Unterstützung der Modellierung verfahrenstechnischer Prozesse durch Nicht-Standardinferenzen in Beschreibungslogiken

Von der
Fakultät für Mathematik, Informatik und Naturwissenschaften
der Rheinisch-Westfälischen Technischen Hochschule Aachen
zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften
genehmigte Dissertation

vorgelegt von
Diplom-Informatiker Ralf Molitor
aus Rheydt

Berichter: Universitätsprofessor Dr.-Ing. Franz Baader
Universitätsprofessor Dr.-Ing. Wolfgang Marquardt

Tag der mündlichen Prüfung: 8. Dezember 2000

D 82 (Diss. RWTH Aachen)

Abstract

In der Prozesstechnik ist, wie in vielen anderen Bereichen, eine strukturierte Darstellung und Speicherung des anwendungsspezifischen Wissens wünschenswert. Im Rahmen einer Kooperation zwischen dem Lehrstuhl für Prozesstechnik und dem Lehr- und Forschungsgebiet Theoretische Informatik wurde bereits nachgewiesen, dass sich Beschreibungslogiken aufgrund ihrer hohen Ausdrucksstärke und mächtigen Inferenzalgorithmen sehr gut für diese Aufgabe eignen. So ermöglichen die standardmäßig von Beschreibungslogik-Systemen bereitgestellten Inferenzdienste beispielsweise die automatische Berechnung der Spezialisierungshierarchie einer Wissensbasis. Es hat sich jedoch herausgestellt, dass sie für eine umfassende Unterstützung der Erstellung und Wartung der Wissensbasis nicht ausreichen.

In dieser Arbeit werden daher die Nicht-Standardinferenzen *Least Common Subsumer*, *Most Specific Concept* und *Rewriting* untersucht, die im Zusammenspiel die Definition neuer Konzepte und damit die Erweiterung und Pflege der Wissensbasis unterstützen. Die Resultate zur Existenz, Berechenbarkeit und Komplexität sowie die Entwicklung vollständiger Algorithmen zur Lösung dieser Inferenzprobleme konzentrieren sich dabei auf Beschreibungslogiken, die bereits erfolgreich in der Prozesstechnik eingesetzt werden.

Inhaltsverzeichnis

Kapitel 1

Einführung

Die Prozesstechnik beschäftigt sich mit der Entwicklung und Analyse chemischer Prozesse, die eine Menge von Ausgangsstoffen durch eine Sequenz von chemischen, physikalischen und manchmal auch biologischen Einzelschritten in ein Endprodukt umwandeln. Sowohl bei diesen Prozessen als auch bei den Anlagen, in denen sie ablaufen, handelt es sich um sehr komplexe Systeme, die meist von einem Team von Experten entwickelt werden. Um die Komplexität dieser Systeme in den Griff zu bekommen, verwenden die Experten mathematische Modelle. Diese Modelle erlauben die Simulation der Prozesse durch entsprechende Programme und machen sie so einer Analyse z.B. bzgl. Qualität und Sicherheit unter ökonomischen und ökologischen Aspekten zugänglich.

Allerdings ist die Entwicklung der Modelle selbst eine sehr komplexe Aufgabe. Die Kosten zur Lösung dieser Aufgabe sowie die hohen Anforderungen an die Qualifikation der Modellierer schränken ihren Nutzen in der Prozesstechnik häufig stark ein. Ein wesentliches Problem ergibt sich aus der Tatsache, dass ein Experte der Verfahrenstechnik meist kein Modellierungsexperte ist und umgekehrt. Daher bilden geeignete Methoden zur Unterstützung der Prozessmodellierung den Ausgangspunkt für eine nachhaltige Verbesserung und Beschleunigung des gesamten Entwicklungsprozesses.

Eine solche Methodologie wurde am Lehrstuhl für Prozesstechnik der RWTH Aachen entwickelt und in der Modellierungsumgebung ModKit realisiert. Zur Unterstützung der Modellierer bei der Erstellung von Prozessmodellen wird in ModKit ein Katalog sogenannter *kanonischer Modellbausteine* zur Verfügung gestellt, aus dem Bausteine ausgewählt, an den betrachteten Prozess angepasst und zu einem Modell des Prozesses verbunden werden können. Ergeben sich durch die Anpassung neue interessante Bausteine, so werden diese in den Katalog aufgenommen. Mit der Zeit wird also eine umfangreiche Wissenbasis mit diesen und über diese Modellbausteine aufgebaut. Um aber aus dieser Wissensbasis den Nutzen einer Vereinfachung und Beschleunigung des Modellierungsprozesses ziehen zu können, muss sie eine effektive Wiederverwendung von Bausteinen ermöglichen. Diese Wiederverwendung von Bausteinen setzt wiederum voraus, dass die Modellierer Bausteine leicht wiederfinden. Zu diesem

Zweck wurde die Wissensbasis wie folgt strukturiert: Mengen ähnlicher Bausteine wurden zu Klassen zusammengefasst, die Bausteine als Instanzen dieser Klassen gespeichert und die Klassen in einer Spezialisierungshierarchie angeordnet. Bestimmte Klassen bzw. Bausteine lassen sich dann leicht durch das Browsen dieser Hierarchie wiederfinden. Aufgrund der großen Anzahl von Klassen und Bausteinen (die Wissensbasis umfasst inzwischen einige hundert Klassen) ist jedoch die Anordnung der Klassen und die Zuordnung von Bausteinen zu Klassen nicht nur eine sehr zeitaufwendige sondern auch fehleranfällige Aufgabe, die per Hand nicht mehr zu bewältigen ist.

An dieser Stelle kommen nun Beschreibungslogiken ins Spiel. Im Rahmen des durch das Graduiertenkolleg „Informatik & Technik“ geförderten Forschungsprojektes „Terminologische Wissensrepräsentationssysteme in einer verfahrenstechnischen Anwendung“ [Sat98] wurde untersucht, inwiefern sich auf Beschreibungslogiken basierende Wissensrepräsentationssysteme (im folgenden kurz BL-Systeme genannt) zur Unterstützung der Strukturierung der oben erwähnten Wissensbasis eignen.

Beschreibungslogiken (BLen) ermöglichen die strukturierte Darstellung terminologischen Wissens aus einem Anwendungsbereich. Sie stammen ab von Semantischen Netzen und Frames [Qui68, Min75] sowie dem unter der Leitung von Ron Brachman entwickelten System KL-ONE [BS85]. Der Name Beschreibungslogik rührt zum einen daher, dass in diesem Wissensrepräsentationsformalismus das wichtigste Ausdrucksmittel die Konzept*beschreibung* ist, mit deren Hilfe die für den Anwendungsbereich relevanten Konzepte formal definiert werden können. Zum anderen unterscheiden sich Beschreibungslogiken von ihren Vorgängerformalismen dadurch, dass sie über eine formale, *logik*-basierte Semantik verfügen, die z.B. durch eine Übersetzung in Prädikatenlogik erster Stufe angegeben werden kann.

Eine weitere charakteristische Eigenschaft von Beschreibungslogiken sind Inferenzen, die es erlauben, implizites Wissen aus dem in der Wissensbasis explizit gespeicherten Wissen herzuleiten. Eine solche terminologische Wissensbasis besteht aus zwei Komponenten: der *TBox* und der *ABox*. In der TBox werden die relevanten Konzepte des Anwendungsbereichs mit Hilfe sogenannter *Konzeptdefinitionen* der Form $A \doteq C$ gespeichert, wobei A ein *definierter Konzeptname* und C eine Konzeptbeschreibung ist. In der ABox werden konkrete Objekte durch sogenannte *ABox-Individuen* repräsentiert und durch *Assertionen* beschrieben. Standardmäßig werden als Inferenzdienste auf solchen Wissensbasen die Prüfung auf *Konsistenz* sowie die Berechnung von Unterkonzept-/Oberkonzeptbeziehungen (sogenannten *Subsumtionsbeziehungen*) sowie *Instanzbeziehungen* (zwischen Objekten und Konzepten) betrachtet (diese Probleme werden formal in Kapitel 3 eingeführt). Die Verwendung einer BL in einem BL-System setzt voraus, dass diese Inferenzprobleme für die Logik entscheidbar sind und die zugehörigen Dienste auch (möglichst effizient) implementiert werden können. Um dies zu ermöglichen, muss die Ausdrucksstärke der verwendeten Beschreibungslogik geeignet eingeschränkt sein. Andererseits erlaubt aber eine zu ausdrucksschwache Logik nicht mehr die Darstellung der anwendungsrelevanten Konzepte. Es ist daher

wichtig, einen guten Kompromiss zwischen der Ausdrucksstärke der Beschreibungslogik einerseits und der Komplexität der Inferenzprobleme andererseits zu finden. Die Suche nach diesem (anwendungsabhängigen) Kompromiss war eine der wesentlichen treibenden Kräfte der Forschung im Bereich BLen (s. z.B. [BS00] oder Kapitel 3 dieser Arbeit für einen Überblick über die dabei erzielten Ergebnisse).
In [Sat98] wurde nachgewiesen, dass es hinreichend ausdrucksstarke BLen mit entscheidbaren Standardinferenzproblemen gibt, die zur Strukturierung der Wissensbasis in der Prozesstechnik eingesetzt werden können. Die Vorgehensweise ist dabei die folgende: Die Klassenbeschreibungen in ModKit werden in eine TBox und die Bausteine und ihre Beschreibungen in eine ABox übersetzt. Das BL-System prüft dann die Wissensbasis automatisch auf Konsistenz und berechnet die gültigen Subsumtions- und Instanzbeziehungen. Schließlich werden die Klassen und Bausteine in ModKit gemäß der erhaltenen Subsumtionshierarchie und Instanzbeziehungen angeordnet (ausführlich wird dieses Szenario in Kapitel 4 beschrieben).
Die automatische Strukturierung der Wissensbasis reicht aber bei der ständig wachsenden Anzahl von Klassen und Bausteinen als Unterstützung noch nicht aus. Erfahrungen der Prozesstechniker zeigen, dass von Zeit zu Zeit eine Restrukturierung der Wissensbasis notwendig ist, um (1) eine ausgewogene Struktur der Spezialisierungshierarchie zu behalten und (2) zu große Klassen auf den unteren Ebenen der Hierarchie zu vermeiden, da beide Situationen einen negativen Einfluss auf das „leichte Wiederfinden“ von Bausteinen haben. Restrukturierung bedeutet in diesem Zusammenhang, dass neue Klassen definiert werden: in (1) als Oberklasse zu einer Menge von Klassen, um die Hierarchie an einer Stelle gezielt tiefer und damit schlanker machen zu können; und in (2) als speziellste Klasse zu einer Menge von Bausteinen, um so große Mengen von Instanzen auf mehrere Klassen verteilen zu können.
Nun ist die Definition einer neuen Klasse aber ein sehr aufwendiger und fehleranfälliger Schritt: häufig ist nicht klar, was die relevanten Eigenschaften der neuen Klasse sind und wie sie im vorgegebenen Formalismus repräsentiert werden können. Wünschenswert ist daher ein Systemdienst, der bei Eingabe weniger und für den Modellierer leicht bereitzustellender Informationen eine Beschreibung der neuen Klasse vorschlägt. In den oben beschriebenen Situationen könnte es sich bei diesen Informationen beispielsweise um die Menge der direkten Subklassen der neuen Klasse handeln oder um einige bereits vorliegende Bausteine, die Instanzen der neuen Klasse sein sollen. Da durch die Standardinferenzdienste nur strukturelles Wissen berechnet wird, d.h. Subsumtions- und Instanzbeziehungen zwischen bereits definierten Konzepten und Individuen, reichen diese offensichtlich nicht aus, um hier eine ausreichende Unterstützung zu bieten.
Das Ziel des ebenfalls durch das Graduiertenkolleg „Informatik & Technik“ geförderten und dieser Arbeit zugrunde liegenden Folgeprojektes besteht darin, neue Dienste zur Unterstützung der Definition neuer Klassen zu entwickeln bzw. bereitzustellen. Der dazu verfolgte Ansatz basiert auf den Nicht-Standardinferenzen[1] *Least Com-*

[1]Der Begriff Nicht-Standardinferenzen wurde gewählt, da entsprechende Systemdienste nicht

mon Subsumer, *Most Specific Concept* und *Rewriting*. Der Least Common Subsumer (LCS) einer Menge $\{C_1, \ldots, C_n\}$ von Konzeptbeschreibungen ist die (bzgl. Subsumtion) speziellste Konzeptbeschreibung, die jedes der Konzepte C_i subsumiert. Das Most Specific Concept (MSC) zu einem Individuum a ist die speziellste Konzeptbeschreibung, von der a Instanz ist. Diese beiden Nicht-Standardinferenzen wurden für Teilsprachen der BL eingeführt und untersucht, die dem bei AT&T entwickelten BL-System Classic zugrunde liegt [BMPS+91, BP94]. Dort wurden sie für das Induktive Lernen eingesetzt, wobei zum einen das Problem der Berechnung des MSC in den betrachteten BLen nicht gelöst werden konnte und zum anderen nur eine ad hoc-Implementierung des LCS in Classic eingebunden wurde [CH94b, FP96]. Inzwischen liegt aber eine vollständige und korrekte Charakterisierung von MSC und LCS in den betrachteten Teilsprachen von Classic vor [BK98, Küs00].

In unserer Prozesstechnikanwendung werden MSC und LCS nun wie folgt eingesetzt: Zur Unterstützung der Definition einer neuen Oberklasse zu einer Menge von Klassen $\{K_1, \ldots, K_n\}$ werden die Klassenbeschreibungen in Konzeptbeschreibungen $\{C_1, \ldots, C_n\}$ übersetzt. Zu dieser Menge wird dann der LCS berechnet. Schließlich wird die Klassenbeschreibung, die sich aus dem Rückübersetzen des LCS ergibt, dem Modellierer als Beschreibung der neuen Klasse vorgeschlagen. Zur Unterstützung der Definition einer speziellsten Klasse zu einer Menge von Bausteinen werden diese zunächst als Individuen $a_1, \ldots, a_m$ in einer ABox repräsentiert und ihre Beschreibungen in entsprechende Assertionen übersetzt. Im nächsten Schritt werden die Beschreibungen der konkreten Objekte durch Berechnung des MSC zu Konzeptbeschreibungen $C_1, \ldots, C_m$ generalisiert. Auf diese kann dann wiederum die LCS-Operation angewendet werden und liefert wie im ersten Fall einen Vorschlag für die neue Klassenbeschreibung.

Die vorgeschlagenen Klassenbeschreibungen können aber in der Regel nicht unbesehen in die Wissensbasis übernommen werden. Vielmehr müssen sie vom Modellierer vorher begutachtet und ggf. noch geeignet modifiziert werden. Es ist daher wichtig, dass die ausgegebenen Beschreibungen möglichst klein und damit gut lesbar sind. Die im Rahmen des Projekts entwickelten Algorithmen zur Lösung der neuen Inferenzprobleme erfüllen diese Anforderung jedoch leider nicht. Gründe hierfür sind zum einen die inhärente Komplexität der Inferenzprobleme und zum anderen die Tatsache, dass die Algorithmen auf sogenannten aufgefalteten Konzeptbeschreibungen arbeiten (d.h. Beschreibungen, bei denen alle in der TBox eingeführten definierten Konzeptnamen durch ihre definierenden Konzepte ersetzt sind) und daher auch als Ausgabe aufgefaltete (und damit sehr große) Beschreibungen liefern. Diesem Problem wird durch einen der LCS-Operation nachgeschalteten Rewriting-Schritt begegnet. Rewriting von Konzeptbeschreibungen bezüglich einer gegebenen Terminologie zielt darauf ab, diese Beschreibungen in äquivalente kleinere Beschreibungen zu transformieren, welche definierte Konzeptnamen enthalten. Das Problem der Berechnung einer sol-

standardmäßig in BL-Systemen zur Verfügung gestellt werden.

chen äquivalenten Beschreibung minimaler Größe wird als das *minimale Rewriting-Problem* bezeichnet.

Das Hauptaugenmerk dieser Arbeit liegt auf der umfassenden Untersuchung von Existenz, Berechenbarkeit und Komplexität für LCS, MSC und Rewriting in einer BL, die ausdrucksstark genug für den Einsatz in der Prozesstechnik ist. Für eine adäquate Übersetzung der Klassenbeschreibungen haben sich sogenannte Existenzrestriktionen als unbedingt notwendig erwiesen. Diese können in der BL von Classic nicht ausgedrückt werden, sodass für die Untersuchung der Nicht-Standardinferenzen nicht auf die in Classic verfolgten Ansätze zurückgegriffen werden konnte. LCS, MSC und minimales Rewriting-Problem werden hier erstmalig für BLen mit Existenzrestriktionen formal und umfassend untersucht. Im Falle des Rewritings zählt dazu auch die Definition eines allgemeinen Rahmens für das Rewriting in BLen, von dem das minimale Rewriting-Problem als eine Instanz zu sehen ist. Außerdem wird über die Berechnung des LCS zu einer Menge $\{C_1, \ldots, C_n\}$ von Konzeptbeschreibungen hinaus auch das Problem der effizienten Berechnung der Subsumtionshierarchie aller LCS zu allen Teilmengen von $\{C_1, \ldots, C_n\}$ betrachtet. Hierzu werden Methoden aus der *Formalen Begriffsanalyse* [GW99] herangezogen.

Struktur dieser Arbeit

Kapitel 2 gibt eine kurze Einführung in den Bereich der Prozesstechnik, insbesondere in die rechnergestützte Modellierung verfahrenstechnischer Prozesse, wie sie am Lehrstuhl für Prozesstechnik der RWTH Aachen betrieben wird. Den Schwerpunkt bildet dabei die Beschreibung der Wissensbasis, die dieser Modellierungsumgebung zugrunde liegt und die durch BL-Systeme strukturiert und gepflegt werden soll.

Kapitel 3 stellt die für diese Arbeit relevanten Grundlagen zu BLen und ihren Standard- und Nicht-Standardinferenzen bereit. Es werden Syntax und Semantik von BLen und alle betrachteten Inferenzprobleme formal definiert. Dabei wird die BL $\mathcal{ALE}$ ($\mathcal{E}$ für Existenzrestriktion) als die Zielsprache für die nachfolgenden Untersuchungen der Nicht-Standardinferenzen eingeführt. Das Kapitel schließt mit einem vollständigen Überblick über die in dieser Arbeit erzielten Resultate.

Kapitel 4 beschreibt die zentralen Szenarien für den Einsatz von BL-Systemen in der Prozesstechnik: die Unterstützung der Strukturierung der Wissensbasis durch Standardinferenzen und die Unterstützung der Erweiterung der Wissensbasis durch Nicht-Standardinferenzen.

Kapitel 5 ist das erste der vier technischen Kapitel der Arbeit. Aufbauend auf einer geeigneten Charakterisierung der Subsumtion liefert es eine Charakterisierung des LCS in $\mathcal{ALE}$, aus der sich leicht ein Algorithmus zur Berechnung des LCS ergibt, der für die Anwendung auch implementiert wurde.

Kapitel 6 zeigt, dass sich Methoden der Formalen Begriffsanalyse für die Berechnung der Subsumtionshierarchie aller LCS zu Teilmengen einer Menge von $\mathcal{ALE}$-

Konzeptbeschreibungen eignen. Die Einführung in die Formale Begriffsanalyse beschränkt sich dabei auf die für dieses Ziel notwendigen Begriffe und Zusammenhänge.

Kapitel 7 untersucht MSCs in $\mathcal{ALE}$ und zeigt, dass das MSC zu einem Individuum in $\mathcal{ALE}$ im allgemeinen nicht existiert. Daher konzentrieren sich die nachfolgenden Untersuchungen auf die Definition und Charakterisierung einer möglichst ausdrucksstarken Approximation des MSC, wobei sich die erzielten Resultate aber auf zwei Teilsprachen von $\mathcal{ALE}$ beschränken.

Kapitel 8 betrachtet zuerst das durch das minimale Rewriting-Problem induzierte Entscheidbarkeitsproblem, um einen Eindruck von der Komplexität des Problems zu vermitteln. Anschließend wird ein nicht-deterministischer Algorithmus zur Berechnung minimaler Rewritings in $\mathcal{ALE}$ vorgestellt, der zudem die Basis für den im letzten Abschnitt des Kapitels beschriebenen deterministischen heuristischen Algorithmus bildet, der (nicht notwendig minimale) Rewritings in $\mathcal{ALE}$ berechnet und für die Anwendung implementiert wurde.

Kapitel 9 beschreibt kurz die prototypische Implementierung der in den vorherigen Kapiteln entwickelten Algorithmen und gibt dann die Erfahrungen wieder, die bei Entwicklung, Anbindung und Einsatz des Prototypen in der Prozesstechnik gemacht wurden.

Kapitel 10 schließt mit einer inhaltlichen Zusammenfassung und einem Ausblick auf weitere Ansätze und Ideen zum Einsatz von BLen in der Prozesstechnik, die bei der Fortführung unserer Kooperation im Rahmen eines DFG-Forschungsprojektes verfolgt werden sollen.

Kapitel 2

Prozesstechnik

Dieses Kapitel gibt eine Einführung in den Bereich der Prozesstechnik, insbesondere in die rechnergestützte Modellierung verfahrenstechnischer Prozesse, wie sie am Lehrstuhl für Prozesstechnik der RWTH Aachen betrieben wird. Nach einer kurzen Motivation und Darstellung der Ziele der Prozesstechnik wird in Abschnitt 2.2 die Bedeutung mathematischer Modelle für eine erfolgreiche Entwicklung und Verbesserung verfahrenstechnischer Prozesse herausgestellt. Die nachfolgenden Abschnitte beschäftigen sich dann mit dem Vorgang der Modellentwicklung, seiner Formalisierung und seiner Unterstützung durch eine geeignete Entwicklungsumgebung. Im letzten Abschnitt wird schließlich die Brücke geschlagen zur logikbasierten Wissensrepräsentation: Durch die Notwendigkeit, die Wissensbasis, die einer solchen Entwicklungsumgebung zugrunde liegt, aufzubauen, zu strukturieren und zu pflegen erschließt sich die Prozesstechnik als Anwendungsbereich für die in Kapitel 3 ausführlich vorgestellten Beschreibungslogik-Systeme.

2.1 Motivation und Ziele

Die Prozesstechnik beschäftigt sich mit der Modellierung verfahrenstechnischer Prozesse, die in meist sehr großen und komplexen chemischen Anlagen ablaufen. Unter einem *Prozess* versteht man dabei eine Folge von chemischen, physikalischen und manchmal auch biologischen Einzelschritten, durch die eine Menge von Ausgangsstoffen in ein wertvolleres Endprodukt (oder mehrere) umgewandelt werden soll. Sich verändernde Umwelt- und Sicherheitsbedingungen, gestiegene Ansprüche an die Qualität der Produkte und zunehmender Konkurrenzdruck machen es nun notwendig, solche Prozesse neu zu entwickeln bzw. ständig zu verbessern. Da hierbei der Zeit- und Kostenaufwand möglichst gering zu halten ist, werden seit einigen Jahren mehr und mehr modellbasierte Verfahren statt experimenteller Untersuchungen während der Prozessentwicklung eingesetzt [Mar96b].

Unter einem *Modell* versteht man eine zielgerichtete Vereinfachung der Realität durch Anwenden des Abstraktionsprinzips. In diesem Sinne bildet ein *Prozessmodell* eine vereinfachte, abstrakte Beschreibung eines chemischen Prozesses. Ziel ist es, den Prozess mit Hilfe des Modells der Analyse und Simulation zugänglich zu machen, um ein besseres Verständnis für den Prozess zu bekommen und schließlich Entwicklung und Betrieb effizienter zu gestalten.

Während des Lebenszyklus eines verfahrenstechnischen Prozesses, d.h. während Entwicklung, Realisierung und Betrieb, gibt es aber nicht nur ein Modell. Tatsächlich benötigt man für einen Prozess ganze Sammlungen von Modellen bzw. Repräsentationen. Diese reichen von natürlich-sprachlichen Beschreibungen über graphische Repräsentationen bis hin zu Systemen von algebraischen, Differential- und Integral-Gleichungen [Mar96b, MvWB99]. Sehr stark vereinfachende, grobe Modelle eignen sich beispielsweise sehr gut, um in frühen Entwicklungsphasen zwischen verschiedenen Alternativen der Prozessgestaltung auszuwählen, wohingegen immer detailliertere Modelle betrachtet werden, je näher man der anvisierten Beschreibung des Prozesses durch ein Fließbild kommt.

