrontiert, um deren Qualität zu evaluieren. Die eigene Erfahrung im Umgang mit Subsumtionsbeweisen zeigt jedoch, dass der tableaubasierte Ansatz durchaus adäquate Erklärungen liefert. Mit einem gewissen Grundwissen bezüglich Logik und Ontologien sollten somit auch unerfahrene Benutzer in der Lage sein, von den gelieferten Erklärungen zu profitieren.

6.2 Implementierung

Mit der Erklärungskomponente MEX wurden die in Kapitel 4 beschriebenen Ansätze anhand einer prototypischen Implementierung umgesetzt. Die konkrete Vorgehensweise zur Generierung von Erklärungen wurde, basierend auf den theoretischen Überlegungen in [BFH⁺99] und zusätzlichen eigenen Erkenntnissen, entwickelt. Ziel der Implementierung war es, die praktische Umsetzbarkeit des erarbeiteten Verfahrens zu zeigen. Dabei stand die Qualität der generierten Erklärungen im Vordergrund. Die Performance des Systems wurde nur am Rande berücksichtigt.

Es hat sich gezeigt, dass zum tableaubasierten Erklären von Subsumtionsbeweisen eine besonders umfangreiche Datenstruktur benötigt wird. Die in MEX verwendete Struktur erwies sich als zu beschränkt, um qualitativ gute Erklärungen für umfangreichere Sprachen zu generieren. Dies war der Hauptgrund, auf Sprachkonstrukte wie GCI's zu verzichten und sich auf die Sprache $\mathcal{AL}\mathcal{E}\mathcal{H}\mathcal{F}_{\mathcal{R}^+}$ ohne GCI's zu beschränken. Für transitive Rollen und domain- und range-Restriktionen sind die von MEX generierten Erklärungen nicht optimal. Dies liegt daran, dass Erklärungen für diese Konstrukte immer eingefügt werden müssen, auch wenn sie für die Gültigkeit der Subsumtion keine Relevanz haben. Hier könnte eine komplexere Datenstruktur, kombiniert mit nachträglicher Generierung der Erklärungen, eine deutliche Optimierung ermöglichen. Diese und weitere Ideen zur Verbesserung der Erklärungsqualität werden in Kapitel 4.5 genauer beschrieben.

Die in MEX verwendete, relativ schlanke Datenmodellierung wurde gewählt,

um Speicherproblemen vorzubeugen. Da der Tableau-Algorithmus unter Umständen exponentiell viele Knoten generiert, ist dies besonders bei größeren Beweisen ein ernstzunehmender Aspekt. Wird die Datenstruktur erweitert, um bessere Erklärungen generieren zu können, muss der benötigte Speicherplatz umso genauer berücksichtigt werden. So empfiehlt es sich, möglichst selten Objekte zur Datenrepräsentation zu verwenden, da diese typischerweise sehr viel Speicherplatz benötigen. Auch sollten erfüllbare Teilbäume des Tableaus abgeschnitten werden, da sie nicht zur Erklärung benötigt werden.

Die typischen Optimierungsverfahren für das Tableau-Verfahren lassen sich für Reasoner mit Erklärungsfunktionalität kaum einsetzen. Eine Ausnahme ist die Technik des Lazy-Unfolding. Um die Laufzeiten von MEX dennoch in akzeptablen Grenzen zu halten, wurde der hochoptimierte Reasoner RACER angebunden. In welcher Form dies zu einer besseren Performance führt, wird in 5.6.2 beschrieben.

Rückblickend kann MEX als erfolgreiche Implementierung eines Prototyps gesehen werden. Diesen jedoch für einen größeren Sprachumfang wie *SHIF* oder gar *SHOIN* zu erweitern, würde einige Probleme bereiten. Um Erklärungsgenerierung für diese Sprachen zu realisieren, scheint es sinnvoller, die Erklärung erst nach dem Aufbau des Tableaus zu generieren. Auch sollte eine umfangreichere Datenstruktur gewählt werden. Beides führt zu einem hohen Bedarf an Arbeitsspeicher. Auch wenn Speicherplatz- und Laufzeit-Performance große Probleme darstellen, spielen sie eine eher untergeordnete Rolle, da es sich beim Erklären typischerweise um einen on-demand Dienst handelt. Das Hauptproblem liegt vielmehr im Finden geeigneter Erklärungen für bestimmte Situationen. Beispielsweise entstehen durch Kombination von inversen Rollen und qualifizierten Kardinalitätseinschränkungen sehr schwer zu erklärende Situationen. Allgemein lässt sich behaupten, dass je ausdrucks-mächtiger die zugrunde liegende Beschreibungslogik ist, desto schwerer ist es, Erklärungsfunktionalität für deren Inferenzdienste zu entwickeln. In diesem Bereich steckt nach wie vor großes Potential für weitere Forschungen.