Als Beispiel für unterschiedliche Modellsichten betrachte man den in Abbildung 2.1 als Fließbild dargestellten Ausschnitt aus dem Ethylen-Glykol-Prozess. Kernstücke dieses Prozesses bilden ein Reaktor, der über eine Rühreinheit und einen Kühlmantel verfügt, und ein Verdampfer. Diese beiden Teile sind mit einer Leitung verbunden, die zusätzlich ein Ventil enthält. In Abbildung 2.2 ist der Reaktor graphisch mit Hilfe des Verfahrenstechnischen Datenmodells VeDa dargestellt. (VeDa wird zusammen mit der Modellierungsumgebung ModKit im folgenden noch vorgestellt werden.) Aufgabe dieses Modells ist die Beschreibung der strukturellen Dekomposition des Reaktors, wobei von Leitung und Verdampfer abstrahiert wird und die Verbindung von Kühlmantel und Umgebung besonders betont wird. Schließlich zeigt Abbildung 2.3 Bilanzgleichungen, durch die die Reaktionsphase modelliert werden kann. Zur Darstellung wird dabei ein sogenannter Und-Oder-Graph verwendet: durch Und-Knoten werden die Terme, aus denen sich eine Gleichung zusammensetzt, erfasst und die Oder-Knoten repräsentieren jeweils mögliche Alternativen zur Beschreibung bzw. Berechnung dieser Terme. Intuitiv legt ein Pfad durch diesen Graphen ein mathematisches Modell für den Reaktionsprozess fest.

2.2 Mathematische Modelle

Typischerweise ist das angestrebte „Zielmodell“ ein möglichst exaktes mathematisches Modell des chemischen Prozesses, da erst ein solches Modell die genaue Analyse und Simulation des Verhaltens des Prozesses ermöglicht. Mathematische Modelle, d.h. Mengen von algebraischen, Differential- und Integralgleichungen, bestehen, je nach Detaillierungsgrad, häufig aus mehreren hundert meist sehr komplexen Gleichungen. So ein Gleichungssystem exakt zu lösen ist praktisch nicht möglich – we-

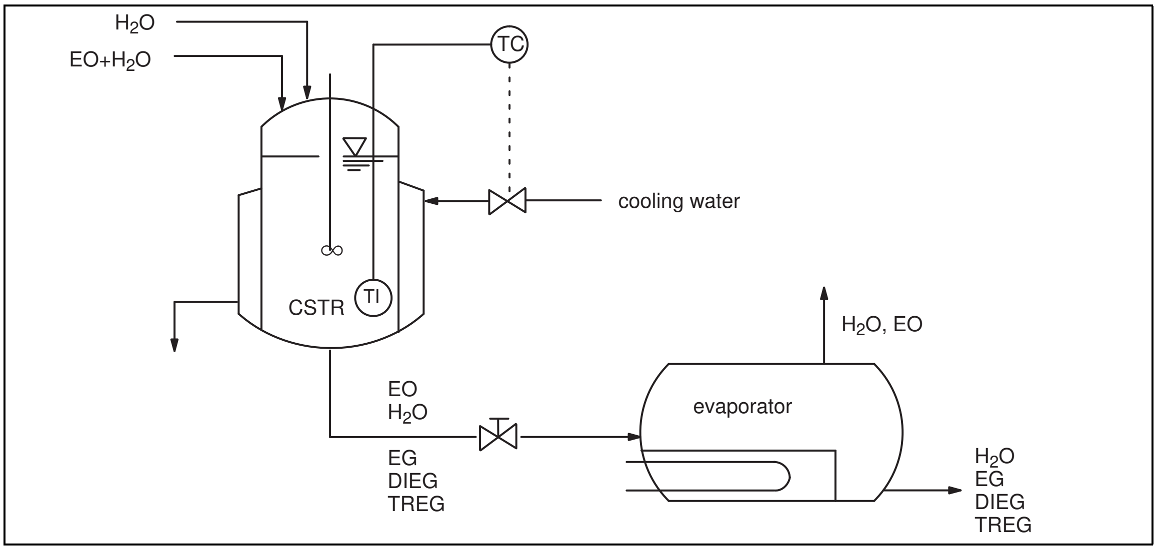


Abbildung 2.1: Das Fließbild zum Ethylen-Glykol-Prozess.

der „von Hand“ noch mit automatischen symbolischen Lösungsverfahren. Tatsächlich werden die Gleichungen in der Praxis in einem ersten Schritt vereinfacht und dann durch numerische Lösungsverfahren iterativ gelöst. Die Nutzung derartiger Verfahren erfordert aber zunächst noch die Übertragung der mathematischen Beschreibungen in das jeweilige spezifische Eingabeformat, bevor der Modellierer durch Simulationen Prozessparameter hinsichtlich ihres Einflusses auf den Gesamtprozess untersuchen und optimieren kann.

Nun ist die Konstruktion eines mathematischen Modells selbst ein sehr komplexer Prozess. Dies liegt natürlich im wesentlichen an den zu modellierenden chemischen Prozessen, die aufgrund der großen Anzahl und der Verschiedenartigkeit der möglichen auftretenden Phänomene schon sehr komplex sind. Je genauer man das Verhalten eines solchen Prozesses in einem mathematischen Modell beschreiben möchte, desto mehr Gleichungen wird es enthalten. Je mehr Gleichungen aber ein Modell enthält, desto mehr Folgerungen und implizite Zusammenhänge herrschen zwischen den (Elementen der) Gleichungen, die vom Modellierer nicht erkannt werden (können) und die dann zu unvollständigen Schlussfolgerungen oder Modellierungsfehlern führen können.

Um diese Komplexität in den Griff zu bekommen, wird ein mathematisches Modell schrittweise entwickelt, wobei verschiedene graphische Repräsentationen die Zwischenergebnisse bilden. Bei der computer-unterstützten Prozessmodellierung steht man damit aber dem Problem gegenüber, die Übersetzung verschiedener graphischer Repräsentationen ineinander sowie die Übersetzung eines graphischen Modells in ein mathematisches und schließlich die Transformation des mathematischen Modells in ein für numerische Lösungsverfahren oder Simulationswerkzeuge lesbares Format unterstützen zu müssen. Derzeit gibt es aber kein Softwarewerkzeug, dass all diese Trans-

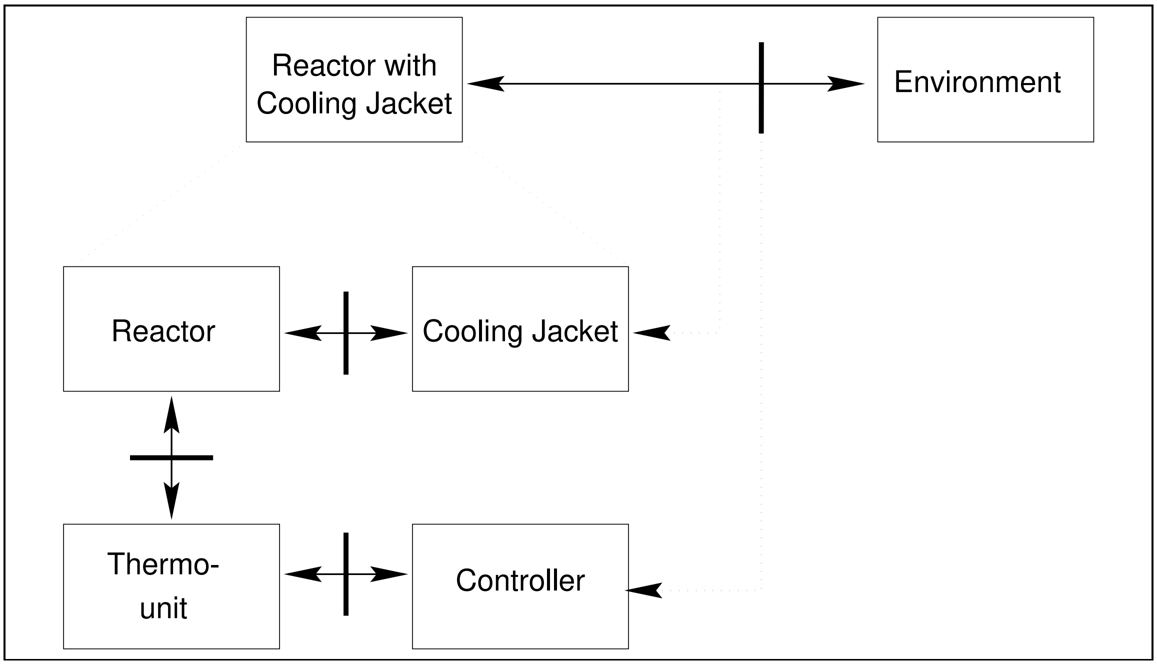


Abbildung 2.2: Die Repräsentation eines Ausschnitts des Ethylen-Glykol-Prozesses in VeDa.

formationen unterstützt [Mar96b]. Darüberhinaus herrscht auch keine Einigkeit über die bei der Modellentwicklung zu verwendende Methodik oder die elementaren Bausteine und Aspekte, die in einem Prozessmodell zu berücksichtigen sind [Mar96b]. Daher ist ein wesentliches Ziel der Forschungsarbeit am Lehrstuhl für Prozesstechnik der RWTH Aachen die Entwicklung sowohl einer Methodik als auch eines geeigneten verfahrenstechnischen Datenmodells zur Prozessmodellierung. Die insbesondere in den vergangenen fünf Jahren erzielten Ergebnisse werden nun im folgenden kurz vorgestellt. Den Ausgangspunkt bildet ein allgemeiner Rahmen für die Beschreibung des Vorgangs der Modellentwicklung.

2.3 Der Vorgang der Modellentwicklung

In [JM95] wird ein aus dem Bereich des Software-Engineering bekanntes Rahmenwerk [Poh96] auf die verfahrenstechnische Prozessmodellierung angepasst. In diesem Rahmenwerk wird die Entwicklung eines Prozessmodells in einem dreidimensionalen Raum beschrieben:

Repräsentationsdimension: Sie umfasst verschiedene Repräsentationsformalismen und spiegelt damit einerseits den Detaillierungsgrad wider, den das Modell aufweist, andererseits aber auch Tiefe und Umfang des Verständnisses, das die Modellierer für den Prozess entwickelt haben. Wie bereits erwähnt reichen die

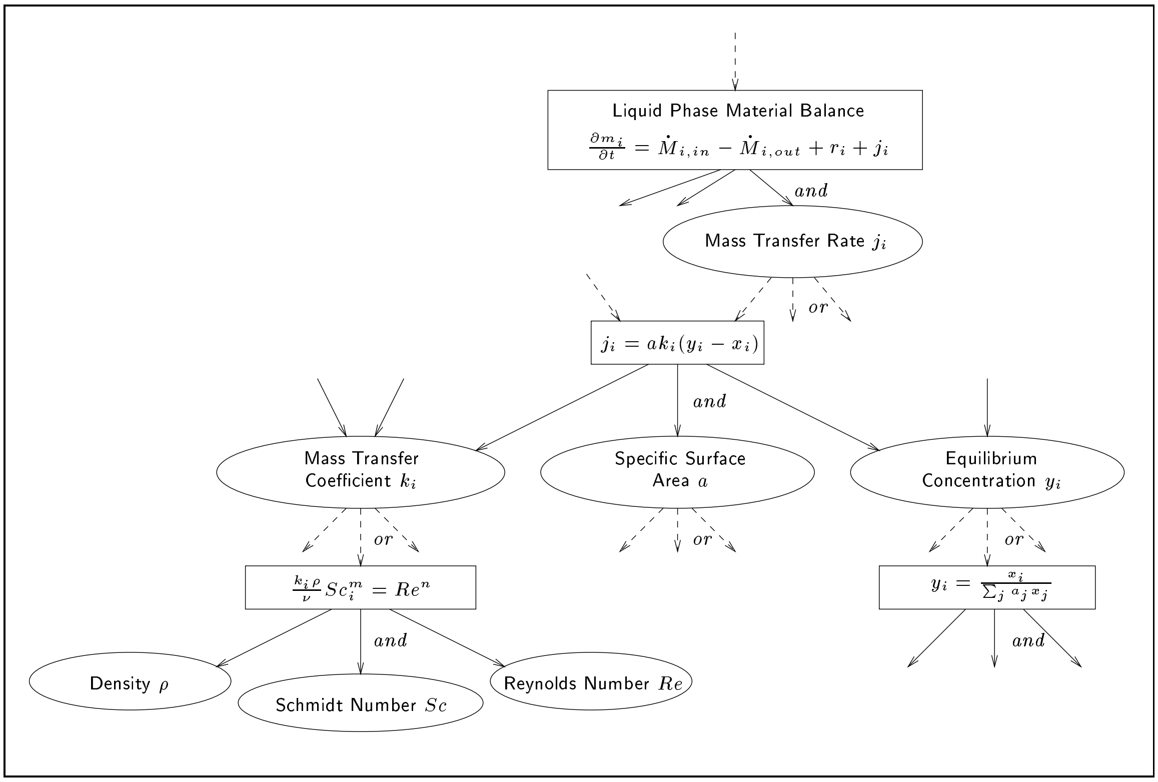


Abbildung 2.3: Gleichungen zur Reaktionsphase im Ethylen-Glykol-Prozess.

Formalismen dabei von einfachem Text über Fließbilder bis hin zu mathematischen Gleichungen (siehe obiges Beispiel).

Konsensdimension: Diese Dimension beschreibt den Grad der Übereinstimmung, der zwischen den beteiligten Ingenieuren bzgl. des zu erstellenden Modells erreicht worden ist. Er sollte während des Entwicklungsprozesses natürlich stetig zunehmen. Im Beispiel würde die Konsensdimension beispielsweise die Entscheidung erfassen, bei der strukturellen Modellierung des Reaktors von den angrenzenden Komponenten zu abstrahieren.

Spezifikationsdimension: Sie erfasst auf verschiedenen Detaillierungsstufen die jeweils verwendeten verfahrenstechnischen Modellierungskonzepte, wie z.B. Komponenten und Verknüpfungen oder die in einem Modell berücksichtigten physikalischen und chemischen Phänomene.

Während der Modellentwicklung bewegen sich die Modellierer nun entlang einer Trajektorie durch den Raum, der durch diese drei Dimensionen aufgespannt wird. Ausgangspunkt ist eine grobe Modellspezifikation in informeller textueller Form. Den Endzustand, welcher meist nicht geradlinig erreicht wird, bildet eine detaillierte Modellspezifikation, die von allen beteiligten Modellierern akzeptiert wird und insbesondere eine mathematische Modellrepräsentation umfasst. Über welche Zwischenschritte man zu einem solchen mathematischen Modell kommt und wie diese Schritte in einer Entwicklungsumgebung unterstützt werden können, wird nun in den folgenden Abschnitten näher beschrieben.

2.4 Die Modellentwicklung mit Bausteinen

Ein traditioneller, an Grundoperationen orientierter Ansatz zur Erstellung eines mathematischen Modells würde das Fließbild des Prozesses in die einzelnen Grundoperationen aufteilen und dann mit der Verhaltensbeschreibung der einzelnen Apparate durch Gleichungen beginnen. Das Resultat wäre eine grobgranulare Modellstruktur, die sehr spezifisch für den betrachteten Prozess wäre. Der Aufwand für die Wartung und Wiederverwendung wäre aber unnötig hoch und würde die Produktivität der Modellentwicklung in späteren Projekten vermindern.

Um diesen Nachteil zu umgehen und den Modellierungs- und Wiederverwendungsaufwand zu verringern, wurden von Marquardt in [Mar96b, Mar96a] verschiedene *kanonische Modellbausteine* vorgeschlagen, die Prozesse auf verschiedenen Abstraktionsebenen beschreiben. Beispielsweise werden für die Strukturbeschreibung zwei Klassen von Modellbausteinen eingeführt: *Komponenten* und *Verknüpfungen*. Komponenten stehen für abgrenzbare materielle Teile eines Prozesses wie z.B. eine Reaktionsphase, wohingegen Verknüpfungen Ströme zwischen Komponenten repräsentieren, also z.B. das Rohr vom Reaktor zum Verdampfer. Sowohl Komponenten als auch Verknüpfungen können elementar oder zusammengesetzt sein. Abbildung 2.4 zeigt eine mögliche Dekomposition des Reaktors aus dem Fließbild zum Ethylen-Glykol-Prozess aus Abbildung 2.1.

Unter Verwendung solcher Modellbausteine lässt sich nun die Struktur eines Prozessmodells beschreiben durch die Menge der Bausteine, aus denen es besteht, sowie einer Beschreibung der Beziehungen, die zwischen diesen Bausteinen herrschen. Im folgenden wird ein aus sieben Schritten bestehendes, mögliches Szenario für die Entwicklung eines Modells aus Modellbausteinen skizziert. Dabei wird davon ausgegangen, dass dem Modellierer ein Katalog von sogenannten *Standard-Modellbausteinen*, d.h. Bausteinen, die häufig benutzt werden und gemeinhin akzeptiert sind, zur Verfügung steht, der insbesondere mit einer geeigneten Suchfunktion für die Bausteine ausgestattet ist.

1. Der Modellierer wählt aus dem Katalog die für den zu modellierenden Prozess benötigten Standard-Modellbausteine.

2. Die gewählten Bausteine werden, falls nötig, an die Anforderungen des betrachteten Prozesses angepasst. Solche Modifikationen reichen dabei von Spezialisierungen existierender Eigenschaften der Bausteine bis zum Löschen oder Hinzufügen von Eigenschaften.

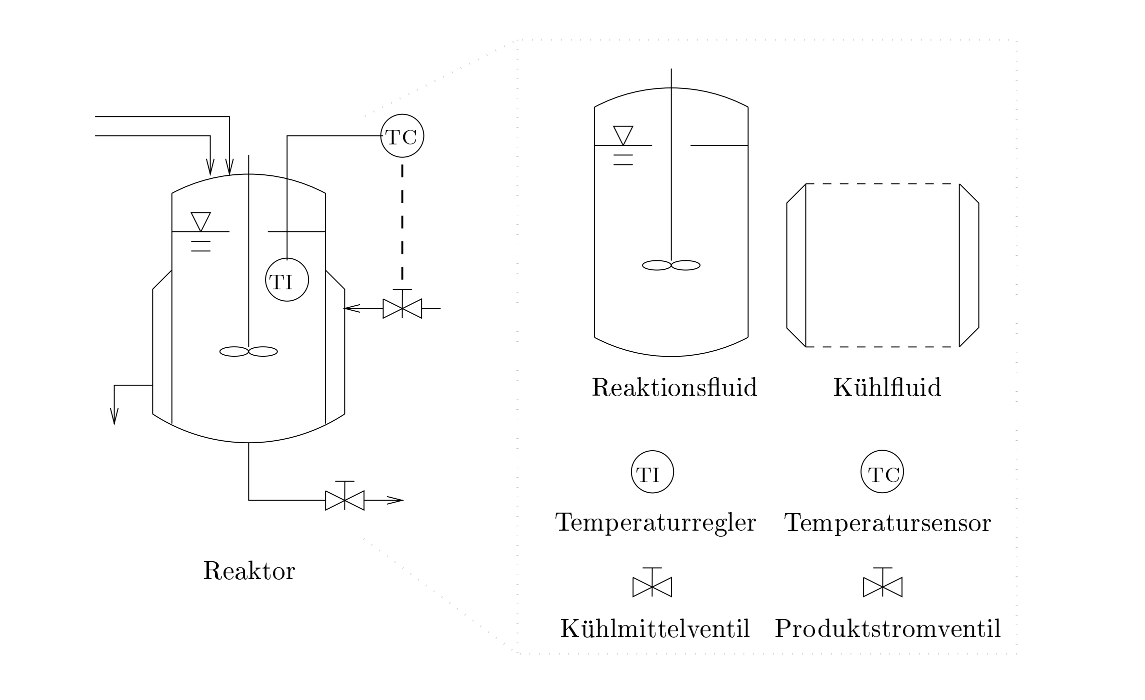


Abbildung 2.4: Eine mögliche strukturelle Dekomposition des Reaktors im Ethylen-Glykol-Prozess in Modellbausteine.

3. Die Bausteine werden miteinander verbunden, typischerweise unter Verwendung ihrer bereits spezifizierten Schnittstellen.

4. Zusammengesetzte Bausteine werden in ihre Komponenten zerlegt, bis der gewünschte Detaillierungsgrad erreicht ist.

5. Das Verhalten wird festgelegt, indem zunächst die zu berücksichtigenden Phänomene bestimmt werden, für jedes dieser Phänomene eine oder mehrere Prozessvariablen eingeführt werden, Gleichungen angegeben werden, die diese Variablen zueinander in Beziehung setzen und schließlich (falls nötig) die Variablen initialisiert werden.

6. Aus den Ergebnissen der vorherigen Schritte wird automatisch die Eingabe bzw. der Eingabecode für ein numerisches Lösungsverfahren oder ein Simulationswerkzeug generiert und das Verhalten simuliert.

7. Sollten bei der Spezifizierung der Bausteine neue Modellbausteine entstanden sein, die wahrscheinlich in anderen Modellen wiederverwendet werden können, so werden diese als neue Standard-Modellbausteine in den Katalog integriert.

Zur Unterstützung des Modellierers sollte nun jeder dieser Schritte in einer entsprechenden Entwicklungsumgebung so kontrolliert ablaufen, dass die Konsistenz der erzeugten Modelle stets gewährleistet ist. Beispielsweise sollte das unterstützende System Modifikationen an Eigenschaften von Bausteinen gegen die tatsächlich hinterlegten Beschreibungen prüfen. Wählt der Modellierer zum Beispiel aus dem Katalog einen Baustein, der einen *Reaktor mit Rühreinheit* repräsentiert und entfernt dann in Schritt 2 die Rühreinheit, so sollte das System erkennen, dass der Baustein dann nurmehr einen *Reaktor* repräsentiert.

Leider unterstützt kein heute auf dem Markt vertretenes Modellierungswerkzeug das oben skizzierte Szenario. Dabei lassen sich die „auf dem Markt vertretenen" Werkzeuge grob in zwei Gruppen einteilen, die *gleichungs-orientierten* und die *block-orientierten* Werkzeuge [Mar96b]:

- Die Idee hinter gleichungs-orientierten Werkzeugen wie z.B. SpeedUp oder gPROMS besteht darin, mathematische Modelle entweder in einer deklarativen Sprache unter Verwendung einer Modellbibliothek oder in einer prozeduralen Sprache mit vordefinierten Hilfsroutinen zu implementieren. Auf dieser Idee basierende Werkzeuge sind natürlich einerseits sehr flexibel, da sie die Erstellung verschiedenartigster Modelle unterstützen. Andererseits ist diese Art der Modellierung sehr aufwendig und stellt hohe Ansprüche an die Qualifikation der Modellierer.

- Block-orientierte Werkzeuge wie z.B. ASPEN PLUS arbeiten auf Fließbild-Ebene: der Modellierer hat die Möglichkeit, Bausteine – in diesem Kontext auch *Prozesseinheiten* genannt – in einer geeigneten Modellierungssprache oder einem graphischen Editor auszuwählen, zu spezifizieren und zu verknüpfen. Dabei schränkt das System aber die Möglichkeiten, Bausteine zu spezifizieren oder zu modifizieren, stark ein, sodass der in den erzeugten Modellen erreichbare Detaillierungsgrad und damit auch die Einsetzbarkeit des Systems begrenzt ist. Demgegenüber sind derartige Systeme aber sehr komfortabel und weniger anspruchsvoll bzgl. der Qualifikation der Modellierer.

Man beachte, dass auch eine Umgebung, in der Werkzeuge beider Typen integriert werden, die Schritte 2, 4, 6 und 7 des Szenarios noch nicht unmittelbar unterstützen würden: Einerseits erlauben die block-orientierten Werkzeuge beispielsweise keine beliebige Dekomposition von Bausteinen. Andererseits erlauben die gleichungsorientierten Werkzeuge eine Modellierung nur mit Hilfe abstrakter mathematischer Konzepte. Wünschenswert ist aber ein Werkzeug, das eine Beschreibung von Modellen mit Hilfe verfahrenstechnischer und physikalischer Konzepte erlaubt, die dabei so präzise ist, dass eine automatische Generierung des zugehörigen mathematischen Modells möglich ist. Erste Versuche einer solchen automatischen Codegenerierung aus Modellbeschreibungen werden im Rahmen der am Lehrstuhl für Prozesstechnik der RWTH Aachen entwickelten und weiter unten sowie in Kapitel 9 noch näher beschriebenen Modellierungsumgebung ModKit unternommen.

Ein weiteres Problem bei der Modellentwicklung mit Bausteinen ergibt sich aus der Tatsache, dass Anzahl und Art der bereitgestellten Bausteine stark von den zur Verfügung stehenden Werkzeugen und Repräsentationsformalismen abhängt. Es ist also heute in der Praxis nicht möglich, während seines Lebenszyklus nur ein Modell eines verfahrenstechnischen Prozesses zu entwickeln und dessen Detaillierungsgrad, die möglichen Sichten auf den Prozess sowie die (graphische) Repräsentation an den aktuellen Zustand des Prozesses bzw. des Modells anzupassen. Dieses Problem erhöht nicht nur die Kosten für die Modellierung, sondern erschwert auch die Wiederverwendung von Modellen und bildet noch eine zusätzliche Fehlerquelle. Um nun dieses Problem in den Griff zu bekommen und darüber hinaus eine formale Grundlage für die Realisierung des oben beschriebenen Szenarios zu schaffen, wurde das *Verfahrenstechnische Datenmodell* VeDa entwickelt.

2.5 Das Verfahrenstechnische Datenmodell VeDa

Das Datenmodell VeDa wurde Anfang der neunziger Jahre erstmalig vorgestellt [Mar92, MGG93] und am Lehrstuhl für Prozesstechnik der RWTH Aachen weiterentwickelt [BM98a, SM98a, SM98b, BM98b, vWM98, KLM98]. VeDa ist ein framebasierter Formalismus, der speziell auf die am Lehrstuhl für Prozesstechnik entwickelte Modellierungsmethodik [Mar96b] zurechtgeschnitten ist (siehe [BM98a, Bau00] für eine umfassende Beschreibung dieses Formalismus). Im folgenden wird nur ein grober Überblick gegeben.

VeDa unterscheidet zwischen zwei wesentlichen Aspekten eines Prozessmodells: der *Struktur* und dem *Verhalten*. Desweiteren werden diese beiden Aspekte mehr oder weniger unabhängig voneinander betrachtet. Diese Unabhängigkeit ist notwendig, da zum einen die Struktur eines Modells nicht durch den zu modellierenden Prozess bzw. sein Verhalten und den gewählten Detaillierungsgrad eindeutig und vollständig festgelegt ist. Vielmehr muss die Struktur flexibel an bestimmte Sichten auf den Prozess angepasst werden können. Zum anderen kann das Verhalten eines Prozesses, selbst wenn die Struktur des Modells bereits festgelegt ist, ebenfalls auf viele verschiedene Arten modelliert werden. So kann beispielsweise das Verhalten eines Reaktors in einem Modell durch Gleichungen modelliert werden, die die kinetischen Verhältnisse der stattfindenden Reaktion beschreiben, wohingegen ein anderes Modell nur ein Gleichgewicht der Reaktion erfasst. In beiden Fällen gibt es dann auch noch verschiedene Möglichkeiten, Parameter der Gleichungen festzulegen, z.B. durch aus der Literatur bekannte Heuristiken oder durch experimentell bestimmte Werte.

Wie bereits angedeutet werden in VeDa Objekte zur Beschreibung der Struktur eines Prozessmodells in zwei Klassen aufgeteilt [SM98a]: die *Komponenten* (wie z.B. Reaktoren und Verdampfer) und die *Verknüpfungen* (wie z.B. Signalleitungen und Ventile). Diese Klassen lassen sich dann jeweils noch zu Subklassen spezialisieren wie z.B. Klassen zur Beschreibung spezieller Reaktionsphasen oder spezieller Ventile. Dabei repräsentieren diese Subklassen bzw. ihre Instanzen im allgemeinen aber keine speziellen Funktionalitäten (wie z.B. reagieren, rühren, mischen, ...) oder konkrete Apparate aus chemischen Anlagen. Vielmehr basiert die Einführung von Subklassen meistens auf gemeinsamen, abstrakten Eigenschaften der Instanzen (wie z.B. Heterogenität oder Homogenität von Phasen). Objekte zur Strukturbeschreibung können sowohl atomar als auch zusammengesetzt sein. Motiviert durch den Wunsch nach einer Unterstützung der Erstellung zusammengesetzter Objekte – diese kann sowohl top-down als auch bottom-up erfolgen – werden die Objekte und insbesondere ihre Schnittstellen dabei so beschrieben, dass jeweils von den Eigenschaften und Schnittstellen der Teile auf die Eigenschaften und Schnittstellen des Ganzen geschlossen werden kann.

Das Verhalten von Objekten wird in VeDa durch Prozessvariablen und Gleichungen modelliert [BM98b]. Den Ausgangspunkt für die Zuordnung von Variablen und Gleichungen zu einem Objekt bilden die Phänomene, die dem Objekt in der abstrakten (strukturellen) Beschreibung zugewiesen wurden. Zu jedem dieser Phänomene werden dann eine oder mehrere Variablen eingeführt, diese werden durch Gleichungen zueinander in Beziehung gesetzt und gegebenenfalls noch durch physikalische Gesetze näher bestimmt oder eingeschränkt.

Neben Formalismus und Methodik zur eigentlichen Modellentwicklung umfasst VeDa auch einen Formalismus zur Modellierung von Arbeitsschritten [KLM98]. Da es sich bei der Prozessmodellierung um einen hoch-kreativen Vorgang handelt, kann hierbei das Ziel natürlich nicht darin bestehen, diesen Vorgang zu automatisieren. Es scheint aber durchaus sinnvoll und vielversprechend, den Modellierer bei der Modellentwicklung z.B. durch Vorschläge für den nächsten Arbeitsschritt zu unterstützen. Mögliche nächste Schritte werden dabei mit Hilfe des aktuellen Zustandes des Modells bzw. der Modellentwicklung bestimmt. Sowohl die Zustände als auch die Arbeitsschritte können in VeDa formalisiert werden. Beispielsweise wird ein Arbeitsschritt im wesentlichen beschrieben durch

- sein *Ziel* (wie z.B. die Erzeugung eines neuen Bausteins),
- *Vorbedingungen*, die erfüllt sein müssen, bevor der Schritt ausgeführt werden kann und
- *Nachbedingungen*, die nach der Ausführung des Schrittes erfüllt sind.

Damit dieser Formalismus nun tatsächlich als eine Basis für die Unterstützung des Modellierungsvorgangs dienen kann, sind Syntax und Semantik (insbesondere der Sprachkonstrukte zur Formulierung der Vor- und Nachbedingungen) so zu wählen, dass einerseits anwendungsrelevante Bedingungen ausgedrückt werden können und andererseits auch Implikationen zwischen den Bedingungen effektiv berechenbar sind. Erste Ergebnisse hierzu finden sich in [KLM98, Kro97].

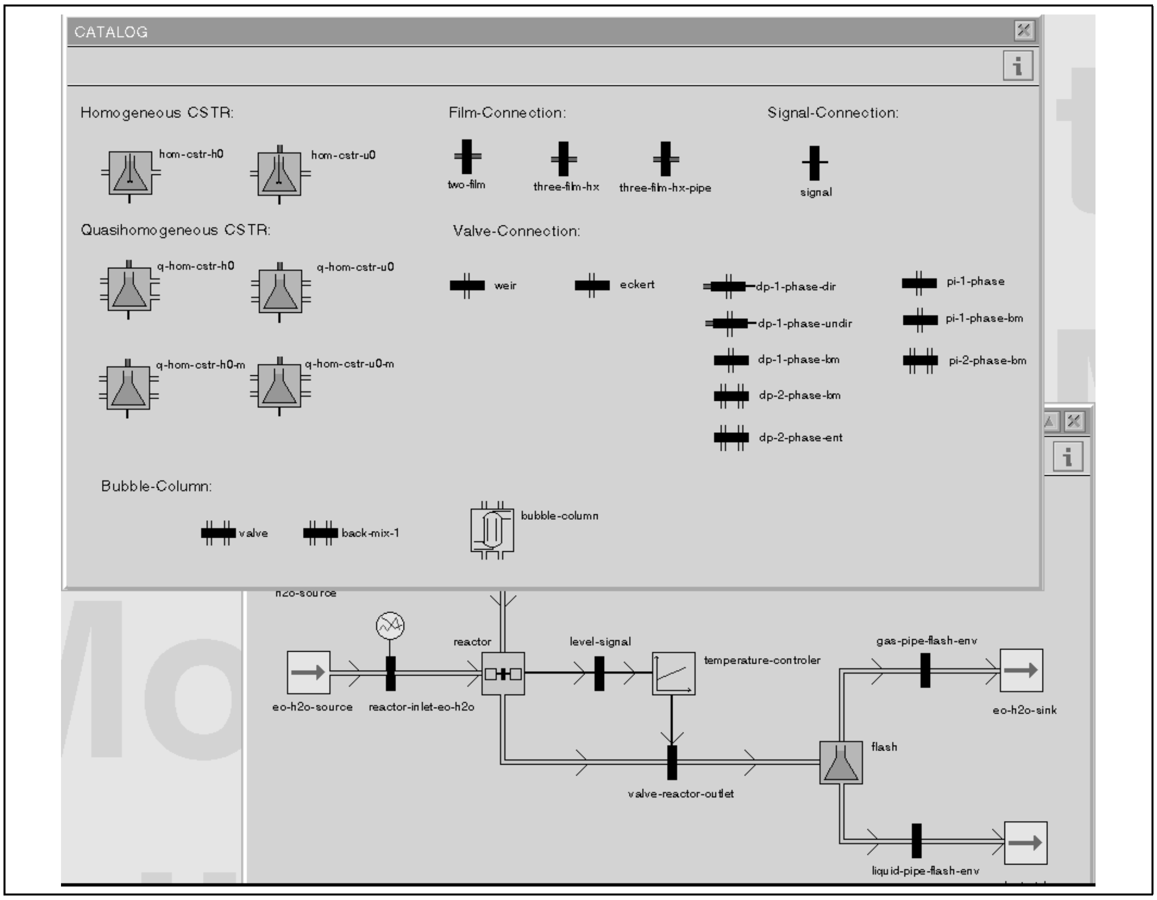


Abbildung 2.5: ModKit-Screenshot: Erstellung der Modellstruktur für den Ethylen-Glykol-Prozess mit Modellbausteinen aus dem Katalog der Modellbibliothek.

2.6 Prozessmodellierung mit ModKit

Die Modellierungsumgebung ModKit [BLM96, BLM97, BLM00] wurde am Lehrstuhl für Prozesstechnik der RWTH Aachen entwickelt, um den in VeDa formalisierten Ansatz zu evaluieren. Die erste Version von ModKit wurde mit Hilfe der wissensbasierten Umgebung G2 [Har93] implementiert, einer Entwicklungsumgebung für den Bau intelligenter, echtzeitfähiger System, die verschiedene Mittel zur Wissensrepräsentation und -verarbeitung (z.B. objektorientierte und regelbasierte Wissensverarbeitung) anbietet. Derzeit wird ModKit einem Re-Engineering unterzogen, bei dem insbesondere durch ein Repository eine verbesserte Repräsentation der Modellbibliothek bereitgestellt werden soll [vWM00, MvWB99].

Die Softwarearchitektur von ModKit ist an den oben eingeführten drei Dimensionen der Modellentwicklung ausgerichtet [BLM97]: aufbauend auf die Modellbibliothek wird die Repräsentationsdimension durch die Bereitstellung eines Hypertexteditors für natürlichsprachliche Beschreibungen, eines Modellstruktureditors für die semi-

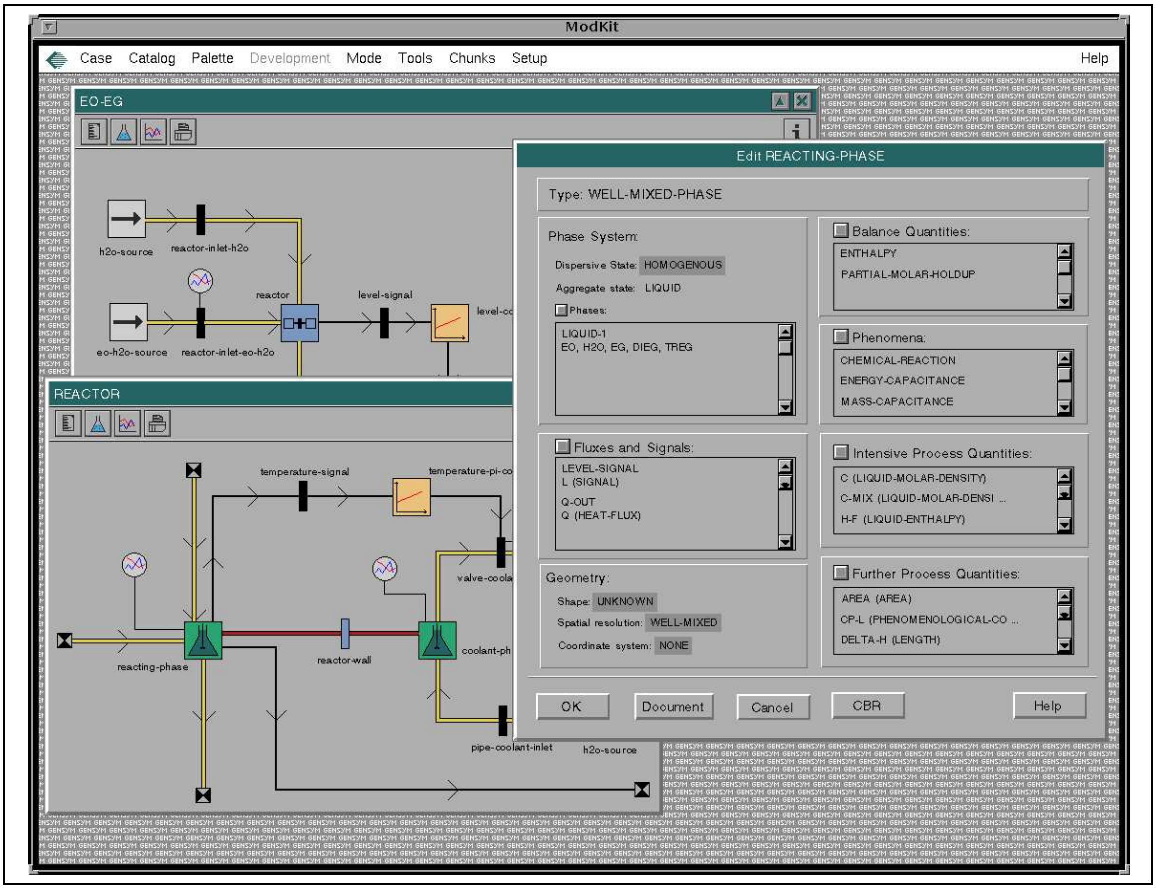


Abbildung 2.6: ModKit-Screenshot: Dialog zur Spezifikation der Attribute zum Ethylen-Glykol-Prozess.

formale Darstellung von Modellstrukturen und Eigenschaften von Bausteinen sowie eines Verhaltenseditors für die formale Definition von Prozessgrößen und Gleichungen realisiert. Die Spezifikationsdimension wird durch Klassenbrowser repräsentiert, die vordefinierte Komponenten, Gleichungen, etc. zur Auswahl anbieten. Der Konsensdimension schließlich entspricht ein Editor, der die Dokumentation während des Modellierungsprozesses getroffener Entscheidungen unterstützt. In Kapitel 9 wird im Zusammenhang mit der Anbindung des Beschreibungslogik-Systems FaCT an die Modellierungsumgebung am Lehrstuhl für Prozesstechnik noch genauer auf die zugrundeliegende Architektur eingegangen.

Für die Erstellung der strukturellen Beschreibung eines Modells bietet der Modellstruktureditor dem Anwender jeweils den Katalog von Bausteinen an. Der Modellierer kann die Bausteine von diesem Katalog in seinen Arbeitsbereich transferieren, dort an den zu modellierenden Prozess anpassen und sie zu einem Modell zusammenfügen. Abbildung 2.5 zeigt einen Auszug aus dem Katalog sowie die in ModKit erstellte Struktur des Ethylen-Glykol-Prozesses. Das Verhalten der Bausteine im Modell wird mit Hilfe des Verhaltenseditors festgelegt: durch Auswahl entsprechender Attribute in einem Dialog erfolgt eine qualitative Verhaltensbeschreibung (s. Abbildung 2.6). Aus dem so erhaltenen Modell kann dann automatisch der Quellcode für die integrierten Simulationswerkzeuge generiert werden, die dann Simulation und Analyse des Prozesses ermöglichen. Darüberhinaus wird der Modellierer gemäß der in VeDa eingeführten Arbeitsschrittunterstützung durch den Modellentwicklungsprozess geleitet. Da es sich hierbei aber um einen hochkreativen Prozess handelt, ist diese Unterstützung natürlich nur partiell und besteht jeweils nur aus Vorschlägen möglicher nächster Schritte hin zu einem vollständigen Prozessmodell.

2.7 Wissensrepräsentation und ModKit

Offensichtlich hängt eine erfolgreiche Realisierung des oben beschriebenen Szenarios zur Modellentwicklung entscheidend davon ab, dass Modellbausteine effektiv wiederverwendet werden. Folglich müssen sie in der Modellbibliothek so gespeichert werden, dass sie leicht wiedergefunden werden können. VeDa sieht hierfür die Zusammenfassung von ähnlichen Bausteinen zu Klassen vor, die dann mit Hilfe der Vererbungsbeziehung angeordnet werden. Gemäß der Semantik von VeDa stimmt die Vererbungsbeziehung mit der „ist-Spezialisierung-von“-Relation auf den Klassen überein, d.h. erbt eine Klasse B Eigenschaften einer Klasse A, so sollte B auch spezieller als A sein. Dabei heisst eine Klasse B *spezieller* als eine Klasse A, wenn jede Instanz von B auch Instanz von A ist. Ordnet man die Klassen und damit auch die Bausteine gemäß dieser Ordnung an, so erhält man eine Klassenhierarchie, die dem Modellierer eine zielgerichtete Suche nach Bausteinen und damit auch ihre effektive Wiederverwendung ermöglicht.

Um allerdings wirklich diesen Nutzen aus der Strukturierung der Modellbibliothek zu ziehen, muss die tatsächliche Klassenhierarchie (inklusive der Instanzbeziehungen zwischen Klassen und Bausteinen) stets konsistent zur Spezialisierungsordnung sein. Da die Modellbibliothek von ModKit mittlerweile mehrere hundert Klassen und entsprechend viele Bausteine umfasst, ist eine Realisierung der Instanz- und Spezialisierungsbeziehungen „von Hand“ nicht mehr möglich. Stattdessen sollen nun Struktur und Verhalten von Klassen und Bausteinen durch einen Formalismus beschrieben werden, der es erlaubt, automatische Inferenzdienste einzusetzen, um dadurch die Konsistenz der Struktur zu garantieren. Zu den gewünschten Diensten zählen

- der Konsistenztest für Klassenbeschreibungen,
- die Berechnung von Spezialisierungsbeziehungen zwischen Klassen und
- die Berechnung von Instanzbeziehungen zwischen Bausteinen und Klassen.

Motiviert durch die Suche nach einem Repräsentationsformalismus, der einerseits ausdrucksstark genug ist, relevante Eigenschaften von Bausteinen und Klassen zu beschreiben und andererseits die automatische Berechnung der gewünschten Inferenzen erlaubt, wurde in der Kooperation zwischen dem Lehr- und Forschungsgebiet Theoretische Informatik und dem Lehrstuhl für Prozesstechnik an der RWTH Aachen gezeigt, dass sich hier Beschreibungslogiken (BLen) sehr gut eignen [Sat95, BS96b, Sat98]: BLen stellen zum einen die nötige Ausdrucksstärke bereit [BS96a, BS96c, HS99, Sat00] und zum anderen sind auch für ausdrucksstarke BLen effiziente Implementierungen der Inferenzdienste möglich [BFT95, Hor98, HST99, HST00].

In der vorliegenden Arbeit ist das Hauptaugenmerk nun auf weitere Dienste gerichtet, die bereits zu Beginn der Kooperation als sinnvolle und notwendige Ergänzung zu den in [Sat98] ausführlich untersuchten Diensten erkannt wurden: Dienste, die die Erweiterung der Klassenhierarchie unterstützen. Erweiterungen der Klassenhierarchie um neue Klassen werden notwendig, wenn (*i*) Klassen zu viele Bausteine enthalten bzw. keine hinreichend speziellen Klassen für neue Bausteine existieren; oder (*ii*) eine Klasse zu viele direkte Unterklassen besitzt und durch neue Klassen eine weitere Abstraktionsebene, die das Browsen der Hierarchie erleichtern soll, eingefügt wird. Kapitel 4 wird das Szenario zum Einsatz von BL-Systemen im Rahmen der Modellierungsumgebung ModKit ausführlich beschreiben. Zuvor werden im folgenden Kapitel aber noch die notwendigen Grundlagen zu BLen und die oben erwähnten Inferenzdienste eingeführt.

Kapitel 3

Terminologische Wissensrepräsentationssysteme

In diesem Kapitel werden die für diese Arbeit relevanten Grundlagen zu terminologischen Wissensrepräsentationssystemen (TWRS), genauer zu den ihnen zugrundeliegenden Beschreibungslogiken (BLen), eingeführt.

Beschreibungslogiken erlauben es, mit Hilfe sogenannter *Konzeptbeschreibungen* die in einem Anwendungsbereich relevanten Konzepte formal zu definieren. Durch Schlussfolgerungsverfahren wird dieses Wissen strukturiert, wobei auch implizit vorhandenes Wissen (z.B. Ober-/Unterkonzeptbeziehungen oder Instanzbeziehungen) explizit gemacht wird. Kennzeichnend für diesen Wissensrepräsentationsformalismus ist dabei, dass er über eine formale, logik-basierte Semantik verfügt.

Im folgenden Abschnitt wird zunächst ein kurzer historischer Überblick über die Entwicklung des Forschungsbereichs Beschreibungslogik gegeben, wobei der Schwerpunkt auf den sogenannten Standardinferenzen liegt. Es folgen die formale Einführung von Syntax und Semantik (Abschnitt 3.2) sowie der Standardinferenzen *Konsistenz, Subsumtion* und *Instanz* für BLen (Abschnitt 3.3). In Abschnitt 3.4 werden dann Nicht-Standardinferenzen motiviert, kurz historisch eingeordnet und formal definiert, wobei das Hauptaugenmerk auf den in den technischen Kapiteln formal untersuchten Problemen Berechnung des *Most Specific Concepts* (MSC) und Berechnung des *Least Common Subsumers* sowie dem *minimalen Rewriting-Problem* liegt.[1] Zum Abschluss des Kapitels wird ein vollständiger Überblick über die neuen Resultate dieser Arbeit gegeben.

[1]In Zusammenhang mit den Nicht-Standardinferenzen werden in dieser Arbeit die aus der Literatur bekannten und auch im deutschen gebräuchlichen englischen Bezeichnungen und Abkürzungen verwendet.

3.1 Historisches

Als Ursprung der Beschreibungslogiken werden die Wissensrepräsentationsformalismen *Semantische Netze* [Qui68] und *Frames* [Min75] angesehen. In Semantischen Netzen werden Konzepte und Individuen in beschrifteten Graphen modelliert, während in Frames Wissen in tabellenartigen Datenstrukturen dargestellt wird. In keinem der beiden Formalismen gibt es jedoch eine formale Semantik, sodass sich die Bedeutung der syntaktischen Strukturen unmittelbar aus den auf ihnen operierenden Inferenzdiensten ergibt und damit von der Intuition der Systementwickler abhängig ist. Motiviert durch die sich daraus ergebenden Probleme [Woo75, Bra77, Bra83] wurde unter der Leitung von Brachman das auf *Strukturierten Vererbungsnetzen* [Bra78] basierende System KL-ONE entwickelt [BS85]. Die Bedeutung der graphischen Konstruktoren in diesen strukturierten Vererbungsnetzen wurde dabei eindeutig durch eine formale Semantik festgelegt [BL84, LB87]. In diesem Sinne bildeten sie die ersten Beschreibungslogiken.