6.3 Ähnliche Projekte

In diesem Abschnitt werden einige Projekte vorgestellt, die sich ebenfalls mit dem Erklären von beschreibungslogischen Inferenzen beschäftigen. Diese lassen sich in T-Box-bezogenes Erklären und A-Box-bezogenes Erklären unterteilen. Erstere lassen sich direkt mit dem in dieser Arbeit vorgestellten Ansatz in Verbindung bringen. A-Box-bezogene Erklärungskomponenten hingegen basieren zumeist auf grundlegend unterschiedlichen Verfahren. Dennoch weisen die durch sie behandelten Fragestellungen durchaus interessante Aspekte für zukünftige Projekte im Bereich des T-Box-bezogenen Erklärens auf.

T-Box-bezogene Ansätze:

- Francis Kwong entwickelt an der University of Manchester einen Ansatz, tableaubasierte Subsumtionsbeweise zu erklären. Dieser ist vermutlich am ehesten mit der in dieser Arbeit vorgestellten Methode vergleichbar.

In [Kwo04] beschreibt er sehr knapp drei Aspekte, welche im Rahmen seiner Arbeit untersucht werden. Leider existieren bisher keine weiteren Veröffentlichungen über diese Arbeit.

- An der University of Maryland wird an einer Erklärungskomponente für die Unerfüllbarkeit von Konzepten gearbeitet [PSK05]. Diese baut auf den ebenfalls dort entwickelten Reasoner Pellet [Min] auf. Pellet basiert auf dem Tableau-Algorithmus, das Generieren der Erklärungen erfolgt also in ähnlicher Art und Weise wie in Kapitel 4 beschrieben. Da jedoch nicht Subsumtionsbeziehungen, sondern Unerfüllbarkeit von Konzepten erklärt wird, müssen die einzelnen Erklärungsschritte deutlich anders formuliert werden. Ziel ist es, den gefundenen Clash tatsächlich zu erklären, wohingegen MEX einen Clash im Tableau als offensichtliche Gültigkeit einer Subsumtionsbeziehung versteht. Zusätzlich wird in [PSK05] eine spezielle Form der Visualisierung in SWOOP beschrieben (Black Box Technik), welche ebenfalls zur Verständlichkeit der Erklärungen beitragen soll.

A-Box-bezogene Ansätze:

- Die in [CR] beschriebene Erklärungskomponente *WhyNot* liefert Erklärungen für das Fehlschlagen von A-Box Anfragen. Es werden möglichst plausible Wege gesucht, die A-Box so zu erweitern, dass die gleiche Anfrage ein Ergebnis liefert. Je nach Art und Anzahl der zusätzlich benötigten Fakten wird den Erweiterungen ein bestimmter Plausibilitätswert zugeordnet. So werden dem Benutzer nur relativ plausible Vorschläge unterbreitet. Das endgültige Urteil muss jedoch der Benutzer selbst fällen, da das System nicht wissen kann, welche Erweiterung in Bezug auf die intendierte Modellierung tatsächlich Sinn macht. Der *WhyNot*-Ansatz ist äußerst interessant und könnte eventuell auch für Subsumtionsanfragen realisiert werden. Ein solches System müsste in der Lage sein, Vorschläge für die Erweiterung von Konzept- oder Relationsdefinitionen zu liefern, um gewünschte Subsumtionsbeziehungen zu erhalten.
- Ein speziell auf die Entwicklung des Semantic Web zugeschnittener Ansatz zur Erklärung von Inferenzdiensten wird an der Stanford University entwickelt. Das so genannte *Inference Web*[MdS04] stellt eine Infrastruktur zur Generierung von Erklärungen dar. Es verfolgt das Ziel, verschiedene Ressourcen aus dem Kontext des Semantic Web sowie verteilte Inferenzdienste zu kombinieren, um dadurch Verständnis und Vertrauen im Umgang mit diesen zu fördern. Die tatsächliche Generierung von Erklärungen ist dabei nur einer von fünf Hauptaspekten. Des Weiteren werden die Herkunft der behandelten Daten, Informationen zu den verwendeten Inferenzdiensten, automatisches Schlussfolgern durch die Kombination mehrerer Systeme und die Präsentation der Ergebnisse berücksichtigt.

Das *IW*-Framework[MdS03] stellt verschiedene Mechanismen zur Verfügung, um diese Bereiche zu kombinieren. Beweise werden in einem spe-

zifischen DAML+OIL Format gespeichert. Dieses trennt zwischen konkreten Beweisschritten, der Referenzierung dieser auf ihren Ursprung und die zu deren Ableitung angewendeten Inferenzregeln. Die Meta-Informationen zu den eingesetzten Inferenzsystemen und deren Vorgehensweisen werden in einer so genannten *registry* gehalten. Jeder Eintrag in dieser registry wird ebenfalls in einer DAML+OIL Datei gespeichert. Durch Kombination der Informationen aus diesen beiden Datenpools kann eine Erklärungskomponente letztendlich Erklärungen generieren. Leider befindet sich diese Komponente noch in Entwicklung, weshalb bisher relativ wenig zu deren Vorgehensweise gesagt wurde. Vermutlich lässt sich diese Erklärungskomponente bis zu einem gewissen Grad mit dem in dieser Arbeit vorgestellten System MEX vergleichen. In den bisher veröffentlichten Beispielen zeichnet sich jedoch ab, dass die im Rahmen des *IW* generierten Erklärungen hauptsächlich auf A-Box-Reasoning, das heißt die Ableitung von Beziehungen zwischen Individuen in einer Wissensbasis, zugeschnitten sind.