Ausgehend von der formalen Semantik konnten dann auch die den gewünschten Inferenzdiensten zur Berechnung der Ober/-Unterkonzeptbeziehung zwischen Konzepten und der Instanzbeziehung zwischen Individuen und Konzepten zugrundeliegenden Inferenzprobleme *Subsumtion* und *Instanz* formal erklärt werden (cf Abschnitt 3.3). Bereits in ersten Untersuchungen stellte man fest, dass diese Probleme für KL-ONE unentscheidbar sind [Sch89] und dass auch kleine Änderungen an der Ausdrucksstärke einer BL erhebliche Auswirkungen auf die Komplexität der Inferenzprobleme in dieser BL haben können [LB87]. Die durch diese Arbeiten ausgelösten umfangreichen Untersuchungen von Entscheidbarkeit und Komplexität der Inferenzprobleme für BLen verschiedenster Ausdrucksstärke lassen sich nach [BS00] grob in vier (teilweise zeitlich überlappende) Phasen einordnen:

Phase 1: Implementierung der ersten Systeme. Das KL-ONE-System hatte schnell eine Vielzahl von Folgesystemen, z.B. NIKL [KBR86], KRYPTON [BFL83], BACK [Pel91], K-REP [MDW91], SB-ONE [Kob91] und LOOM [Mac91] (das einzige System, dass derzeit noch aktiv „auf dem Markt“ ist). All diese Systeme haben gemeinsam, dass sie einerseits auf recht ausdrucksstarken BLen aufbauen und andererseits sogenannte strukturelle Subsumtionsalgorithmen einsetzen. Diese meist sehr effizienten Algorithmen vergleichen zur Berechnung von Subsumtionsbeziehungen die syntaktische Struktur der Konzeptbeschreibungen. Durch die Effizienz geht aber im Falle ausdrucksstarker BLen Vollständigkeit verloren. Allerdings war den Entwicklern der Systeme zum einen diese Unvollständigkeit nicht immer bewusst und zum anderen herrschte (auch bedingt durch Randbedingungen, die sich aus der zur Verfügung stehenden Hardware ergaben) die Meinung, dass nur polynomielle Schlussfolgerungsverfahren sinnvoll einsetzbar sind.

Phase 2: Erste Komplexitäts- und Unentscheidbarkeitsresultate. Teilweise parallel zur ersten Phase begannen die ersten formalen Untersuchungen der Standardschlussfolgerungsprobleme. Es stellte sich heraus, dass bereits sehr ausdrucksschwache Beschreibungslogiken keine polynomiellen Subsumtionsalgorithmen mehr zulassen [BL84, Neb90b] und dass die im System KL-ONE verwendete Sprache sogar ein unentscheidbares Subsumtionsproblem hat [Sch89]. Insbesondere ergab sich daraus die Unvollständigkeit der in den oben erwähnten Systemen verwendeten Algorithmen. Die Reaktion auf diese Resultate war zum einen, nun „bewusst" zugunsten der Ausdrucksstärke auf Vollständigkeit zu verzichten. Zum anderen wurden ausdrucksschwache Beschreibungslogiken betrachtet, die gerade noch polynomielle Standardschlussfolgerungen erlauben. Die zweite Vorgehensweise liegt beispielsweise dem bei AT&T entwickelten System Classic [BMPS+91, BP94] zugrunde.

Phase 3: Entwicklung tableau-basierter Algorithmen für ausdrucksstarke Beschreibungslogiken sowie genaue Untersuchung der Komplexität. Für ausdrucksstärkere Beschreibungslogiken (insbesondere Logiken, die Disjunktion und Negation zulassen) führt der strukturelle Ansatz nicht mehr zu vollständigen Schlussfolgerungsverfahren. Für derartige Logiken haben sich tableau-basierte Algorithmen als erfolgreicher herausgestellt. Diese Algorithmen sind auch für ausdrucksstarke Beschreibungslogiken korrekt und vollständig (d.h. sie finden genau die gültigen Subsumtionsbeziehungen), haben aber deshalb häufig eine höhere Komplexität als die strukturellen Algorithmen. Der erste derartige Algorithmus wurde von Schmidt-Schauß und Smolka [SS91] für die Beschreibungslogik $\mathcal{ALC}$ entwickelt. Darauf aufbauend wurden Tableau-Algorithmen für Subsumtion und das Instanzproblem für eine Vielzahl von Beschreibungslogiken verschiedener Ausdrucksstärke entworfen (siehe z.B. [HNS90, HB91, BH91a, Baa91, BBN+93, BBH96, BS96a, BS96c, Sat96, BS98]). Zusätzlich wurde die (worst-case) Komplexität der Schlussfolgerungsprobleme sehr gründlich untersucht (siehe z.B. [DLNN91a, DLNN91b, DHL+92, Lut99, Tob99]). Die ersten auf diesen Algorithmen beruhenden Systeme KRIS [BH91b] und Crack [BFT95] zeigten, dass tableau-basierte Algorithmen durchaus zu vertretbaren Laufzeitresultaten führen [BFH+94]. Stärker optimierte neuere Systeme, wie FaCT [Hor98, HT00], konnten diese Laufzeitresultate noch wesentlich verbessern.

Phase 4: Algorithmen und effiziente Systeme für sehr ausdrucksstarke Beschreibungslogiken. Motiviert durch Anwendungen (unter anderem im Datenbankbereich [CGL98a, CGL+98c, CLN99, CGL99]) wurden Beschreibungslogiken untersucht, deren Ausdrucksstärke weit über die von $\mathcal{ALC}$ hinausgeht. Erste Entscheidbarkeits- und Komplexitätsresultate für derartige Logiken konnten mit Hilfe des von Schild [Sch91] aufgezeigten Zusammenhangs zwischen PDL (propositional dynamic logic) und Beschreibungslogiken erhalten werden. Die Idee der von Lenzerini und De Giacomo perfektionierten Methode ist hierbei, die beschreibungslogischen Formeln erfüllbarkeitserhaltend (und mit polynomiellem Aufwand) in PDL-Formeln

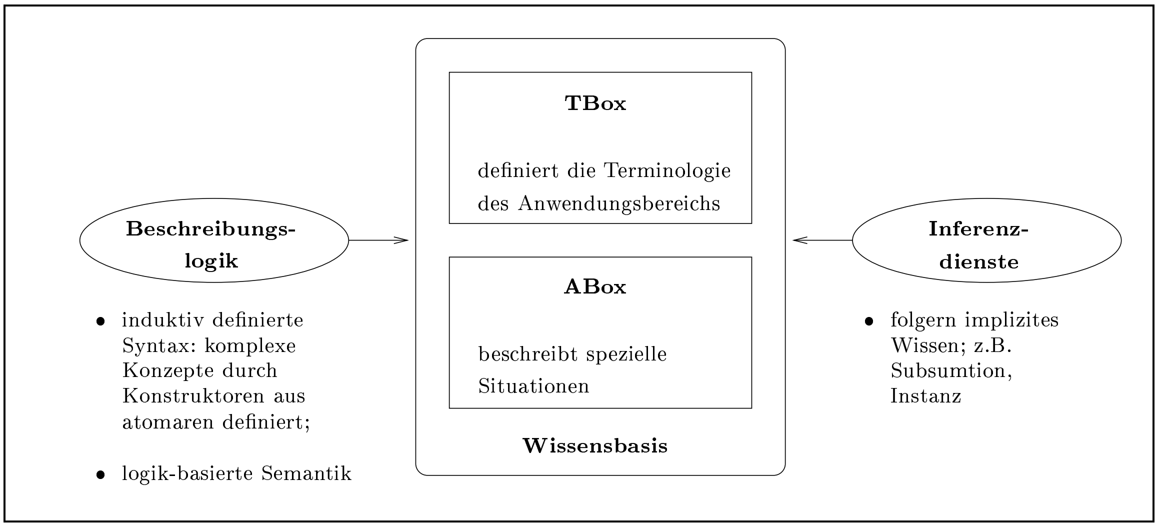


Abbildung 3.1: Die Architektur eines DL-Systems.

zu übersetzen und dann die bekannten (Exptime-) Algorithmen für PDL einzusetzen [GL94a, GL94b, Gia95, Gia96, GL96]. Da diese Übersetzung aber bereits für kleine Eingabeformeln PDL-Formeln liefert, die für die nach unserem Wissen derzeit einzige effiziente Implementierung der PDL-Erfüllbarkeitsalgorithmen [HP98] nicht handhabbar sind, wird derzeit verstärkt an praktikablen tableau-basierten Algorithmen für sehr ausdruckstarke Beschreibungslogiken gearbeitet. Erste Ergebnisse hierzu liegen bereits vor [HS99, HST99, HST00] und werden derzeit in das System FaCT integriert. Dieses System besitzt bereits eine sehr effiziente Implementierung eines tableau-basierten Subsumtionsalgorithmus für eine ausdrucksstarke Erweiterung von $\mathcal{ALC}$ und wird auch in der in dieser Arbeit betrachteten Prozesstechnik-Anwendung erfolgreich eingesetzt.

3.2 Syntax und Semantik

Das Herz eines BL-Systems ist die *Wissensbasis*, die wiederum aus zwei Komponenten besteht: der *TBox* und der *ABox* (s. Abbildung 3.1). In der TBox wird das konzeptuelle Wissen über den betrachteten Anwendungsbereich, das Vokabular, hinterlegt, wohingegen die ABox assertionales Wissen über spezielle Situationen im Anwendungsbereich enthält. Die in einer solchen Wissensbasis auftretenden Konzepte werden durch *Konzeptbeschreibungen* der dem System zugrundeliegenden BL repräsentiert. Den zweiten wichtigen Aspekt eines BL-Systems bilden die vom System zur Verfügung gestellten Inferenzdienste, die es erlauben, implizites Wissen aus dem explizit hinterlegten Wissen zu folgern. Standardmäßig, d.h. in „allen“ Systemen angeboten, sind dies Konsistenz, Subsumtion und Instanz. Wir werden diese im Abschnitt 3.3 formal einführen und nun zunächst Syntax und Semantik von Konzeptbeschreibungen, TBoxen und ABoxen definieren.

3.2.1 Konzeptbeschreibungen

Konzeptbeschreibungen werden ausgehend von einer Menge N_C von *Konzeptnamen* (unären Prädikaten) und einer Menge N_R von *Rollennamen* (binären Prädikaten) induktiv mit Hilfe einer Menge von Konstruktoren aufgebaut. Die folgende Definition führt die für diese Arbeit relevanten Konstruktoren ein (siehe auch Tabelle 3.1). Einen Überblick über (fast) alle in der Literatur betrachteten Konstruktoren findet man z.B. in [BBH+91].

Definition 3.1 (Syntax) *Sei N_C eine Menge von Konzeptnamen und N_R eine Menge von Rollennamen.* Konzeptbeschreibungen *werden induktiv wie folgt definiert:*

- $\top$, $\bot$ *und jeder Konzeptname ist eine Konzeptbeschreibung;*
- *sind C, D Konzeptbeschreibungen, $P \in N_C$ ein Konzeptname, $r \in N_R$ ein Rollenname und $n \in \mathbb{N}$, so sind auch*

  $\neg P$ *(primitive Negation),*
  $\neg C$ *(Negation),*
  $C \sqcap D$ *(Konjunktion),*
  $C \sqcup D$ *(Disjunktion),*
  $\forall r.C$ *(Werterestriktion),*
  $\exists r.C$ *(Existenzrestriktion),*
  $(\geq n\ r)$ *(Minimum-Zahlenrestriktion) und*
  $(\leq n\ r)$ *(Maximum-Zahlenrestriktion)*

  Konzeptbeschreibungen.

Als Anwendungsbereich für ein fortlaufendes Beispiel betrachten wir in diesem Kapitel den Bereich Familie.

Beispiel 3.2 *Als Mengen von Konzept- bzw. Rollennamen seien gegeben*

$$\begin{aligned} N_C &= \{\mathsf{Mensch}, \mathsf{Männlich}, \mathsf{Gross}, \mathsf{Reich}, \mathsf{Doktor}\}, \\ N_R &= \{\mathsf{hatKind}, \mathsf{verheiratet}\}. \end{aligned}$$

Die $\mathcal{ALE}$-Konzeptbeschreibung

$$\mathsf{Mensch} \sqcap \mathsf{Männlich} \sqcap \exists \mathsf{hatKind}.(\mathsf{Mensch} \sqcap \mathsf{Gross}) \sqcap \forall \mathsf{hatKind}.\mathsf{Reich}$$

beschreibt alle Väter, die ein großes Kind haben und deren Kinder alle reich sind. Demgegenüber beschreibt die folgende $\mathcal{ALE}$-Konzeptbeschreibung alle männlichen Doktoren, die (falls sie verheiratet sind) mit Frauen verheiratet sind:

$$\mathsf{Doktor} \sqcap \mathsf{Männlich} \sqcap \forall \mathsf{verheiratet}.(\mathsf{Mensch} \sqcap \neg\mathsf{Männlich}).$$

Konstruktor	Syntax	Semantik
Top	$\top$	$\Delta_{\mathcal{I}}$
Bottom	$\bot$	$\emptyset$
Primitive Negation ($P \in N_C$)	$\neg P$	$\Delta_{\mathcal{I}} \setminus P^{\mathcal{I}}$
Negation	$\neg C$	$\Delta_{\mathcal{I}} \setminus C^{\mathcal{I}}$
Konjunktion	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
Disjunktion	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
Existenzrestriktion	$\exists r.C$	$\{x \in \Delta_{\mathcal{I}} \mid \exists y : (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
Werterestriktion	$\forall r.C$	$\{x \in \Delta_{\mathcal{I}} \mid \forall y : (x,y) \in r^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$
Minimum-Zahlenrestriktion	$(\geq n\ r)$	$\{x \in \Delta_{\mathcal{I}} \mid\ \lvert\{y \in \Delta_{\mathcal{I}} \mid (x,y) \in r^{\mathcal{I}}\}\rvert \geq n\}$
Maximum-Zahlenrestriktion	$(\leq n\ r)$	$\{x \in \Delta_{\mathcal{I}} \mid\ \lvert\{y \in \Delta_{\mathcal{I}} \mid (x,y) \in r^{\mathcal{I}}\}\rvert \leq n\}$

Tabelle 3.1: Syntax und Semantik von Konzeptbeschreibungen.

Die Ausdrucksstärke einer Beschreibungslogik $\mathcal{L}$ ergibt sich aus der Menge der Konstruktoren, die zur Bildung von Konzeptbeschreibungen in $\mathcal{L}$ verwendet werden dürfen. Tabelle 3.2 fasst die in dieser Arbeit betrachteten Beschreibungslogiken zusammen. $\mathcal{ALC}$ wurde von Schmidt-Schauß und Smolka in [SS91] eingeführt und bildete den Ausgangspunkt für die Untersuchung von Ausdrucksstärke und Komplexität vieler ausdruckstarker BLen. In dieser Arbeit werden wir uns im wesentlichen mit der Klasse der Subsprachen von $\mathcal{ALC}$ beschäftigen, die Existenzrestriktionen umfassen, also die BLen $\mathcal{EL}$, $\mathcal{FLE}$ und $\mathcal{ALE}$. Die Beschreibungslogik $\mathcal{ALN}$, die als Kernsprache des BL-Systems Classic angesehen werden kann, spielt also in dieser Arbeit eine untergeordnete Rolle. Die BL $\mathcal{FL}_0$ bildet den gemeinsamen Kern aller heute anwendungsrelevanten BLen und ihre Betrachtung ist daher trotz ihrer geringen Ausdrucksstärke durchaus von theoretischem Interesse.

Die genaue Bedeutung einer Konzeptbeschreibung ergibt sich aus der modell-theoretischen Definition der Semantik.

Definition 3.3 (Semantik) *Sei $\mathcal{I} = (\Delta_{\mathcal{I}}, \cdot^{\mathcal{I}})$ eine* Interpretation *mit nicht-leerem* Wertebereich $\Delta_{\mathcal{I}}$ *und einer* Interpretationsfunktion $\cdot^{\mathcal{I}}$, *die jedem Konzeptnamen $P \in N_C$ eine Teilmenge $P^{\mathcal{I}} \subseteq \Delta_{\mathcal{I}}$ und jedem Rollennamen $r \in N_R$ eine binäre Relation $r^{\mathcal{I}} \subseteq \Delta_{\mathcal{I}} \times \Delta_{\mathcal{I}}$ zuordnet. Die induktive Erweiterung von $\cdot^{\mathcal{I}}$ auf beliebige Konzeptbeschreibungen ist in der dritten Spalte von Tabelle 3.1 angegeben.*

Es sei an dieser Stelle angemerkt, dass die Semantik von Konzeptbeschreibungen auch durch eine Übersetzung von Konzeptbeschreibungen in Formeln der Prädikatenlogik erster Stufe festgelegt werden kann. Auf diese alternative Definition der Semantik soll hier aber nicht näher eingegangen werde, da sie im folgenden keine Rolle spielen wird (für die genaue Definition (insbesondere der Übersetzung) siehe z.B. [Bor96]).

Konstruktor	$\mathcal{FL}_0$	$\mathcal{EL}$	$\mathcal{FLE}$	$\mathcal{ALE}$	$\mathcal{ALC}$	$\mathcal{ALN}$
Top	x	x	x	x	x	x
Bottom				x	x	x
Primitive Negation				x	x	x
Negation					x	
Konjunktion	x	x	x	x	x	x
Disjunktion					x	
Existenzrestriktion		x	x	x	x	
Werterestriktion	x		x	x	x	x
Minimum-Zahlenrestriktion						x
Maximum-Zahlenrestriktion						x

Tabelle 3.2: Die betrachteten Beschreibungslogiken.

3.2.2 TBoxen

Die Terminologie eines Anwendungsbereichs wird in der sogenannten *TBox* in Form von *Konzeptdefinitionen* hinterlegt. Intuitiv führt eine Konzeptdefinition einen *definierten Konzeptnamen* für ein Konzept ein, welches durch eine Konzeptbeschreibung repräsentiert wird. In diesem Sinne lässt sich ein definierter Konzeptname als Abkürzung für ein relevantes Konzept in der Anwendung betrachten.

Definition 3.4 (TBox) *Eine* Konzeptdefinition *ist von der Form $A \doteq C$, wobei $A \in N_C$ ein Konzeptname und C eine Konzeptbeschreibung sei. Eine* TBox $\mathcal{T}$ *ist eine endliche Menge von Konzeptdefinitionen, wobei jeder Konzeptname höchstens einmal auf der linken Seite einer Konzeptdefinition auftreten darf. Ein Konzeptname $A \in N_C$ heißt* definiert *(in $\mathcal{T}$), wenn er auf der linken Seite einer Konzeptdefinition in $\mathcal{T}$ auftritt; sonst heißt er* primitiv. *Die Konzeptbeschreibung C in $A \doteq C$ wird als das* definierende Konzept *zu A bezeichnet.*

Treten in einer TBox $\mathcal{T}$ nur Konzeptbeschreibungen einer BL $\mathcal{L}$ auf, so wird $\mathcal{T}$ auch als $\mathcal{L}$-*TBox* bezeichnet. Prinzipiell dürfen definierte Konzeptnamen zur Bildung von definierenden Konzepten verwendet werden. Betrachtet man allerdings $\mathcal{L}$-TBoxen für eine BL $\mathcal{L}$, die nur primitive Negation erlauben, so dürfen definierte Konzeptnamen nicht negiert auftreten, d.h. Negation darf nur unmittelbar vor primitiven Konzeptnamen auftreten.

Eine TBox heißt *zyklisch*, wenn die Definition eines Konzeptnamens A direkt oder indirekt von A selbst abhängt. Formal *hängt $A \in N_C$ direkt von $B \in N_C$ ab*, wenn B im definierenden Konzept zu A auftritt. Bezeichnet nun *abhängen* den transitiven Abschluss der Relation *direkt abhängen*, so heißt die TBox $\mathcal{T}$ *zyklisch*, wenn ein definierter Name A in $\mathcal{T}$ existiert, der von sich selbst abhängt; sonst heißt $\mathcal{T}$ *azyklisch*. Wir werden uns im folgenden auf die Betrachtung azyklischer TBoxen beschränken.

Zyklische TBoxen, insbesondere Charakterisierungen verschiedener möglicher Semantiken, werden z.B. in [Neb91, Baa96, Küs98] betrachtet.

Beispiel 3.5 *Für ein Beispiel für eine azyklische $\mathcal{ALE}$-TBox $\mathcal{T}$ erweitern wir die Menge von Konzeptnamen aus Beispiel 3.2 um die Menge*

$$\{\mathsf{Mann}, \mathsf{Frau}, \mathsf{Vater}, \mathsf{Mutter}, \mathsf{VaterVonS\ddot{o}hnen}\}.$$

Die folgende Menge von Konzeptdefinitionen bildet offensichtlich eine azyklische $\mathcal{ALE}$-TBox:

$$\begin{array}{lrcl}
\mathcal{T} = \{ & \mathsf{Mann} & \doteq & \mathsf{Mensch} \sqcap \mathsf{M\ddot{a}nnlich} \\
& \mathsf{Frau} & \doteq & \mathsf{Mensch} \sqcap \neg\mathsf{M\ddot{a}nnlich} \\
& \mathsf{Vater} & \doteq & \mathsf{Mann} \sqcap \exists \mathsf{hatKind}.\mathsf{Mensch} \\
& \mathsf{Mutter} & \doteq & \mathsf{Frau} \sqcap \exists \mathsf{hatKind}.\mathsf{Mensch} \\
& \mathsf{VaterVonS\ddot{o}hnen} & \doteq & \mathsf{Vater} \sqcap \forall \mathsf{hatKind}.\mathsf{Mann} \\
& \mathsf{ReicheEltern} & \doteq & \mathsf{Reich} \sqcap \forall \mathsf{hatKind}.\mathsf{Reich}\}
\end{array}$$

Es sei noch angemerkt, dass in der Literatur auch TBoxen betrachtet werden, die *Axiome* der Form $A \sqsubseteq C$ zur Beschreibung notwendiger Bedingungen an den definierten Namen $A \in N_C$ erlauben oder sogar *general concept inclusions* (GCIs) der Form $C \sqsubseteq D$, wobei C *und* D Konzeptbeschreibungen sein können (siehe z.B. [Cal96, Gia95, GL96, HST99]). Solche TBoxen werden beispielsweise zur Beschreibung von Schemata in Datenmodellen eingesetzt [CLN99, CGL+98b]. Im Rahmen dieser Arbeit wird diese Art von TBox jedoch nicht weiter betrachtet. Vielmehr wird im folgenden stets von *azyklischen TBoxen* ausgegangen, wie sie in Definition 3.4 eingeführt wurden.

Die Semantik von TBoxen ist gegeben durch die Menge ihrer Modelle.

Definition 3.6 (Modell einer TBox) *Eine Interpretation $\mathcal{I} = (\Delta_\mathcal{I}, \cdot^\mathcal{I})$ heißt* Modell der TBox $\mathcal{T}$ *genau dann, wenn $\mathcal{I}$ alle Konzeptdefinitionen in $\mathcal{T}$ erfüllt, d.h. wenn $A^\mathcal{I} = C^\mathcal{I}$ für alle Konzeptdefinitionen $A \doteq C$ in $\mathcal{T}$ gilt.*

Im Falle einer azyklischen TBox $\mathcal{T}$ existiert zu einer sogenannten *primitiven Interpretation* $\mathcal{J} = (\Delta_\mathcal{J}, \cdot^\mathcal{J})$, die nur die primitiven Konzeptnamen und die Rollennamen interpretiert, stets eine eindeutige Erweiterung von $\mathcal{J}$ zu einem Modell $\mathcal{I}$ von $\mathcal{T}$. Eine *Erweiterung* von $\mathcal{J}$ hat den gleichen Wertebereich $\Delta_\mathcal{J}$ und stimmt bei der Interpretation der primitiven Konzeptnamen und der Rollennamen mit $\mathcal{J}$ überein. Die Interpretation der definierten Konzeptnamen ergibt sich nun unmittelbar aus den *aufgefalteten* Konzeptbeschreibungen zu den definierenden Konzepten in $\mathcal{T}$.

Eine Konzeptbeschreibung C heißt *aufgefaltet (bzgl. einer TBox $\mathcal{T}$)*, wenn C keine definierten Namen aus $\mathcal{T}$ enthält und eine TBox $\mathcal{T}$ heißt *aufgefaltet*, wenn kein definierendes Konzept in $\mathcal{T}$ einen definierten Konzeptnamen enthält. Offensichtlich lässt sich zu einer azyklischen TBox $\mathcal{T}$ stets eine aufgefaltete TBox $\mathcal{T}'$ bestimmen, indem definierte Namen auf den rechten Seiten erschöpfend durch ihre definierenden Konzepte ersetzt werden. Konzeptdefinitionen in $\mathcal{T}'$ sind also von der Form $A \doteq C'$, wobei C' eine Konzeptbeschreibung ist, die keine definierten Namen enthält. In Beispiel 3.5 ist die aufgefaltete Konzeptbeschreibung zum definierten Konzeptnamen $\mathsf{VaterVonS\ddot{o}hnen}$ gegeben durch

$$\mathsf{Mensch} \sqcap \mathsf{M\ddot{a}nnlich} \sqcap \exists \mathsf{hatKind}.\mathsf{Mensch} \sqcap \forall \mathsf{hatKind}.(\mathsf{Mensch} \sqcap \mathsf{M\ddot{a}nnlich}).$$

Erweitert man nun eine primitive Interpretation $\mathcal{J} = (\Delta, \cdot^{\mathcal{J}})$ zu einer Interpretation $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ mit $A^{\mathcal{I}} := C'^{\mathcal{J}}$ für alle Konzeptdefinitionen $A \doteq C'$ in $\mathcal{T}'$, so ist $\mathcal{I}$ trivialerweise ein Modell von $\mathcal{T}'$ und man sieht leicht, dass $\mathcal{I}$ die eindeutige Erweiterung von $\mathcal{J}$ zu einem Modell von $\mathcal{T}$ ist.

Man beachte allerdings, dass das Auffalten einer TBox exponentiellen Aufwand verursachen kann: die Größe von $\mathcal{T}'$ kann exponentiell in der Größe von $\mathcal{T}$ sein [Neb90a]. Daher muss bei Komplexitätsresultaten zu Inferenzproblemen, die TBoxen einbeziehen (wie z.B. Subsumtion modulo TBox oder auch das Rewriting von Konzeptbeschreibungen), zwischen aufgefalteten und nicht aufgefalteten TBoxen unterschieden werden.

Im Falle zyklischer TBoxen existiert im allgemeinen keine eindeutige Erweiterung einer primitiven Interpretation zu einem Modell der zyklischen TBox. Tatsächlich sind hier je nach Anwendung unterschiedliche Semantiken für die zyklischen Konzeptbeschreibungen zu betrachten [Neb91] (siehe z.B. [Baa96] und [Küs98] für eine automatentheoretische Charakterisierung verschiedener Semantiken für zyklische $\mathcal{FL}_0$- bzw. $\mathcal{ALN}$-TBoxen).

3.2.3 ABoxen

Konkretes, situationenbezogenes Wissen in einem Anwendungsbereich wird in sogenannten ABoxen modelliert. Konkrete Objekte werden benannt und als Individuen in der ABox eingeführt. Ihre Eigenschaften werden durch Zuordnen von Individuen zu Konzeptbeschreibungen festgelegt und Beziehungen zwischen Objekten werden durch die Verknüpfung der zugehörigen Individuen durch Rollennamen dargestellt. Im folgenden bezeichne N_I die Menge von Individuen, die zur Definition einer ABox zur Verfügung stehen.

Definition 3.7 (ABox) *Eine* Konzeptassertion *ist von der Form $a : C$, wobei C eine Konzeptbeschreibung und $a \in N_I$ ein Individuum sei. Eine* Rollenassertion *ist von der Form $(a, b) : r$, wobei $r \in N_R$ ein Rollenname und $a, b \in N_I$ Individuen seien. Eine* ABox $\mathcal{A}$ *ist eine endliche Menge von Konzept- und Rollenassertionen.*

Wie im Falle von TBoxen spricht man von einer *$\mathcal{L}$-ABox* $\mathcal{A}$, wenn alle Konzeptbeschreibungen in $\mathcal{A}$ aus $\mathcal{L}$ sind. Häufig wird eine ABox $\mathcal{A}$ auch bzgl. einer TBox

$\mathcal{T}$ betrachtet, d.h., dass in den Konzeptassertionen in $\mathcal{A}$ auch definierte Namen aus $\mathcal{T}$ auftreten dürfen. Ist man aber nur an azyklischen TBoxen interessiert, so kann man o.B.d.A. von einer leeren TBox ausgehen, da alle definierten Namen durch ihre aufgefalteten definierenden Konzepte ersetzt werden können (wobei wiederum der mögliche exponentielle Blow-Up beim Auffalten zu beachten ist).

Beispiel 3.8 *Im Familienbeispiel betrachten wir die Individuen* $N_I = \{\mathsf{Hans}, \mathsf{Helmut}, \mathsf{Hanni}\}$. *Die folgende ABox beschreibt (unter Verwendung der definierten Konzeptnamen aus Beispiel 3.5) eine dreiköpfige Familie:*

$$\begin{aligned}\mathcal{A} = \{ \quad & \mathsf{Helmut} : \mathsf{Mann} \sqcap \mathsf{Doktor}, \\ & \mathsf{Hans} : \mathsf{Mann}, \\ & \mathsf{Hanni} : \mathsf{Mensch} \sqcap \neg\mathsf{Männlich}, \\ & (\mathsf{Helmut}, \mathsf{Hanni}) : \mathsf{verheiratet}, \\ & (\mathsf{Hanni}, \mathsf{Hans}) : \mathsf{hatKind}, \\ & (\mathsf{Helmut}, \mathsf{Hans}) : \mathsf{hatKind}\}\end{aligned}$$

Um die Semantik von ABoxen formal zu fassen, ist zunächst der Begriff der Interpretation auf die Menge N_I zu erweitern: eine Interpretation $\mathcal{I} = (\Delta_\mathcal{I}, \cdot^\mathcal{I})$ weist dabei jedem Individuum $a \in N_I$ ein Element $a^\mathcal{I} \in \Delta_\mathcal{I}$ zu. Desweiteren gehen wir in dieser Arbeit (wie in der Literatur üblich) von der *unique name assumption* (UNA) aus, die besagt, dass verschiedene Individuen $a, b \in N_I$ stets verschieden interpretiert werden, d.h. $a^\mathcal{I} \neq b^\mathcal{I}$.

Definition 3.9 (Modell einer ABox) *Eine Interpretation* $\mathcal{I} = (\Delta_\mathcal{I}, \cdot^\mathcal{I})$ *heißt* Modell der ABox $\mathcal{A}$ *genau dann, wenn* $\mathcal{I}$ *jede Assertion in* $\mathcal{A}$ *erfüllt, d.h.* $a^\mathcal{I} \in C^\mathcal{I}$ *für alle Konzeptassertionen* $a : C \in \mathcal{A}$ *und* $(a^\mathcal{I}, b^\mathcal{I}) \in r^\mathcal{I}$ *für alle Rollenassertionen* $(a, b) : r \in \mathcal{A}$ *gilt.*

Es sei noch angemerkt, dass die obige Definition der Semantik einer ABox von der sogenannten *open world assumption* ausgeht (im Gegensatz zu der im klassischen Datenbankbereich gültigen *closed world assumption*), sodass das in einer ABox repräsentierte Wissen als unvollständig zu betrachten ist. Beispielsweise darf man aus der Nicht-Gültigkeit von $a \in_\mathcal{A} C$ nicht $a \in_\mathcal{A} \neg C$ folgern. Dieser Aspekt wirkt sich insbesondere auf Instanzbeziehungen (cf Abschnitt 3.3) und die Charakterisierung und Berechnung des *Most Specific Concepts* (cf Kapitel 7) aus.

3.3 Standardinferenzen

Neben der Repräsentation von Wissen ist ein ebenso wichtiger Aspekt eines BL-Systems das Ziehen von Schlussfolgerungen aus dem explizit in der Wissensbasis gespeicherten Wissen. Hierzu stellen BL-Systeme sogenannte Inferenzdienste zur Verfügung. Zu den bekanntesten und in den vergangenen Jahren umfassend untersuchten Diensten zählen:

Konsistenz: Ist ein Konzept oder eine ABox erfüllbar, d.h. nicht widersprüchlich?

Subsumtion: Ist ein Konzept allgemeiner als ein anderes Konzept?

Instanz: Ist ein Individuum Instanz eines Konzeptes bzgl. einer Wissensbasis?

In dieser Arbeit werden diese Dienste als *Standardinferenzdienste* angesehen, da sie

1. von (fast) allen BL-Systemen bereitgestellt werden,
2. unerlässlich für die Strukturierung einer Wissensbasis sind und
3. ihre Untersuchung seit Einführung der ersten BL-Systeme einen Kernpunkt der Arbeit im Bereich BLen bildet.

Formal sind diese Inferenzprobleme wie folgt definiert.

Definition 3.10 (Konsistenz) *Eine Konzeptbeschreibung C heißt* konsistent (bzgl. der TBox $\mathcal{T}$) *genau dann, wenn eine Interpretation $\mathcal{I}$ (ein Modell $\mathcal{I}$ von $\mathcal{T}$) existiert mit $C^{\mathcal{I}} \neq \emptyset$.*
Eine ABox $\mathcal{A}$ heißt konsistent (bzgl. der TBox $\mathcal{T}$) *genau dann, wenn eine Interpretation $\mathcal{I}$ existiert, die Modell von $\mathcal{A}$ (und $\mathcal{T}$) ist.*

Inferenzdienste, die entscheiden, ob ein Konzept oder eine ABox konsistent ist, unterstützen den Anwender bei der Definition der Wissensbasis, da sie Widersprüche in Konzepten und ABoxen aufdecken. Liegt dann eine konsistente Wissensbasis vor, so erlauben es die Dienste Subsumtion und Instanz, implizites Wissen über die definierten Konzepte und Individuen in Form von Subsumtionsbeziehungen zwischen Konzepten und Instanzbeziehungen zwischen Individuen und Konzepten zu folgern.

Definition 3.11 (Subsumtion und Äquivalenz) *Die Konzeptbeschreibung D* subsumiert *die Konzeptbeschreibung C (bzgl. der TBox $\mathcal{T}$), kurz $C \sqsubseteq D$ ($C \sqsubseteq_{\mathcal{T}} D$) genau dann, wenn $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ für alle Interpretationen $\mathcal{I}$ (für alle Modelle $\mathcal{I}$ von $\mathcal{T}$). C ist* äquivalent *zu D (bzgl. der TBox $\mathcal{T}$), kurz $C \equiv D$ ($C \equiv_{\mathcal{T}} D$), genau dann, wenn $C \sqsubseteq D$ und $D \sqsubseteq C$ ($C \sqsubseteq_{\mathcal{T}} D$ und $D \sqsubseteq_{\mathcal{T}} C$).*

Die atomaren und definierten Konzeptnamen aus Beispiel 3.5 liefern die in Abbildung 3.2 dargestellte Subsumtionshierarchie, wobei die allgemeineren Konzepte jeweils über den spezielleren Konzepten stehen.

Definition 3.12 (Instanz) *Ein Individuum $a \in N_I$ heißt* Instanz *der Konzeptbeschreibung C bzgl. der ABox $\mathcal{A}$ (und der TBox $\mathcal{T}$), kurz $a \in_{\mathcal{A}} C$ ($a \in_{\mathcal{T},\mathcal{A}} C$), genau dann, wenn $a^{\mathcal{I}} \in C^{\mathcal{I}}$ für alle Modelle $\mathcal{I}$ von $\mathcal{A}$ (und $\mathcal{T}$).*

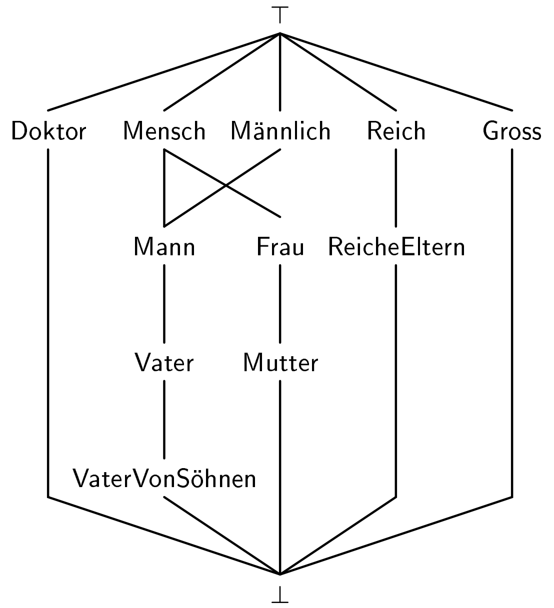


Abbildung 3.2: Die Subsumtionshierarchie der Konzeptnamen aus Beispiel 3.5.

Bezüglich der TBox $\mathcal{T}$ aus Beispiel 3.5 und der ABox $\mathcal{A}$ aus Beispiel 3.8 ist das Individuum Hanni Instanz von Frau und Mutter. Helmut ist Instanz von Vater, nicht aber von VaterVonSöhnen, da ja nicht von der CWA ausgegangen wird, Helmut also weitere Kinder und damit insbesondere Töchter haben könnte.

Wir werden in Kapitel 4 noch ausführlich darauf eingehen, inwiefern diese Standardinferenzdienste in Anwendungen, insbesondere in der betrachteten Anwendung in der Prozesstechnik, beim Aufbau und bei der Wartung einer Wissensbasis eingesetzt werden können.

Es ist aus der Literatur wohlbekannt, dass enge Zusammenhänge zwischen den verschiedenen Standardinferenzen bestehen. Sie sind im folgenden Satz zusammengefasst.

Satz 3.13 *Seien C, D Konzeptbeschreibungen, $\mathcal{T}$ eine TBox, $\mathcal{A}$ eine ABox und a ein Individuum. Dann gilt*

1. *$C \sqsubseteq D$ ($C \sqsubseteq_{\mathcal{T}} D$) genau dann, wenn $C \sqcap \neg D$ (bzgl. $\mathcal{T}$) inkonsistent ist;*

2. *C ist (in)konsistent (bzgl. $\mathcal{T}$) genau dann, wenn die ABox $\{a : C\}$ (in)konsistent (bzgl. $\mathcal{T}$) ist und das ist genau dann der Fall, wenn $C \sqsubseteq \bot$ ($C \sqsubseteq_{\mathcal{T}} \bot$);*

3. *a ist Instanz von C bzgl. $\mathcal{A}$ (und $\mathcal{T}$) genau dann, wenn $\mathcal{A} \cup \{a : \neg C\}$ inkonsistent (bzgl. $\mathcal{T}$) ist.*

Erlaubt also die betrachtete BL Konzeptkonjunktion und volle Negation, so können alle oben genannten Inferenzprobleme auf das Konsistenzproblem für ABoxen reduziert werden. Subsumtion lässt sich in solchen BLen auf Konsistenz von Konzeptbeschreibungen reduzieren. Um Konsistenz von Konzepten in $\mathcal{ALC}$ bzw. Erweiterungen von $\mathcal{ALC}$ zu entscheiden, haben sich sogenannte Tableau-Algorithmen als besonders erfolgreich herausgestellt [SS91, HB91, BH91a, Baa91, HST99].

Demgegenüber basiert das bei AT&T entwickelte System Classic auf einem strukturellen Subsumtionsalgorithmus [BP94]. Da die in Classic verwendete BL nicht unter Komplement abgeschlossen ist, kann Subsumtion nicht auf Konsistenz reduziert werden. Die Idee hinter strukturellen Algorithmen besteht darin, Subsumtion $C \sqsubseteq D$ durch einen syntaktischen Vergleich geeigneter Normalformen von C und D zu entscheiden. Es hat sich gezeigt, dass derartige Charakterisierungen der Subsumtion einen sehr guten Ausgangspunkt für Charakterisierung und Berechnung von Matchern, Least Common Subsumern und Most Specific Concepts bilden [BK00a, BKBM99, BK98, BKM99a, Küs00]. In Kapitel 5 wird eine entsprechende Charakterisierung der Subsumtion für die betrachteten BLen mit Existenzrestriktionen gegeben, die den Ausgangspunkt für strukturelle Subsumtionsalgorithmen bilden kann. In der Anwendung in der Prozesstechnik haben wir aber auf die Implementierung eines strukturellen Subsumtionsalgorithmus für $\mathcal{ALE}$ verzichtet und greifen statt dessen auf das sehr effiziente BL-System FaCT zurück, dem ein Tableaux-Algorithmus für eine Erweiterung von $\mathcal{ALC}$ zugrunde liegt [Hor98].

3.4 Nicht-Standardinferenzen

Beim Einsatz des BL-Systems Classic in verschiedenen Anwendungen [WWV$^+$93, MW98b, MW98a, MRI95] sowie dem Einsatz der BL-Systeme Crack bzw. FaCT in unserer Anwendung in der Prozesstechnik [BS96b, Sat98] hat sich gezeigt, dass für eine umfassende Unterstützung des Aufbaus und der Wartung großer Wissensbasen weitere *Nicht-Standardinferenzdienste* nötig sind [MPS98]. Hierzu zählen *Unifikation und Matching von Konzeptbeschreibungen*, die Berechnung von *Most Specific Concept* und *Least Common Subsumer* und das *Rewriting von Konzeptbeschreibungen*. Für diese Dienste werden in BL-Systemen derzeit (wenn überhaupt) lediglich ad-hoc Implementierungen zur Verfügung gestellt, ohne dass die zugrundeliegenden Inferenzprobleme zuvor formal näher untersucht worden sind.

Im folgenden wird nun zunächst das Problem Unifikation und Matching von Konzeptbeschreibungen kurz dargestellt, wobei auf formale Definitionen verzichtet wird, da diese Probleme hier nicht näher betrachtet werden. Danach werden die Begriffe Most Specific Concept (MSC) und Least Common Subsumer (LCS) eingeführt und verschiedene Anwendungen skizziert. Als letzte Nicht-Standardinferenz wird das Rewriting von Konzeptbeschreibungen betrachtet: es wird zunächst ein allgmeiner Rahmen für dieses Problem definiert und anschließend werden neben dem insbesondere durch die Anwendung in der Prozesstechnik motivierten *minimalen Rewriting-Problem* als weitere interessante Instanzen noch das *Rewriting von Datenbankanfragen mit Views* sowie die *Übersetzung von Wissensbasen* eingeführt.

3.4.1 Unifikation und Matching von Konzeptbeschreibungen

Die Idee hinter diesen Inferenzproblemen ist, dass man gewisse Konzeptnamen in Konzeptbeschreibungen als Variablen auffasst, die vor einem Test auf Subsumtion oder Äquivalenz geeignet durch Konzeptbeschreibungen ersetzt werden dürfen. Von Unifikation spricht man, wenn beide Beschreibungen Variablen enthalten dürfen und von Matching, wenn nur eine Beschreibung Variablen enthalten darf. In [BN98] wurde das Unifikationsproblem in $\mathcal{FL}_0$ gelöst, wobei der dort gewählte Ansatz bisher nicht auf Erweiterungen von $\mathcal{FL}_0$ übertragen werden konnte, d.h. Unifikation in ausdrucksstärkeren BLen ist ein offenes Problem. Das Matchingproblem wurde erstmalig in [BM96] formal spezifiziert und inzwischen für die BLen $\mathcal{ALE}$, $\mathcal{ALN}$ und $\mathsf{Classic}$ umfassend untersucht [BKBM99, BK00a, Küs00]. Als Anwendungen für das Matching seien hier zum einen das *Pruning von Konzeptbeschreibungen* [MPS98, MIP$^+$98, MRI95] sowie die *Integration von Wissensbasen* [BK00b] genannt: Beim Pruning von Konzeptbeschreibungen ist das Ziel, unwesentliche Teile aus sehr großen Konzeptbeschreibungen (die meist mehrere Seiten füllen und damit für den Anwender zunächst unlesbar sind) herauszuschneiden und somit das Herausfiltern relevanter Aspekte aus diesen Konzepten zu unterstützen. Bei der Integration von (großen) Wissensbasen tritt das Problem auf, dass intuitiv gleiche Konzepte in verschiedenen Wissensbasen verschiedene Namen besitzen können oder auf unterschiedlichen Granularitätsstufen beschrieben werden. Hier können Matching (und Unifikation) helfen, mögliche Namenskonflikte aufzudecken.

3.4.2 Most Specific Concept und Least Common Subsumer

Die Betrachtung von MSC und LCS ist durch verschiedene Anwendungen motiviert, insbesondere natürlich durch das in Kapitel 4 ausführlich beschriebene Anwendungsszenario für den Einsatz von BL-Systemen in der Prozesstechnik. Bevor auf die verschiedenen Anwendungen eingegangen wird, werden MSC und LCS formal eingeführt.

Definition 3.14 (Most Specific Concept) *Sei C eine $\mathcal{L}$-Konzeptbeschreibung, $\mathcal{A}$ eine $\mathcal{L}$-ABox und a ein Individuum in $\mathcal{A}$. Die $\mathcal{L}$-Konzeptbeschreibung C ist* Most Specific Concept von a bzgl. $\mathcal{A}$ *(kurz: $C = \mathsf{msc}_{\mathcal{A}}(a)$) genau dann, wenn*

1. *$a \in_{\mathcal{A}} C$ und*
2. *C die speziellste $\mathcal{L}$-Konzeptbeschreibung mit dieser Eigenschaft ist, d.h. für jede $\mathcal{L}$-Konzeptbeschreibung C' mit $a \in_{\mathcal{A}} C'$ gilt $C \sqsubseteq C'$.*

Ist aus dem Zusammenhang klar, bezüglich welcher ABox das MSC bestimmt wird, so fällt der Index $\mathcal{A}$ weg und man schreibt nur $\mathsf{msc}(a)$.

Der Begriff MSC erlaubt es, von konkreten Individuen in der Wissensbasis zu abstrahieren, dabei aber soviel Informationen wie möglich, d.h. wie in der zugrundeliegenden BL ausdrückbar, über dieses Individuum zu berücksichtigen. Die Berechnung des MSC steht also für den Schritt vom konkreten Individuum hin zu einer abstrakten Konzeptbeschreibung. Nebel [Neb90a] hat den Begriff MSC erstmalig eingeführt im Zusammenhang mit der Reduktion des Instanzproblems auf das Subsumtionsproblem: die Instanzbeziehung $a \in_{\mathcal{A}} C$ gilt genau dann, wenn die Subsumtionsbeziehung $\mathsf{msc}_{\mathcal{A}}(a) \sqsubseteq C$ gilt. Ein solcher Reduktionsschritt setzt natürlich voraus, dass das MSC in der betrachteten BL stets existiert. Aus den Ergebnissen in [BK98, Küs00] folgt, dass diese Voraussetzung zwar für $\mathcal{FL}_0$ erfüllt ist. Allerdings ist sie weder in BLen mit Existenzrestriktionen (cf Kapitel 7) noch in $\mathcal{ALN}$ erfüllt [BK98]. Im Falle von $\mathcal{ALN}$ lässt sich das MSC exakt repräsentieren, wenn man zyklische Konzeptbeschreibungen, d.h. durch zyklische TBoxen definierte Konzeptnamen, zulässt [BK98], wohingegen für $\mathcal{ALE}$ unklar ist, ob und wie das Problem mit zyklischen Konzeptbeschreibungen gelöst werden kann. Kapitel 7 beschäftigt sich daher im wesentlichen mit der Approximation des MSC in BLen mit Existenzrestriktionen und diskutiert das offene Problem der exakten Repräsentation des MSC in diesen BLen nur knapp.