Zur Präsentation von Beweisen und Erklärungen wurde ein spezieller *IW*-Browser entwickelt. Dieser ist web-basiert, lässt sich also in einem Web-Browser öffnen und bedienen. Zu finden ist er auf der Homepage des *Inference Web* unter <http://iw.stanford.edu/>. Der *IW*-Browser stellt verschiedene Möglichkeiten der Darstellung zur Verfügung. Unter anderem wurde auch eine baumartige Präsentation der Beweisschritte entwickelt. Diese Form der Visualisierung wurde ebenfalls bei der Anbindung von MEX an ONTOTRACK gewählt.

Literaturverzeichnis

- [BCM⁺03] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider. *The Description Logic Handbook*. Cambridge University Press, 2003.
- [BFH⁺99] Alexander Borgida, Enrico Franconi, Ian Horrocks, Deborah L. McGuinness, and Peter F. Patel-Schneider. Explaining ALC Subsumption. In *Description Logics*, 1999.
- [CR] H. Chalupsky and T. Russ. WhyNot: Debugging Failed Queries in Large Knowledge Bases.
- [Gre88] Robert M. Mac Gregor. A Deductive Pattern Matcher. pages 403–408, Saint Paul, Minnesota, 1988.
- [HM01] V. Haarslev and R. Möller. Racer system description. In R. Goré, A. Leitsch, and T. Nipkow, editors, *International Joint Conference on Automated Reasoning, IJCAR'2001, June 18-23, Siena, Italy*, pages 701–705. Springer-Verlag, 2001.
- [Hor98] I. Horrocks. The FaCT System. pages 307–312, 1998.
- [HPSvH03] I. Horrocks, P. Patel-Schneider, and F. van Harmelen. From *SHIQ* and RDF to OWL: The making of a web ontology language, 2003.
- [HST99] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical Reasoning for Expressive Description Logics. In Harald Ganzinger, David McAllester, and Andrei Voronkov, editors, *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705, pages 161–180. Springer-Verlag, 1999.
- [HT00a] I. Horrocks and S. Tobies. Optimisation of Terminological Reasoning. In *Proceedings of the International Workshop in Description Logics 2000 (DL2000)*, 2000.
- [HT00b] I. Horrocks and S. Tobies. Reasoning with Axioms: Theory and Practice. In A. G. Cohn, F. Giunchiglia, and B. Selman, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Seventh International Conference (KR2000)*, San Francisco, CA, 2000. Morgan Kaufmann Publishers.

- [Kwo04] Francis Kwong. Explaining Description Logic Reasoning. In *Description Logics*, 2004.
- [LN03] Thorsten Liebig and Olaf Noppens. OntoTrack: Fast Browsing and Easy Editing of Large Ontologies. In *Proceedings of The Second International Workshop on Evaluation of Ontology-based Tools (EON2003)*, located at ISWC03, Sanibel Island, USA, October 2003.
- [LN04] Thorsten Liebig and Olaf Noppens. ONTOTRACK: Combining Browsing and Editing with Reasoning and Explaining for OWL Lite Ontologies. In *Proceedings of the 3rd International Semantic Web Conference (ISWC2004)*, number 3298 in Lecture Notes in Computer Science, pages 244–257, Hiroshima, Japan, November 2004. Springer Verlag. to appear.
- [Mds03] D. McGuinness and P. Pinheiro da Silva. Infrastructure for Web Explanations, 2003.
- [Mds04] Deborah L. McGuinness and Paulo Pinheiro da Silva. Explaining answers from the Semantic Web: the Inference Web approach. *J. Web Sem.*, 1(4):397–413, 2004.
- [Min] Mindswap Research Group. Pellet OWL Reasoner. <http://www.mindswap.org/2003/pellet/>.
- [PSK05] Bijan Parsia, Evren Sirin, and Aditya Kalyanpur. Debugging OWL Ontologies. In *The 14th International World Wide Web Conference (WWW2005)*, Chiba, Japan, May 2005. To Appear.
- [SSS91] Manfred Schmidt-Schauß and Gert Smolka. Attributive Concept Descriptions with Complements. *Artif. Intell.*, 48(1):1–26, 1991.

Name: Michael Halfmann

Matrikelnummer: 448418

Erklärung

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

Ulm, den 26. Mai 2005

.....