Definition 3.15 (Least Common Subsumer) *Seien C und $C_1, \ldots, C_n$ Konzeptbeschreibungen einer BL $\mathcal{L}$. Die $\mathcal{L}$-Konzeptbeschreibung C ist* Least Common Subsumer *von $C_1, \ldots, C_n$ (kurz: $C = \mathsf{lcs}(C_1, \ldots, C_n)$) genau dann wenn*

1. *$C_i \sqsubseteq C$ für alle $1 \leq i \leq n$ und*
2. *C die speziellste Konzeptbeschreibung mit dieser Eigenschaft ist, d.h. für jede $\mathcal{L}$-Konzeptbeschreibung C' mit $C_i \sqsubseteq C'$ für alle $1 \leq i \leq n$ gilt $C \sqsubseteq C'$.*

Die Idee hinter dem Begriff LCS besteht darin, die Gemeinsamkeiten aus Konzeptbeschreibungen herauszuziehen. Im Sinne einer Ober-/Unterkonzeptbeziehung zwischen Ergebnis und Ausgangskonzepten bildet somit – analog zur Berechnung des MSC – auch die Berechnung des LCS zu einer Menge von Konzeptbeschreibungen einen Abstraktionsschritt.

Der LCS zu $\mathcal{L}$-Konzeptbeschreibungen $C_1, \ldots, C_n$ ist nach Definition eindeutig modulo Äquivalenz, muss aber nicht für alle BLen $\mathcal{L}$ existieren. Die Existenz des LCS zu $C_1, \ldots, C_n$ hängt unmittelbar mit dem Begriff einer *minimalen vollständigen Menge minimaler Common Subsumer von* $C_1, \ldots, C_n$ zusammen. Eine $\mathcal{L}$-Konzeptbeschreibung C heißt *Common Subsumer* von $C_1, \ldots, C_n$, wenn $C_i \sqsubseteq C$ für alle $1 \leq i \leq n$. Eine Menge M von (bzgl. Subsumtion) minimalen Common Subsumern von $C_1, \ldots, C_n$ heißt *minimal vollständig*, wenn für alle Common Subsumer C von $C_1, \ldots, C_n$ ein

$D \in \mathcal{M}$ existiert mit $D \sqsubseteq C$ und M eine (bzgl. $|\cdot|$) minimale Menge mit dieser Eigenschaft ist.

Der LCS von $C_1, \ldots, C_n$ existiert genau dann, wenn eine minimal vollständige Menge minimaler Common Subsumer von $C_1, \ldots, C_n$ der Mächtigkeit 1 existiert. Tatsächlich können in BLen, die Konjunktion zulassen, nur zwei Fälle auftreten: entweder die Menge hat Mächtigkeit 1 oder sie ist leer. Wäre die Mächtigkeit der Menge größer 1 aber endlich, so würde die Konjunktion der angeblich minimalen aber paarweise unvergleichbaren minimalen Common Subsumer einen spezielleren Common Subsumer liefern im Widerspruch zur Minimalität der Elemente der Menge. Wäre die Menge unendlich, so würden die Konjunktionen der angeblich minimalen Common Subsumer eine unendlich absteigende Kette spezieller werdender Common Subsumer bilden wiederum im Widerspruch zur Minimalität der Elemente der Menge. Ein einfaches Beispiel für eine BL, in der der LCS nicht stets existiert, ist die BL, die nur atomare Konzeptnamen und Konjunktion erlaubt: hier existiert der LCS zweier verschiedener Konzeptnamen nicht. Es gibt aber auch ausdrucksstärkere Subsprachen von Classic, in denen der LCS im allgemeinen nicht existiert [Küs00]. Dass der LCS von $C_1, \ldots, C_n$ in $\mathcal{L}$ stets existiert, gilt trivialerweise in BLen $\mathcal{L}$, in denen Disjunktion ausgedrückt werden kann, da dann die Disjunktion $C_1 \sqcup \ldots \sqcup C_n$ genau den LCS von $C_1, \ldots, C_n$ liefert. Er existiert aber auch stets in der BL $\mathcal{ALN}$ [Küs00] und—wie in Kapitel 5 dieser Arbeit gezeigt wird—in den BLen $\mathcal{EL}$, $\mathcal{FLE}$ und $\mathcal{ALE}$.

Formal eingeführt wurde der LCS erstmalig in [CBH92]. Bereits in dieser Arbeit wurde der enge Zusammenhang zwischen der strukturellen Subsumtion und der Berechnung des LCS herausgearbeitet, obwohl sich die in [CBH92] für die Berechnung des LCS in Classic entwickelten Algorithmen als teilweise inkorrekt herausgestellt haben [KB99]. Inzwischen ist aber Existenz und Komplexität des LCS in Classic (und relevanten Teilsprachen) umfassend untersucht und es wurden korrekte Algorithmen vorgestellt [KB99, Küs00].

Die motivierende Anwendung für die Betrachtung von MSC und LCS in Classic [CBH92, CH94b, FP96] war das *Lernen von Konzeptbeschreibungen*, welches im folgenden Absatz vorgestellt wird. Im Rahmen der vorliegenden Arbeit spielen MSC und LCS eine zentrale Rolle bei der *Bottom-Up Konstruktion von Wissensbasen*. Da dieses Anwendungsszenario in Kapitel 4 noch ausführlich beschrieben wird, wird es hier nur skizziert und gegen den durch Standardinferenzen unterstützten Top-Down-Ansatz zur Konstruktion von Wissensbasen sowie das Lernen von Konzeptbeschreibungen abgegrenzt.

Lernen von Konzeptbeschreibungen. Erste Lösungsansätze für dieses Problem wurden (wie beim Matching) für das System Classic bzw. geeignete Teilsprachen der zugrundeliegenden BL betrachtet. Die Idee ist jeweils, ausgehend von Mengen von positiven und negativen Beispielen ein Zielkonzept iterativ zu lernen. Um den Begriff des Lernens formal zu fassen wurde dabei jeweils auf bekannte Lernmodelle zurückgegriffen. So wurde beispielsweise gezeigt, dass im Sinne des von Valiant eingeführten Modells der *PAC-Lernbarkeit* (PAC für *Probably Approximately Correct*) [Val84] Classic-Konzeptbeschreibungen nicht lernbar sind, wohl aber $\mathcal{ALN}$-Konzeptbeschreibungen [CH94b]. In [FP96] wurde dann später nachgewiesen, dass Classic im Rahmen eines erweiterten PAC-Lernmodells lernbar ist.

Um die im nächsten Absatz und ausführlich in Kapitel 4 vorgestellte Bottom-Up-Konstruktion von Wissensbasen (bei der man schlagwortartig ebenfalls vom *Lernen von Konzepten aus Beispielen* sprechen kann) gegen diesen Lernbegriff abzugrenzen, soll nun noch kurz auf das PAC-Lernmodell eingegangen werden.

Das Lernmodell lässt sich grob wie folgt beschreiben: Ausgehend von einer Menge von positiven und einer Menge von negativen Beispielen wird ein Zielkonzept iterativ gelernt, wobei ein solches Konzept *mit hoher Wahrscheinlichkeit* gefunden werden soll und es außerdem *annähernd korrekt* ist, d.h. die Ausgangsbeispiele sollen bzgl. des Zielkonzeptes bis auf einen kleinen Fehler korrekt klassifiziert werden. Dabei ist jeder Iterationsschritt in polynomieller Zeit auszuführen. Ein wesentlicher Bestandteil des Modells ist ein Lehrer, der insbesondere die Iteration steuert: in jedem Schritt fragt das Verfahren den Lehrer (dies kann ein menschlicher Benutzer oder auch ein Programm sein), ob das Zielkonzept erreicht ist. Wenn „ja", ist man fertig, wenn „nein" wird ein weiteres Beispiel geliefert, welches zur Verfeinerung der Hypothese im nächsten Schritt herangezogen wird. Kernpunkt des diesem Modell genügenden Lernverfahrens für (Teilsprachen von) Classic sind nun die Operation MSC und LCS [CH94b, CH94a, FP96]. Mit Hilfe des MSC abstrahiert man von den Beispielen (die als ABox-Individuen präsentiert werden) und durch den LCS werden Gemeinsamkeiten dieser MSCs extrahiert, die jeweils in die neue Hypothese einfließen. Es sei an dieser Stelle nochmals darauf hingewiesen, dass bei der praktischen Realisierung das MSC jeweils nur approximiert wurde, da zu diesem Zeitpunkt noch keine exakte Charakterisierung des MSC in (Teilsprachen von) Classic bekannt war [CH94b]. Inzwischen wurde gezeigt [BK98, Küs00], dass das MSC in $\mathcal{ALN}$ (der Kernsprache von Classic) auch nur unter Verwendung zyklischer Konzeptbeschreibungen exakt repräsentiert werden kann. Mit den in Classic zur Verfügung stehenden ad-hoc Implementierungen zum LCS können solche zyklischen Konzepte aber auch nicht weiterverarbeitet werden.

Unterstützung der Bottom-Up-Konstruktion von Wissensbasen. Traditionell werden terminologische Wissensbasen „top-down" aufgebaut, indem zunächst die relevanten Konzepte des Anwendungsbereichs in der TBox definiert werden. In einem zweiten Schritt werden dann Eigenschaften konkreter Objekte mit Hilfe dieser definierten Konzepte in einer ABox spezifiziert. Die klassischen Inferenzdienste Konsistenz-, Subsumtions- und Instanztest unterstützen dieses Vorgehen auf natürliche Weise: Konzeptbeschreibungen und ABox können auf Konsistenz getestet werden und die TBox kann in Form der Subsumtionshierarchie der definierten Konzepte strukturiert werden. Schließlich liefert der Instanztest genau die durch die Beschreibungen der Individuen und Konzepte implizierten Instanzbeziehungen.

Allerdings ist dieser traditionelle Ansatz nicht immer geeignet. Häufig ist bei der Modellierung eines Anwendungsbereichs nicht von Anfang an klar, was die relevanten und interessanten Konzepte des Bereichs sind. Desweiteren ist es – nachdem man sich entschieden hat, welche Begriffe modelliert werden sollen – nicht einfach, diese Begriffe formal in einer BL zu definieren. Tatsächlich hat sich bei der Anwendung in der Prozesstechnik herausgestellt, dass die Modellierer es vorziehen, zunächst typische Beispiele für Instanzen eines zu definierenden Konzeptes einzuführen, um diese dann in einem zweiten Schritt zu einer Konzeptdefinition zu generalisieren. Um diesen zweiten Schritt (zumindest teilweise) zu automatisieren, eignen sich Verfahren zur Berechnung von MSC und LCS: die typischen Beispiele werden durch ABox-Individuen repräsentiert. Zu jedem dieser Individuen wird dann das MSC bestimmt und zu den so erhaltenen Konzeptbeschreibungen dann der LCS. Dies liefert per Konstruktion das speziellste Konzept, das alle vom Modellierer angegebenen Beispiele als Instanzen hat.

Dieses Anwendungsszenario unterscheidet sich im wesentlichen durch die folgenden zwei Punkte vom oben beschriebenen Lernen von Konzepten:

- Es gibt kein Zielkonzept, das durch ein induktives Verfahren approximiert werden soll. Vielmehr wird ein eindeutig spezifiziertes Konzept durch klar definierte, deduktive Schritte berechnet.

- Bei der Bottom-Up-Konstruktion gibt es keine negativen Beispiele und keinen Lehrer, der in den Berechnungsprozess eingreifen kann. Vielmehr bekommt der Modellierer das zu den Beispielen berechnete Konzept zur Inspektion vorgeschlagen. Sollte das Ergebnis für den Modellierer unbrauchbar sein (zu speziell, zu allgemein, nicht die gewünschten Aspekte erfassen), so bleibt es ihm überlassen, die Beispiele zu revidieren und für eine neue Menge von Beispielen einen neuen Vorschlag berechnen zu lassen oder das gelieferte Konzept per Hand zu modifizieren.

Gemeinsam haben beide Szenarien lediglich, dass sie durch Angabe von (Mengen von) Beispielen vom Benutzer initiiert werden und die Berechnung von MSC und LCS jeweils eine zentrale Rolle spielt.

3.4.3 Rewriting von Konzeptbeschreibungen

Im Unterschied zu Standardinferenzdiensten, die als Entscheidungsverfahren nur „ja“ oder „nein“ ausgeben, liefern Verfahren für die oben eingeführten Nicht-Standardinferenzen Konzeptbeschreibungen oder Zuweisungen von Konzeptbeschreibungen an Variablen zurück. Diese so erhaltenen Konzeptbeschreibungen werden in der Regel nicht vollautomatisch weiterverarbeitet (also z.B. beim Lernen von Konzepten unbesehen in die Wissensbasis eingeordnet), sondern dem Anwender zur Inspektion angezeigt. Daher ist es notwendig, dass sie möglichst klein und damit gut lesbar sind. Leider erfüllen die bisher bekannten Verfahren für die Nicht-Standardinferenzen diese Bedingung nicht. Wie wir insbesondere in den Kapiteln 5 und 7 sehen werden, liegt dies daran, dass die Verfahren auf aufgefalteten Konzeptbeschreibungen arbeiten und auch aufgefaltete Konzeptbeschreibungen zurückliefern. Tatsächlich füllen die in der Prozesstechnikanwendung erhaltenen LCS häufig mehrere Druckseiten (sie umfassen mehrere hundert Vorkommen atomarer Konzept- und Rollennamen). Das *Rewriting von Konzeptbeschreibungen bzgl. einer gegebenen TBox* zielt nun darauf ab, diese Konzeptbeschreibungen in äquivalente kleinere Beschreibungen zu transformieren, indem Teile der Beschreibung jeweils durch definierte Namen aus einer zugrundeliegenden TBox ersetzt werden.

In der Literatur findet sich keine formale Definition, die einen allgemeinen Rahmen für das Rewriting von Konzeptbeschreibungen in BLen liefern würde. Daher wird ein solcher Rahmen zunächst einmal in der folgenden Definition festgelegt. Die für die Prozesstechnikanwendung relevante Instanz, das sogenannte minimale Rewriting-Problem, wird im nachfolgenden Unterabschnitt eingeführt. Den Abschluss des Abschnitts bilden zwei weitere interessante und in der Literatur bereits teilweise untersuchte Rewriting-Probleme, die sich ebenfalls in den allgemeinen Rahmen einbetten lassen.

Definition 3.16 *Sei N_R eine Menge von Rollennamen und N_P eine Menge primitiver Konzeptnamen. Desweiteren seien $\mathcal{L}_s$, $\mathcal{L}_d$ und $\mathcal{L}_t$ drei BLen (s für* source, *d für* destination *und t für* TBox*). Ein* Rewriting-Problem *ist gegeben durch*

- *eine $\mathcal{L}_t$-TBox $\mathcal{T}$, die nur Rollennamen aus N_R und primitive Konzeptnamen aus N_P enthält; die Menge der definierten Namen aus $\mathcal{T}$ wird mit N_D bezeichnet;*
- *eine $\mathcal{L}_s$-Konzeptbeschreibung C, die nur Namen aus N_R und N_P enthält;*
- *einer binären Relation $\rho \subseteq \mathcal{L}_s \times \mathcal{L}_d$ zwischen $\mathcal{L}_s$- und $\mathcal{L}_d$-Konzeptbeschreibungen.*

*Ein $\mathcal{L}_d$-*Rewriting von C bzgl. $\mathcal{T}$ *ist eine $\mathcal{L}_d$-Konzeptbeschreibung D, die mit Namen aus N_R und $N_P \cup N_D$ aufgebaut ist und für die $C \rho D$ gilt.*

Für eine geeignete Ordnung $\preceq$ auf $\mathcal{L}_d$-Konzeptbeschreibungen heißt ein Rewriting D $\preceq$-minimal *genau dann, wenn kein Rewriting D' existiert mit $D' \prec D$.*

Das minimale Rewriting-Problem

Die für unsere Anwendung in der Prozesstechnik wichtigste Instanz des allgemeinen Rahmens ist das sogenannte *minimale Rewriting-Problem*, d.h. die Instanz bei der

- alle drei BLen $\mathcal{L}_s = \mathcal{L}_t = \mathcal{L}_d$ die gleiche BL $\mathcal{L}$ sind,
- als binäre Relation ρ Äquivalenz modulo der TBox betrachtet wird und
- die $\mathcal{L}$-Konzeptbeschreibungen nach ihrer Größe geordnet sind, d.h. $D \preceq D'$ genau dann, wenn $|D| \leq |D'|$.

Tatsächlich hat die Definition der Größe einer Konzeptbeschreibung einen entscheidenden Einfluss auf die Formulierung der Resultate. Wir stützen uns insbesondere bei den technischen Resultaten in Kapitel 8 zur Berechenbarkeit minimaler Rewritings in $\mathcal{ALE}$ auf das durch die folgende Definition gegebene Größenmaß.

Definition 3.17 *Sei C eine Konzeptbeschreibung, die nur Konstruktoren aus Tabelle 3.1 enthält. Die* Größe $|C|$ *von C ist induktiv definiert durch*

$$\begin{array}{rcl@{\qquad\qquad}rcl}
|\top| & := & 0, & |C \sqcap D| & := & |C| + |D|, \\
|\bot| & := & 0, & |C \sqcup D| & := & |C| + |D|, \\
|P| & := & 1, & |\neg C| & := & |C|, \\
|(\leq n\ r)| & := & n+1, & |\exists r.C| & := & 1 + |C|, \\
|(\geq n\ r)| & := & n+1, & |\forall r.C| & := & 1 + |C|,
\end{array}$$

wobei $P \in N_C$ ein Konzeptname, $r \in N_R$ ein Rollenname und $n \in \mathbb{N}$ eine natürliche Zahl sei.

Intuitiv betrachten wir als die Größe einer Konzeptbeschreibung die Anzahl der Vorkommen von Konzept- und Rollennamen; insbesondere werden also die Vorkommen der Konstruktoren *nicht* gezählt.
Als einfaches Beispiel für das Rewriting betrachte man die $\mathcal{ALE}$-Konzeptbeschreibung

$$\begin{array}{rcl}
C & = & \mathsf{Mensch} \sqcap \mathsf{M\ddot{a}nnlich} \sqcap \mathsf{Reich} \sqcap \exists \mathsf{hatKind}.\mathsf{Mensch} \sqcap \\
 & & \forall \mathsf{hatKind}.(\mathsf{Mensch} \sqcap \mathsf{M\ddot{a}nnlich} \sqcap \mathsf{Reich})
\end{array}$$

und die $\mathcal{ALE}$-TBox $\mathcal{T}$ aus Beispiel 3.5. Man sieht leicht, dass die Konzeptbeschreibungen

$$\begin{array}{l}
\mathsf{VaterVonS\ddot{o}hnen} \sqcap \mathsf{ReicheEltern}, \\
\mathsf{VaterVonS\ddot{o}hnen} \sqcap \mathsf{Reich} \sqcap \forall \mathsf{hatKind}.\mathsf{Reich}, \\
\mathsf{Vater} \sqcap \mathsf{ReicheEltern} \sqcap \forall \mathsf{hatKind}.(\mathsf{Mensch} \sqcap \mathsf{M\ddot{a}nnlich})
\end{array}$$

$\mathcal{ALE}$-Rewritings von C bzgl. $\mathcal{T}$ sind, wobei die erste Konzeptbeschreibung ein minimales Rewriting von C bzgl. $\mathcal{T}$ ist.
Ein Spezialfall des minimalen Rewriting-Problems ergibt sich, wenn man eine leere TBox betrachtet, d.h. wenn man an einer minimalen Repräsentation einer gegebenen

Konzeptbeschreibung in der zur Verfügung stehenden BL interessiert ist. Derartige minimale Repräsentanten spielen z.B. beim Matching von Konzeptbeschreibungen eine wichtige Rolle [BK00a, Küs00]. In [Küs00] wird gezeigt, dass solche minimalen Repräsentanten in $\mathcal{ALE}$ in polynomieller Zeit mit Orakel für Subsumtion in $\mathcal{ALE}$ und in $\mathcal{ALN}$ in polynomieller Zeit berechnet werden können. Diese Ergebnisse erhält man auch für den Spezialfall $\mathcal{T} = \emptyset$ aus den Komplexitätsresultaten in Abschnitt 8.1.

Rewriting von Datenbankanfragen mit Views

Hierbei handelt es sich um ein sehr bekanntes Rewriting-Problem, das im Datenbankbereich, insbesondere bei der Optimierung von Datenbankanfragen, eine wichtige Rolle spielt und dort bereits umfassend untersucht wurde. Ein recht enger Zusammenhang zum oben eingeführten Rewriting in BLen ergibt sich für Arbeiten, die das Problem für Datenbankanfragen und Views betrachten, die durch BL-Konzeptbeschreibungen repräsentiert werden. In [BLR97] werden Views betrachtet, die als Konzeptdefinitionen in einer $\mathcal{ALCNR}$- bzw. $\mathcal{ALN}$-TBox $\mathcal{V}$ gegeben sind und in [GR00a] werden die BLen $\mathcal{ALE}$ und $\mathcal{ALEN}$ betrachtet.[2] Ziel ist es jeweils, Anfragen unter Verwendung von Views möglichst gut zu approximieren, d.h. eine Anfrage q (gegeben als $\mathcal{ALCNR}$-/$\mathcal{ALN}$- bzw. $\mathcal{ALE}$-/$\mathcal{ALEN}$-Konzeptbeschreibung) soll durch eine speziellere Konzeptbeschreibung, die nur definierte Namen (Views aus $\mathcal{V}$) und eine möglicherweise eingeschränkte Menge von Konstruktoren enthält, approximiert werden.

Dieses Problem lässt sich wie folgt in unseren allgemeinen Rahmen einbetten: Als BLen betrachtet man in [BLR97] $\mathcal{L}_s = \mathcal{L}_t = \mathcal{ALN}/\mathcal{ALCNR}$ und als Zielsprache $\mathcal{L}_d = \{\sqcap, \sqcup\}$ und in [GR00a, GR00b] $\mathcal{L}_s = \mathcal{L}_t = \mathcal{L}_d = \mathcal{FLE}/\mathcal{ALE}/\mathcal{ALEN}$. Die zu berechnenden Rewritings sollen maximal sein aber von der Anfrage subsumiert werden, d.h. als Relation zwischen Eingabekonzept C und Rewriting D betrachtet man die umgekehrte Subsumtionsbeziehung $C \sqsupseteq D$ und als Ordnung $\preceq$ ebenfalls die umgekehrte Subsumtionsbeziehung $\sqsupseteq$. Interessiert man sich (wie in allen genannten Arbeiten) für *totale* Rewritings, so lässt sich dies durch folgende Spezialisierung von $\preceq$ ausdrücken: $D \preceq D'$ genau dann, wenn (a) D keine primitiven Konzeptnamen enthält, D' aber wohl; oder (b) D und D' keine primitiven Konzeptnamen enthalten und $D \sqsupseteq D'$. Existiert ein totales Rewriting von C bzgl. $\mathcal{V}$, so ist jedes bzgl. der modifizierten Ordnung $\preceq$ minimale Rewriting von C total.

Aus den oben genannten Arbeiten ergeben sich die folgenden Resultate:

- Aus den Ergebnissen in [BLR97] folgt, dass für $\mathcal{L}_s = \mathcal{L}_t = \mathcal{ALCNR}$ bzw. $\mathcal{L}_s = \mathcal{L}_t = \mathcal{ALN}$ und $\mathcal{L}_d = \{\sqcap, \sqcup\}$ ein maximales spezielleres, totales Rewriting D von C bzgl. $\mathcal{V}$ in exponentieller Zeit berechenbar ist (falls eines existiert). Für

[2] $\mathcal{ALCNR}$ erlaubt neben den Konstruktoren aus $\mathcal{ALC}$ auch Zahlenrestriktionen und *Rollenkonjunktion* $r_1 \sqcap r_2$; mit $\mathcal{ALEN}$ wird die Erweiterung von $\mathcal{ALE}$ um Zahlenrestriktionen bezeichnet.

$\mathcal{ALCNR}$ bzw. $\mathcal{ALN}$ kann, ebenfalls in exponentieller Zeit, entschieden werden, ob ein totales Rewriting äquivalent zu C modulo $\mathcal{V}$ existiert.

- Aus den Ergebnissen in [GR00a] bzw. [GR00b] folgt, dass für $\mathcal{L}_s = \mathcal{L}_t = \mathcal{L}_d = \mathcal{FLE}$ bzw. $\mathcal{L}_s = \mathcal{L}_t = \mathcal{L}_d = \mathcal{ALE}$ die Menge der totalen, maximal spezielleren Rewritings bis auf Äquivalenz endlich ist und in exponentieller Zeit berechnet werden kann. Für $\mathcal{L}_s = \mathcal{L}_t = \mathcal{L}_d = \mathcal{ALEN}$ wird in [GR00a] gezeigt, dass es unendlich viele paarweise unvergleichbare, totale, maximal speziellere Rewritings geben kann.

Es sei an dieser Stelle noch angemerkt, dass in der Zusammenfassung in [BLR97] behauptet wird, dass in $\mathcal{ALN}$ ein maximales spezielleres, totales Rewriting in *polynomieller* Zeit berechnet werden kann. Tatsächlich liefert aber Theorem 3.2 des Artikels lediglich eine exponentielle obere Schranke. Diese Schranke deckt sich mit den Komplexitätsresultaten für das Entscheidungsproblem zum minimalen Rewriting in $\mathcal{ALN}$ aus Kapitel 8 dieser Arbeit.

Übersetzung von Wissensbasen

Als letzte Instanz soll hier die Übersetzung von Wissensbasen erwähnt werden, d.h. die Instanz, bei der

- $\mathcal{L}_s \neq \mathcal{L}_d$,
- $\mathcal{T} = \emptyset$ und
- ρ als eine der Relationen $\equiv, \sqsubseteq, \sqsupseteq$ gegeben ist.

Ziel ist es, den Austausch von Wissensbasen zwischen BL-Systemen, die auf zwei verschiedenen BLen $\mathcal{L}_s$ und $\mathcal{L}_d$ basieren, zu ermöglichen. Offensichtlich ist ein solcher Austausch möglich, wenn zu jeder $\mathcal{L}_s$-Konzeptbeschreibung C ein äquivalentes $\mathcal{L}_d$-Rewriting D existiert. Häufig wird eine solche exakte Übersetzung natürlich nicht möglich sein. In solchen Fällen kann dann aber noch versucht werden, C in $\mathcal{L}_d$ von unten (oben) durch D zu approximieren, d.h. man sucht bzgl. $\sqsubseteq$ eine maximale (minimale) $\mathcal{L}_d$-Konzeptbeschreibung D mit $D \sqsubseteq C$ ($C \sqsubseteq D$).

Eine mögliche Anwendungssituation für diesen Dienst im Rahmen der Prozesstechnik lässt sich wie folgt beschreiben: Ein für die Evaluation einer Nicht-Standardinferenz geeigneter Teil der Wissensbasis der Prozesstechniker ist in der BL $\mathcal{L}_s$ darstellbar, für das zu evaluierende Schlussfolgerungsproblem stehen aber nur Verfahren einer anderen Beschreibungslogik, $\mathcal{L}_d$, zur Verfügung. Dann kann man die $\mathcal{L}_s$-Wissensbasis mit Hilfe des Rewritings durch eine $\mathcal{L}_d$-Wissensbasis approximieren und darauf das Schlussfolgerungsverfahren für $\mathcal{L}_d$ anwenden. Anschließend muss man das so berechnete $\mathcal{L}_d$-Ergebnis wieder in $\mathcal{L}_s$ approximieren.

Die Übersetzung von Wissensbasen bildet auch häufig beim *Merging von Wissensbasen* einen notwendigen Schritt: Gegeben zwei Wissensbasen, formalisiert in zwei verschiedenen Formalismen, ist der erfolgversprechendste Weg diese zu mergen oft, beide zunächst in Wissensbasen eines einzigen Formalismus zu übersetzen und diese dann zu mergen. Diese Idee liegt dem in [Cha00] beschriebenen System OntoMorph zugrunde, das eine regelbasierte Übersetzung zwischen verschiedenen Wissensrepräsentationsformalismen erlaubt. Besonders interessant ist dabei, dass das BL-System PowerLoom (die aktuelle Version von LOOM) in das System integriert ist und als Zwischensprache zur Verfügung steht.

3.5 Neue Resultate

An dieser Stelle werden die technischen Resultate dieser Arbeit zusammengefasst. Die zugehörigen Beweise finden sich in den jeweiligen technischen Kapiteln.

Berechnung des LCS in BLen mit Existenzrestriktionen. Ausgehend von einer syntaktischen Charakterisierung der Subsumtion in $\mathcal{ALE}$ und den Teilsprachen $\mathcal{EL}$ und $\mathcal{FLE}$ (analog zum strukturellen Subsumtionsalgorithmus für Classic) zeigt Kapitel 5, dass in $\mathcal{EL}$, $\mathcal{FLE}$ und $\mathcal{ALE}$ der LCS einer endlichen Menge von Konzeptbeschreibungen stets existiert. In $\mathcal{EL}$ ist der LCS zweier Konzeptbeschreibungen polynomiell groß und in polynomieller Zeit berechenbar (polynomiell bzgl. der Größe der Ausgangskonzepte). Der LCS von $n > 2$ $\mathcal{EL}$-Konzeptbeschreibungen kann aber exponentiell groß in der Größe der Ausgangskonzepte sein. Demgegenüber kann bereits der LCS zweier $\mathcal{FLE}$- bzw. $\mathcal{ALE}$-Konzeptbeschreibungen exponentiell groß in der Größe der Ausgangskonzepte sein.

Berechnung des MSC in BLen mit Existenzrestriktionen. Zunächst wird die syntaktische Charakterisierung der Subsumtion für $\mathcal{EL}$ erweitert zu einer Charakterisierung der Instanz für $\mathcal{EL}$ und $\mathcal{EL}_\neg$, ($\mathcal{EL}_\neg$ bezeichnet die Erweiterung von $\mathcal{EL}$ um primitive Negation). Diese erlaubt den Nachweis, dass das MSC für *azyklische* $\mathcal{EL}$- bzw. $\mathcal{EL}_\neg$-ABoxen stets existiert und effektiv berechnet werden kann. Für *zyklische* $\mathcal{EL}$- bzw. $\mathcal{EL}_\neg$-ABoxen gelingt lediglich der Nachweis, dass für eine gegebene Schranke $k \in \mathbb{N}$ die sogenannte *k-Approximation*, d.h. die bzgl. $\sqsubseteq$ speziellste Konzeptbeschreibung mit Rollentiefe $\leq k$, von dem ein Individuum Instanz ist, effektiv berechnet werden kann. Sowohl das MSC im azyklischen Fall als auch die k-Approximation im zyklischen Fall können exponentiell groß bzgl. der Größe der ABox bzw. der Größe der ABox und k sein.

Für $\mathcal{ALE}$ ergibt sich nur das folgende, schwächere Resultat: Existenz und Berechenbarkeit der k-Approximation wurden nur für endliche Signaturen, d.h. endliche Mengen N_C und N_R, nachgewiesen. Diese Approximation kann wiederum exponentiell groß

TBox	aufgefaltet	nicht aufgefaltet
$\mathcal{FL}_0$	NP-vollständig	NP-vollständig
$\mathcal{ALN}$	NP-vollständig	in Σ_2^p, NP-hart
$\mathcal{ALE}$	NP-vollständig	in PSPACE, NP-hart
$\mathcal{ALC}$	PSPACE-vollständig	PSPACE-vollständig

Tabelle 3.3: Komplexitätsresultate für das Entscheidbarkeitsproblem zum minimalen Rewriting-Problem.

in k und der Größe der ABox sein. Für azyklische $\mathcal{ALE}$-ABoxen konnte Existenz und Berechenbarkeit des MSC nicht nachgewiesen werden. Grund hierfür ist, dass keine korrekte und vollständige Charakterisierung der Instanz in $\mathcal{ALE}$ gefunden wurde, aus der sich eine Charakterisierung des MSC für azyklische ABoxen ableiten oder zumindest die Menge der zu berücksichtigenden $\mathcal{ALE}$-Konzeptbeschreibungen hinreichend weit einschränken lässt.

Das minimale Rewriting-Problem. Das minimale Rewriting-Problem induziert auf natürliche Weise das folgende Entscheidbarkeitsproblem: Gegeben eine $\mathcal{L}$-Konzeptbeschreibung C, eine $\mathcal{L}$-TBox $\mathcal{T}$ und eine obere Schranke $k \in \mathbb{N}$, gibt es ein Rewriting D von C bzgl. $\mathcal{T}$ mit $|D| \leq k$? Dieses Problem wird für die BLen $\mathcal{FL}_0$, $\mathcal{ALN}$, $\mathcal{ALE}$ und $\mathcal{ALC}$ untersucht. Die erhaltenen Ergebnisse sind in Tabelle 3.3 zusammengefasst.

Für das eigentliche Berechnungsproblem folgt, da jede Konzeptbeschreibung Rewriting von sich selbst ist, dass stets ein minimales Rewriting existiert. Andererseits kann es in allen betrachteten BLen exponentiell viele minimale Rewritings zu einer Konzeptbeschreibung C bzgl. einer TBox $\mathcal{T}$ geben, sodass zwei Fälle zu unterscheiden sind: man interessiert sich (1) für die Berechnung *eines* minimalen Rewritings; oder (2) für die Berechnung *aller* minimalen Rewritings. Es ergeben sich die folgenden Resultate:

1. Für alle betrachteten BLen ist *ein* minimales Rewriting mit polynomiellem *Platz*aufwand berechenbar.

2. Für alle betrachteten BLen sind *alle* minimalen Rewritings mit exponentiellem *Zeit*aufwand berechenbar.

Der Nachweis der Berechenbarkeit beruht dabei auf dem nicht praktikablen Algorithmus, der im wesentlichen durch Aufzählen der Menge aller Konzeptbeschreibungen, die (bzgl. $|\cdot|$) kleiner als das Eingabekonzept sind, ein bzw. alle minimalen Rewritings bestimmt. Daher wurde (mit Blick auf die Anwendung) ein Algorithmus entwickelt, der minimale Rewritings in $\mathcal{ALE}$ zielgerichtet, d.h. durch direkte Manipulation des zu rewritenden Konzeptes, berechnet. Dieser Algorithmus bildet außerdem die Grundlage für einen heuristischen Algorithmus, der zwar im allgemeinen keine minimalen Rewritings berechnet, in der Anwendungen aber zu sehr guten Ergebnissen geführt hat. Darüberhinaus konnte der Unterschied zwischen den heuristisch berechneten und minimalen Rewritings formal gefasst und analysiert werden. Sowohl der verbesserte als auch der heuristische Algorithmus lassen sich auf $\mathcal{ALN}$ übertragen [BKM99b], allerdings wird im Rahmen dieser Arbeit auf die explizite Angabe dieser Algorithmen verzichtet, da sie zum einen keine neuen Erkenntnisse oder Einsichten liefern und zum anderen nicht in der Anwendung eingesetzt wurden.

Zu den bisher erzielten Resultaten für Nicht-Standardinferenzen lässt sich abschließend sagen, dass insgesamt – insbesondere durch die am Lehr- und Forschungsgebiet Theoretische Informatik der RWTH Aachen erzielten Ergebnisse [BN98, BK98, BKM99a, BKBM99, BK00a, BKM00, BM00, Küs00] – ein Stand der Forschung erreicht ist, der bereits über die Ergebnisse zu den Standardinferenzen in der zweiten Phase hinausgeht (vgl. die historische Einteilung in vier Phasen aus Abschnitt 3.1).

Kapitel 4

Prozessmodellierung und Beschreibungslogik-Systeme

Ziel dieses Kapitels ist die Beschreibung des für diese Arbeit zentralen Szenarios für den Einsatz der in Kapitel 3 eingeführten BL-Systeme im Anwendungsbereich Prozesstechnik, genauer: bei der in Kapitel 2 vorgestellten rechnergestützten Modellierung verfahrenstechnischer Prozesse.

Das Szenario wurde am Lehr- und Forschungsgebiet Theoretische Informatik entwickelt und mit Mitarbeitern des Lehrstuhls für Prozesstechnik auf die dortige Anwendung abgestimmt. Den Ausgangspunkt bildeten die Ergebnisse aus dem durch das Graduiertenkolleg „Informatik & Technik" geförderten Forschungsvorhaben „Terminologische Wissensrepräsentationssysteme in einer verfahrenstechnischen Anwendung" [Sat98]: es konnte nachgewiesen werden, dass sich BL-Systeme sehr gut eignen, um die der Modellierungsumgebung ModKit zugrundeliegende Wissensbasis auf Konsistenz zu prüfen und zu strukturieren, d.h. die Klassen in einer Spezialisierungshierarchie anzuordnen und Instanzbeziehungen zwischen Bausteinen und Klassen zu bestimmen. Hierauf aufbauend zielt das ebenfalls durch das Graduiertenkolleg „Informatik & Technik" geförderte und dieser Arbeit zugrundeliegende Folgeprojekt auf eine noch umfassendere Unterstützung: ausgehend von theoretischen Untersuchungen zu Nicht-Standardinferenzen werden neuartige Systemdienste entwickelt und bereitgestellt, die Anwender auch beim Aufbau und der Erweiterung einer Wissensbasis unterstützen.

Der Historie folgend beschreibt Abschnitt 4.1 zunächst die Idee hinter dem Einsatz von Standardinferenzen (bzw. der zugehörigen Systemdienste) zur Strukturierung der Wissensbasis. Abschnitt 4.2 geht dann auf das Szenario zur Unterstützung der Erweiterung der Wissensbasis durch Nicht-Standardinferenzen ein. In diesem Kapitel beschränken wir uns dabei auf eine abstrakte Darstellung der Ideen und verfolgten Vorgehensweisen. Konkrete Beispiele und Erfahrungen werden zusammen mit der Implementierung in Kapitel 9 beschrieben.

4.1 Unterstützung der Strukturierung der Wissensbasis durch Standardinferenzen

Wie bereits in Kapitel 2 erläutert, hängt eine erfolgreiche Prozessmodellierung mit ModKit unter anderem davon ab, dass Bausteine effektiv wiederverwendet werden können, d.h., dass sie (*i*) leicht wiedergefunden, (*ii*) an das aktuelle Modell angepasst und (*iii*) die angepassten Bausteine wieder in die Wissensbasis integriert werden können. Dies erfordert eine strukturierte Repräsentation und Speicherung der Bausteine. Im Falle von ModKit erfolgt die strukturierte Repräsentation (dem zugrundeliegenden Datenmodell VeDa folgend) framebasiert: Bausteine erhalten einen eindeutigen Namen und werden über einen entsprechenden Eintrag einer Klasse zugeordnet. Klassen werden ebenfalls durch Frames beschrieben und dann in einer Spezialisierungshierarchie angeordnet (s. [BM98a, Bau00] für die Definition von Syntax und Semantik der verwendeten Frame-Sprache).

Da die Wissensbasis von ModKit inzwischen einige hundert Klassen und entsprechend viele Bausteine enthält, ist intuitiv klar, dass die Anordnung der Klassen und die Zuordnung von Bausteinen zu Klassen nicht nur eine sehr zeitaufwendige, sondern auch eine sehr fehleranfällige Aufgabe ist, die per Hand nicht mehr zu bewältigen ist. Daher sind die Prozesstechniker an einem Repräsentationsformalismus interessiert, der ausdrucksstark genug ist, Klassen und Bausteine zu beschreiben, und zugleich eine Automatisierung der Strukturierung der Wissensbasis erlaubt. Im Rahmen ihrer Promotion [Sat98] konnte Ulrike Sattler in Zusammenarbeit mit Mitarbeitern vom Lehrstuhl für Prozesstechnik zeigen, dass Beschreibungslogiken einen solchen Formalismus liefern. Die dabei entwickelte Vorgehensweise zur Berechnung von Spezialisierungshierarchie und Instanzbeziehungen wird in Abbildung 4.1 illustriert.

Im ersten Schritt werden Klassenbeschreibungen in Konzeptdefinitionen einer TBox und Bausteine in Individuen einer ABox übersetzt. Zu einer Klasse f, die durch einen Frame F beschrieben wird, erhält man eine Konzeptdefinition der Form $f \doteq C(F)$, wobei sich $C(F)$ aus ausgewählten Einträgen in F ergibt. Auf Einzelheiten der Übersetzung werden wir in Kapitel 9 eingehen. Klassennamen entsprechen also den definierten Namen in der resultierenden TBox. Analog liefern die Namen der Bausteine die Individuen der ABox. Ein Baustein b, der durch einen Frame B beschrieben wird, liefert die Konzeptassertion $b : C(B)$, wobei sich $C(B)$ wiederum aus ausgewählten Einträgen in B ergibt. Darüberhinaus gibt es in ModKit Relationen, die Bausteine miteinander in Beziehung setzen. Solche Verknüpfungen werden in Rollenassertionen der Form $(b_1, b_2) : r$ übersetzt.

Die so erhaltene Wissensbasis wird im zweiten Schritt mit einem BL-System weiterverarbeitet. Treten Inkonsistenzen auf, so deuten diese auf Fehler in den Klassen- und Bausteinbeschreibungen hin. Sind diese Fehler beseitigt und die erhaltene Wissensbasis konsistent, so wird sie mit Hilfe der Standardinferenzen automatisch klassifiziert: die definierten Konzeptnamen werden in einer Subsumtionshierarchie angeordnet und

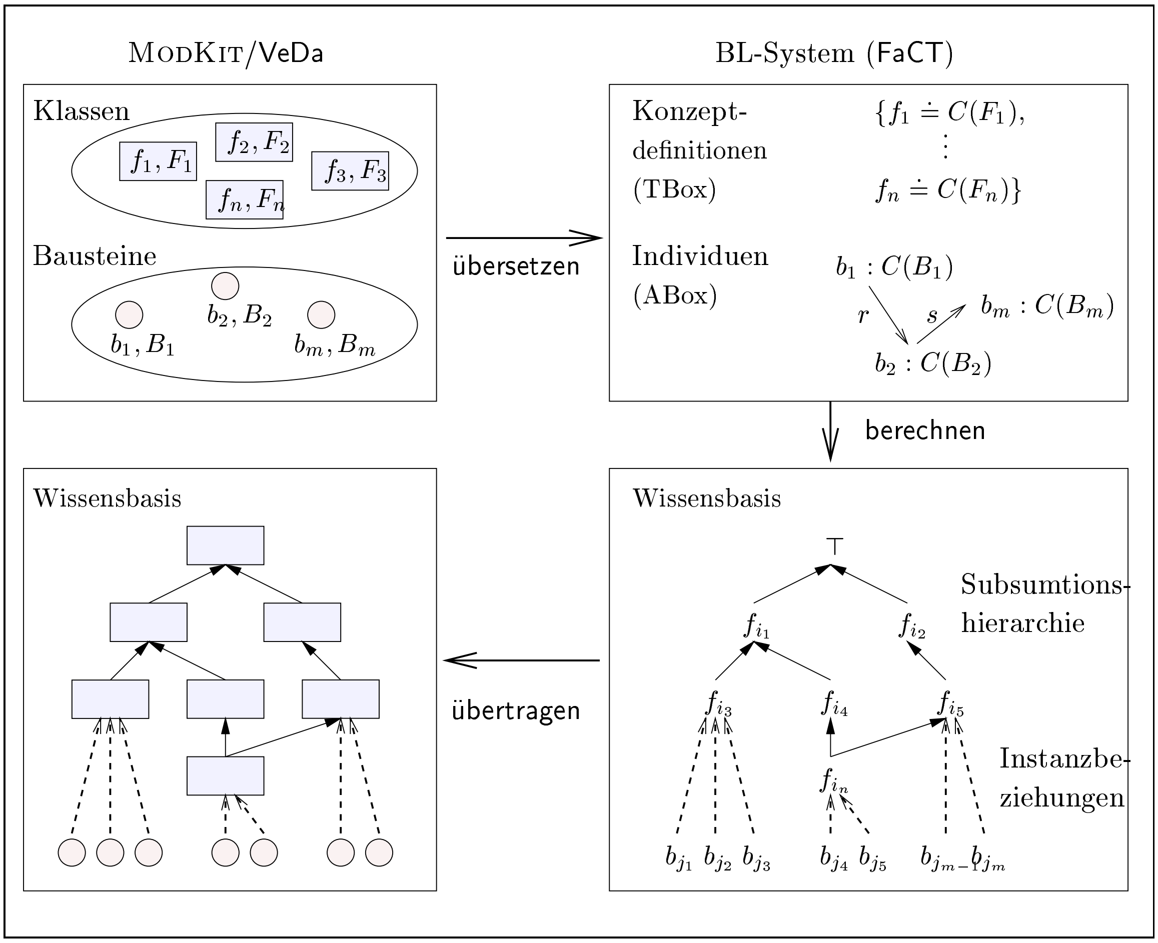


Abbildung 4.1: Der Einsatz von Standardinferenzen zur Strukturierung der Wissensbasis in der Prozesstechnikanwendung.

zu den Individuen wird jeweils die Menge der definierten Konzepte bestimmt, von denen sie Instanz sind. Man beachte, dass ein Individuum zu einem Baustein b aufgrund der möglichen zusätzlichen Rollenassertionen zu b auch Instanz eines Konzeptes sein kann, das nicht unmittelbar in $C(B)$ auftritt. Insbesondere lassen sich also die gültigen Instanzbeziehungen im allgemeinen nicht trivial aus der framebasierten Repräsentation ablesen.

Im letzten Schritt werden dann die Subsumtionshierarchie und die Instanzbeziehungen auf die Klassen und Bausteine übertragen. Wichtig ist hierbei natürlich, dass die Subsumtionshierarchie mit der Spezialisierungshierarchie sowie die Instanzbeziehungen miteinander übereinstimmen. Dazu wiederum sind auf Seiten der Klassen und Bausteine entsprechend relevante Aspekte aus den Beschreibungen auszuwählen und auf Seiten der BL-Systeme hinreichend ausdrucksstarke Logiken bereitzustellen, d.h. Logiken, in denen diese Aspekte ausgedrückt werden können. Für die gewählte BL muss außerdem das Erfüllbarkeits-, Subsumtions- und Instanzproblem entscheidbar sein und Implementierungen der entsprechenden Systemdienste zur Verfügung stehen. Das Hauptaugenmerk der theoretischen Arbeit in [Sat98] lag daher auf der Untersuchung von Entscheidbarkeit und Komplexität der Standardinferenzprobleme für sehr ausdrucksstarke BLen, z.B. Erweiterungen von $\mathcal{ALC}$ um transitive Rollen und Rollenhierarchien, durch die sich hat-Teil-Beziehungen in Modellen beschreiben lassen.

Im praktischen Teil der Arbeit wurde in Zusammenarbeit mit Mitarbeitern am Lehrstuhl für Prozesstechnik die Übersetzung der Beschreibungen in Konzeptdefinitionen und Assertionen automatisiert. Bei dieser Übersetzung liefern die aus Sicht der strukturellen Modellbeschreibung relevanten Aspekte der Klassen und Bausteine die ausgewählten Einträge für die Bestimmung der zugehörigen Konzeptbeschreibungen. Eingeschränkt durch die zum damaligen Zeitpunkt zur Verfügung stehenden BL-Systeme wurde als Zielsprache zunächst $\mathcal{ALC}$ gewählt und das BL-System Crack [BFT95] angebunden.[1] Genauere Untersuchungen der erhaltenen Wissensbasis ergaben, dass fast alle der von den Prozesstechnikern als relevant eingestuften Aspekte in Konjunktionen von Existenz- und Werterestriktionen übersetzt werden können. So geht die vorliegende Arbeit von einer Übersetzung der Wissensbasis aus, die sich aus einer Einschränkung der Zielsprache auf $\mathcal{ALE}$ mit azyklischen TBoxen ergibt. Die erhaltene Subsumtionshierarchie stimmt im wesentlichen mit der zuvor erhaltenen Hierarchie überein: nur vereinzelt werden Klassen wegen der Einschränkungen nun in äquivalente Konzeptbeschreibungen übersetzt und in der Subsumtionshierarchie nicht mehr unterschieden. Im folgenden konzentrieren wir uns also (insbesondere bei der Untersuchung der Nicht-Standardinferenzen in dieser Arbeit) auf die BL $\mathcal{ALE}$ mit azyklischen TBoxen.

Die Standardinferenzdienste eines BL-Systems können nach [Sat98] wie folgt bei der Strukturierung der Wissensbasis genutzt werden:

- Durch die Visualisierung der durch das BL-System berechneten Subsumtionshierarchie und Instanzbeziehungen erhält man einen Überblick über die Struktur der Wissensbasis und erkennt unerwartete oder unerwünschte Zusammenhänge.

- Bei der Erstellung eines neuen Modells ist es üblich, dass der Modellierer Bausteine aus dem Katalog oder anderen Modellen kopiert und an das aktuelle Modell anpasst. Die anschließende Einordnung der neuen Bausteine in die Wissensbasis wird durch den Instanztest unterstützt: mit ihm kann die speziellste Klasse aus der Menge der bereits definierten Klassen bestimmt werden, von der der neue Baustein Instanz ist.

[1] De facto stand nur das System Crack als effiziente Implementierung zur Verfügung. Da dieses System inzwischen nicht mehr gepflegt und weiterentwickelt wird, wurde es durch das leistungsfähigere System FaCT [Hor98] ersetzt.

- Die Definition einer neuen Klasse kann in ModKit über einen entsprechenden Dialog vom Modellierer vorgenommen werden, wobei er durch den Subsumtionstest unterstützt wird: durch das Übersetzen der Klassenbeschreibung und die automatische Einordnung in die Subsumtionshierarchie kann er feststellen, ob die Klasse an der erwarteten Position in der Hierarchie liegt. Ist dies nicht der Fall, kann er die Klassenbeschreibung geeignet modifizieren bevor sie tatsächlich in die Wissensbasis eingefügt wird.

Bereits in der ersten Phase unserer Kooperation wurde aber festgestellt, dass die oben skizzierte Unterstützung insbesondere bei der Definition neuer Klassen nicht ausreicht. Die Definition einer neuen Klasse ist ein sehr aufwendiger und fehleranfälliger Arbeitsschritt, da häufig nicht klar ist, was die relevanten Eigenschaften der neuen Klasse sind und wie sie im vorgegebenen Formalismus repräsentiert werden können. Wünschenswert ist daher ein Systemdienst, der bei Eingabe weniger und für den Modellierer leicht bereitzustellender Informationen eine Klassendefinition vorschlägt. Bei diesen Informationen könnte es sich beispielsweise um einige bereits vorliegende Bausteine handeln, die Instanzen der neuen Klasse sein sollen, oder aber um die Position der neuen Klasse in der Hierarchie der Wissensbasis. Der folgende Abschnitt beschreibt zwei Situationen, in denen ein solcher Systemdienst unter Verwendung der im vorherigen Kapitel eingeführten Nicht-Standardinferenzen bzw. der zugehörigen BL-Systemdienste realisiert werden kann.

4.2 Unterstützung der Erweiterung der Wissensbasis durch Nicht-Standardinferenzen

Erfahrungen der Prozesstechniker zeigen, dass von Zeit zu Zeit eine Restrukturierung der Wissensbasis notwendig ist, um (1) eine ausgewogene Struktur zu behalten und (2) zu große Klassen auf den unteren Ebenen der Hierarchie zu vermeiden. Restrukturierung bedeutet in diesem Zusammenhang, dass neue Klassen eingeführt werden: in (1) als Oberklasse einer Menge von Klassen bzw. in (2) als speziellste Klasse zu einer Menge von Bausteinen.

Im ersten Fall wird durch die Einführung einer neuen Klasse eine Zwischenebene in die Hierarchie eingezogen mit dem Ziel, Situationen aufzulösen, in denen eine Klasse sehr viele direkte Subklassen hat (s. Abbildung 4.2 links). Es ist intuitiv klar, dass diese Situation einen negativen Einfluss auf das Browsen der Hierarchie hat und damit das Wiederfinden von Bausteinen erschwert. Im zweiten Fall bestimmt man zu einer Menge von Bausteinen eine neue Klasse, die (*i*) alle diese Bausteine als Instanzen hat und (*ii*) die speziellste Klasse mit dieser Eigenschaft ist. Ordnet man diese neue Klasse in die Hierarchie ein, so erhält sie (mindestens) die Eingabemenge als Instanzen (s. Abbildung 4.4 links). Dadurch werden Situationen vermieden, in denen existierende Klassen (insbesondere auf den unteren Ebenen der Hierarchie) zu

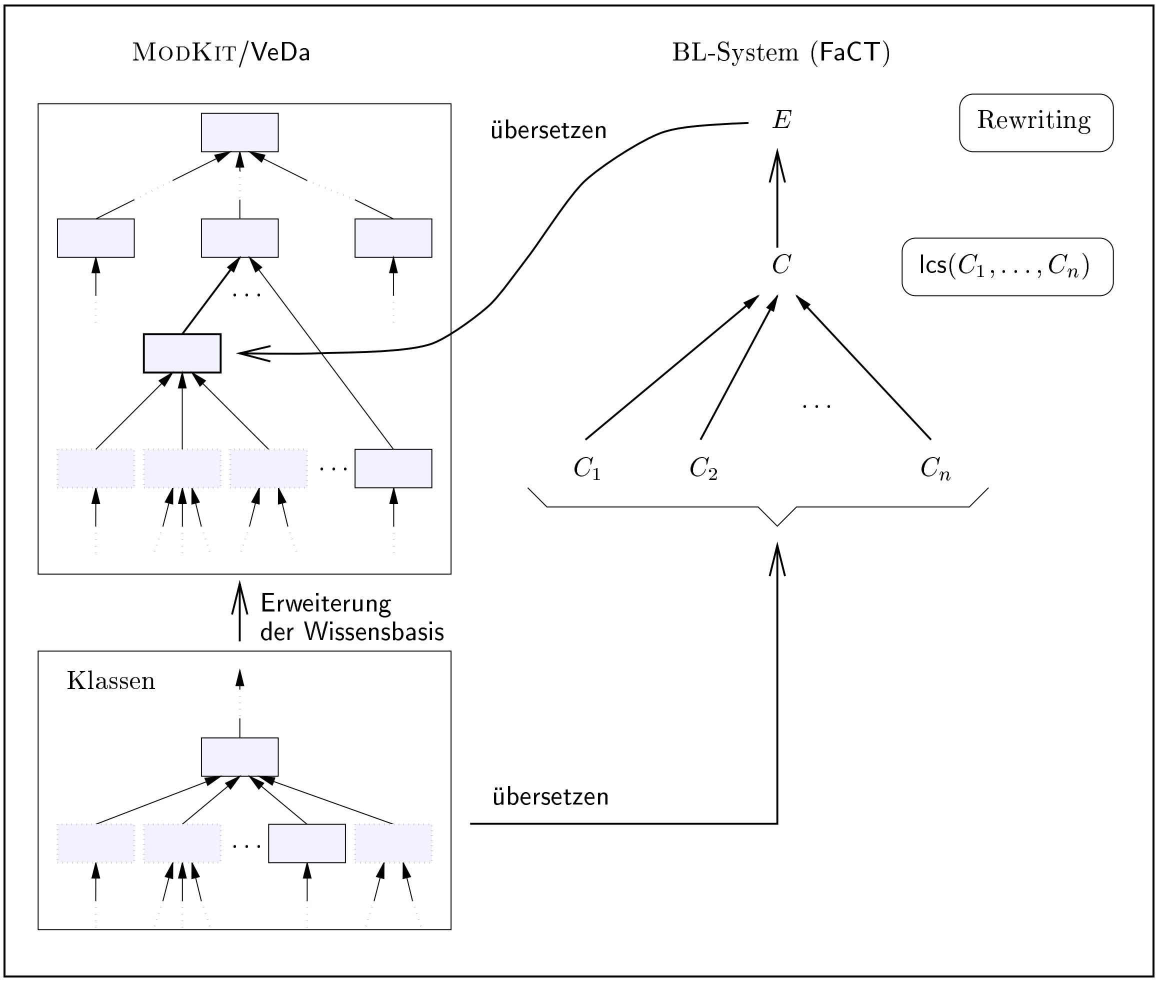


Abbildung 4.2: Der Einsatz von Nicht-Standardinferenzen bei der Definition einer neuen Oberklasse für eine Menge von Klassen.

groß werden, in denen also das Wiederfinden bestimmter Bausteine in einer Klasse zu aufwendig wird. In beiden Fällen bilden die neuen Klassen Abstraktionen der Eingabemengen, d.h. sie liegen in der Hierarchie der Wissensbasis über den betrachteten Klassen bzw. Bausteinen. In diesem Sinne wird also eine bottom-up Konstruktion bzw. Erweiterung der Wissensbasis vorgenommen (vgl. Abschnitt 3.4.2). Wie diese durch den Einsatz von Nicht-Standardinferenzdiensten unterstützt werden kann, wird nun im folgenden beschrieben.

Unterstützung der Definition einer neuen Oberklasse

Die Unterstützung der Definition einer neuen Oberklasse zu einer Menge von Klassen basiert im wesentlichen auf den folgenden drei Schritten:

1. übersetze die gegebenen Klassen in Konzeptbeschreibungen;

2. berechne zu diesen Konzeptbeschreibungen den LCS;
3. übersetze diese Konzeptbeschreibung zurück in eine Klassenbeschreibung.

Da die Übersetzung in Konzeptbeschreibungen nicht alle Informationen der Klassenbeschreibungen berücksichtigt, ist nicht sichergestellt, dass der LCS der erhaltenen Konzeptbeschreibungen auch tatsächlich eine sinnvolle neue Klasse liefert. Eine automatische Erzeugung und Einordnung einer neuen Klasse aus dem LCS ist also nicht möglich. Vielmehr wird dem Modellierer die zum LCS erhaltene Klassenbeschreibung für die Definition einer neuen Klasse vorgeschlagen. Diesen Vorschlag kann er akzeptieren, verwerfen oder die Beschreibung per Hand noch geeignet modifizieren.
Ein Systemdienst, der sich aus der unmittelbaren Umsetzung der drei Schritte ergeben würde, liefert aber aufgrund der folgenden Probleme im allgemeinen noch keine sinnvollen Vorschläge:

(i) Um vom Modellierer begutachtet werden zu können, muss der LCS lesbar und verständlich sein. Der aus den technischen Resultaten in Kapitel 5 gewonnene Algorithmus zur Berechnung des LCS in unserer Zielsprache $\mathcal{ALE}$ benötigt als Eingabe aufgefaltete Konzeptbeschreibungen in einer speziellen Normalform. Wie erste Experimente gezeigt haben, enthalten die mit diesem Algorithmus berechneten LCS einige hundert Vorkommen atomarer Konzept- und Rollennamen und sind damit unlesbar und können nicht begutachtet werden.

(ii) Die Nutzung der Nicht-Standardinferenzdienste bei der Definition neuer Klassen erfordert die Rückübersetzung von Konzeptbeschreibungen in Klassenbeschreibungen. Informell lässt sich diese Rückübersetzung von $\mathcal{ALE}$-Konzeptbeschreibungen in die Framesprache von ModKit wie folgt beschreiben: die Konzeptnamen auf Toplevel der Konzeptbeschreibung liefern die direkten Oberklassen der Klasse und jede Werte- und Existenzrestriktion der Form $\forall r.C/\exists r.C$ liefert einen Eintrag im Frame der Klasse. Diese Einträge sind im wesentlichen von der Form $\langle$*attribute name*, *domain*, *flags*$\rangle$. Der Rollenname r liefert den *attribute name* und der Quantor legt die Werte der *flags* fest. Handelt es sich bei C um einen Konzeptnamen, so ist er gleich dem *domain*. Handelt es sich bei C jedoch um eine komplexe Konzeptbeschreibung, so ist aus technischen Gründen eine Hilfsklasse einzuführen, deren Name dann im Eintrag zu $\forall r.C/\exists r.C$ als *domain* eingesetzt wird. Konzeptbeschreibungen mit einer großen Rollentiefe (wie etwa die berechneten LCS) führen also offensichtlich zu Kaskaden von Hilfsklassen und lassen sich nur schwer oder nicht sinnvoll rückübersetzen.

(iii) Um die Struktur der Wissensbasis an der betrachteten Stelle nachhaltig zu verbessern, d.h. tiefer und damit schlanker zu machen, reicht die Einführung einer neuen Oberklasse zur Menge der eingegebenen Klassen häufig nicht aus. Iteriertes Auswählen einer Menge von Klassen und Berechnen zugehöriger Oberklassen ist aber zum einen sehr aufwendig und zum anderen hängt die Qualität der erhaltenen Erweiterung sehr stark von der Auswahlfolge ab. Als Beispiel für eine qualitativ gute Erweiterung denke man dabei an eine Erweiterung, in der die aktuelle Oberklasse durch die neuen Klassen so aufgeteilt wird, dass diese Aufteilung der Intuition der Modellierer entspricht und sie daher beim zukünftigen Browsen der Hierarchie an dieser Stelle leicht eine Auswahl treffen können. Die Bestimmung jeweils *einer* Oberklasse zu einer Menge von Klassen bietet hier also noch keine ausreichende Unterstützung.

Zu den Problemen (i) und (ii): Um diese Probleme in den Griff zu bekommen, wird—wie in Abbildung 4.2 dargestellt—der Berechnung des LCS noch ein Rewriting-Schritt nachgeschaltet: durch einsetzen definierter Namen aus der Wissensbasis wird eine zum LCS äquivalente aber kleinere (nach Möglichkeit sogar minimale) Konzeptbeschreibung bestimmt (vgl. Abschnitt 3.4.3), die für den Modellierer lesbar und verständlich ist bzw. sinnvoll in Klassenbeschreibungen übersetzt werden kann. Ein alternativer Lösungsansatz besteht darin, den LCS-Algorithmus selbst so zu modifizieren, dass er definierte Namen korrekt berücksichtigt und keine aufgefalteten Konzepte zurückliefert. Gegen diesen Ansatz sprechen aber mehrere Gründe: So ist beispielsweise nicht von vorneherein klar, ob ein solcher Algorithmus existiert, d.h. ob ohne Auffalten und Bestimmen der Normalform auch tatsächlich der *Least* Common Subsumer berechnet werden kann. Außerdem werden wir in Kapitel 5 sehen, dass der LCS zweier $\mathcal{ALE}$-Konzeptbeschreibungen exponentiell groß in der Größe der Eingabekonzepte sein kann. In diesem Fall würde auch der modifizierte Algorithmus kein kleineres Konzept liefern. Desweiteren ist das Rewriting auch mit Blick auf andere Systemdienste interessant. Beispielsweise werden auch beim Matching von Konzeptbeschreibungen aufgefaltete und daher meist sehr große und schwer lesbare Konzepte berechnet. Es ist daher sinnvoll, nicht alle bereits entwickelten Algorithmen selbst zu ändern, sondern statt dessen das Rewriting als einen Dienst zur Verfügung zu stellen, der den anderen Diensten jeweils nachgeschaltet werden kann.

Zum Problem (iii): Ein natürlicher Lösungsansatz für dieses Problem lässt sich wie folgt beschreiben: statt zur gegeben Menge $\{C_1, \ldots, C_n\}$ von Konzeptbeschreibungen nur $\mathsf{lcs}(C_1, \ldots, C_n)$ zu berechnen, berechnet man die LCS aller Teilmengen von $\{C_1, \ldots, C_n\}$, ordnet diese in einer Subsumtionshierarchie an und präsentiert diese dem Anwender. Er wählt dann nach geeigneten Kriterien Kandidaten aus dieser Hierarchie, zu denen ihm dann die zugehörigen Klassenbeschreibungen vorgeschlagen werden. Diese kann er wie im einfachen Fall weiterverarbeiten und schließlich als neue Klassen in die Wissensbasis einfügen. Abbildung 4.3 veranschaulicht diesen Ansatz.

Bei der Umsetzung dieses Ansatzes muss aber besonderes Augenmerk auf einen effizienten Algorithmus zur Berechnung der LCS-Hierarchie gelegt werden: Zum einen sind formal die exponentiell vielen Teilmengen der Eingabemenge $\{C_1, \ldots, C_n\}$ zu betrachten und zum anderen handelt es sich bei der Berechnung des LCS um eine sehr komplexe Operation (cf Kapitel 5). In Kapitel 6 werden wir einen bekannten Algorithmus aus der *Formalen Begriffsanalyse* [GW99] vorstellen, der sich hier als geeignet erwiesen hat (cf Kapitel 9).

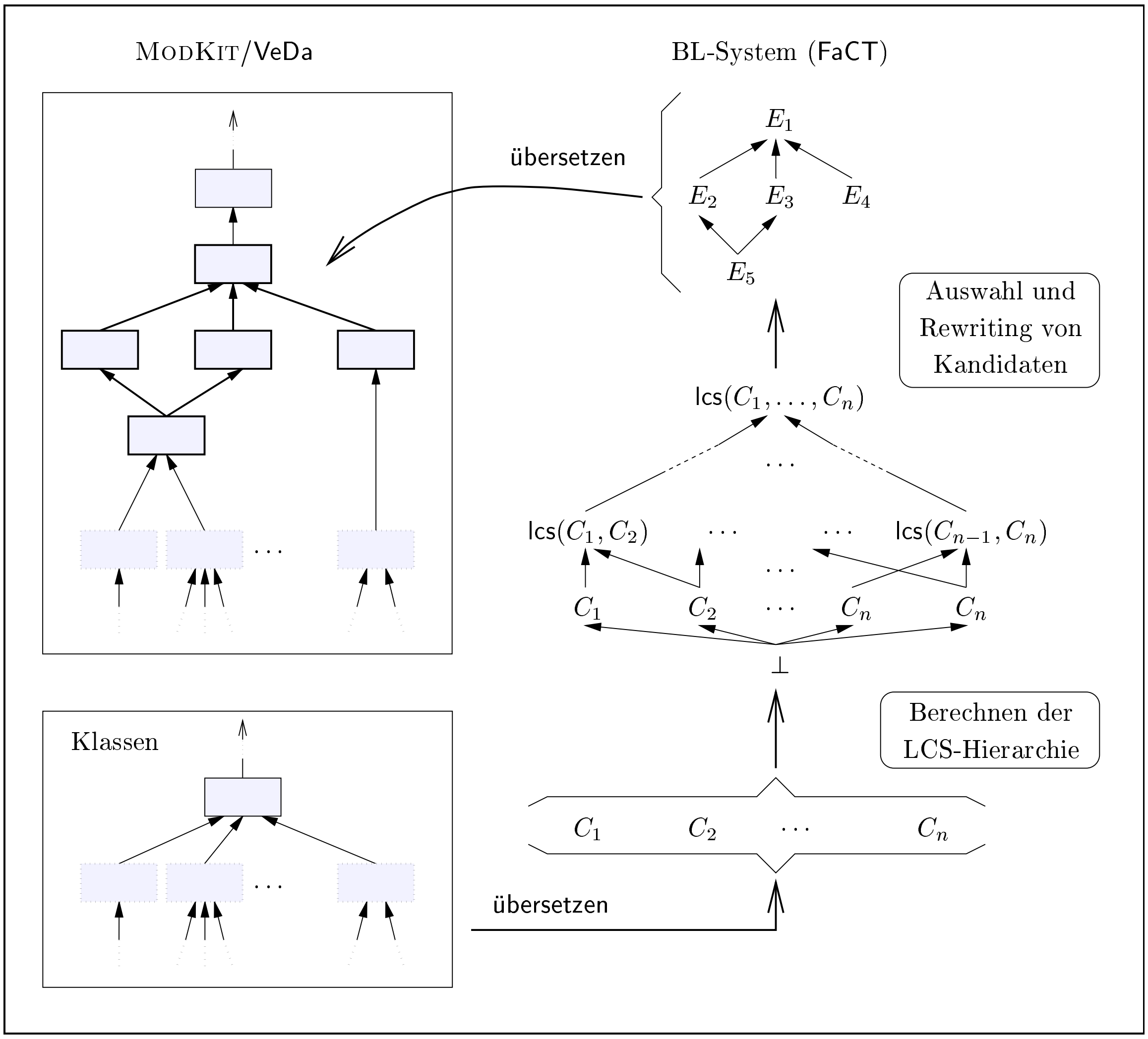


Abbildung 4.3: Die Berechnung der LCS-Hierarchie zu einer Menge von Klassen zur Bestimmung einer Erweiterung der Wissensbasis.

Werden die in den Lösungsansätzen zu den oben genannten Problemen geforderten Dienste auf Seiten der BL-Systeme bereitgestellt, so lässt sich mit dem in den Abbildungen 4.2 und 4.3 skizzierten Szenario die Definition einer neuen Oberklasse bzw. sogar eine weitreichendere Erweiterung der Hierarchie unterstützen. Für die Unterstützung der Definition einer neuen Klasse zu einer Menge von Bausteinen reichen die in diesem Szenario vorkommenden BL-Systemdienste aber noch nicht aus.

Unterstützung der Definition einer Klasse zu einer Menge von Bausteinen

Auch hier besteht die Idee der Unterstützung im wesentlichen darin, ausgehend von den Übersetzungen der Bausteine und ihrer Beschreibungen unter Verwendung von

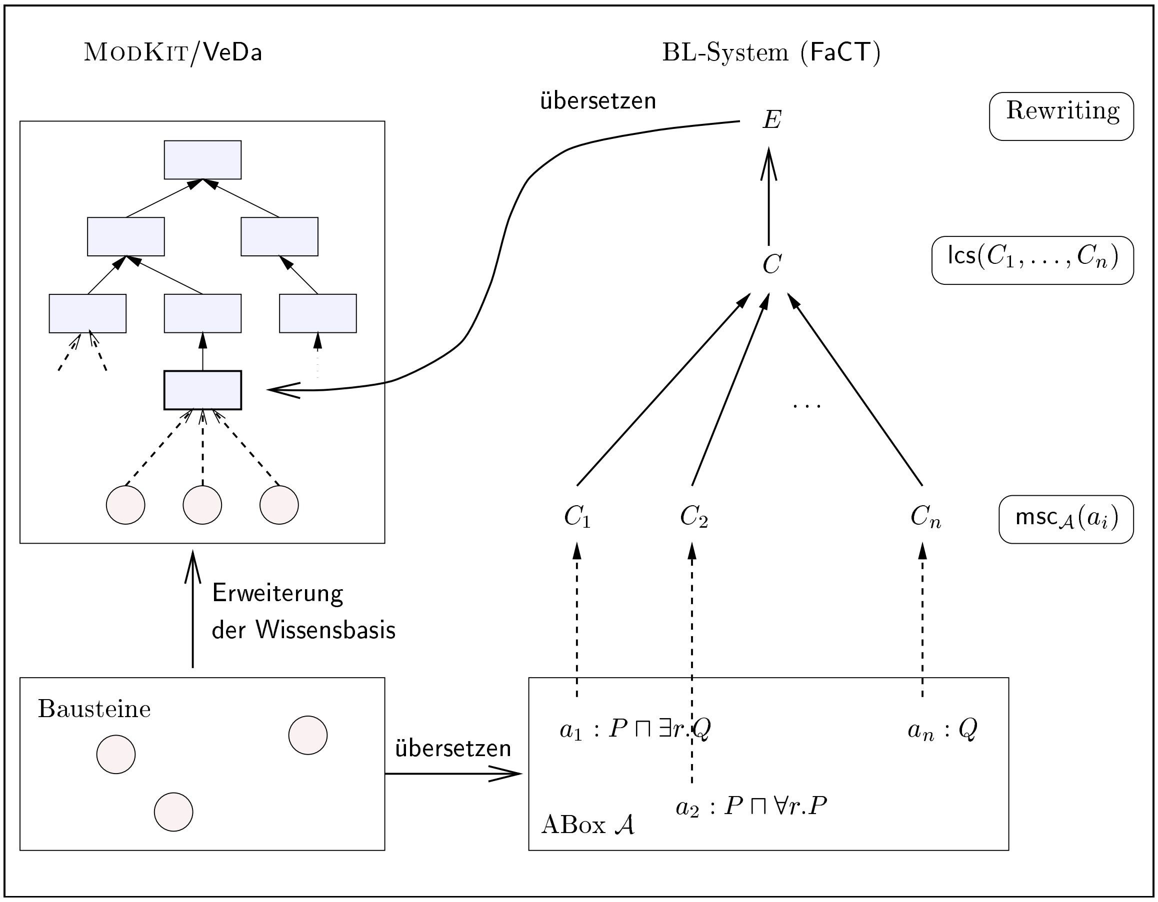


Abbildung 4.4: Der Einsatz von Nicht-Standardinferenzen bei der Definition einer neuen Klasse für eine Menge von Bausteinen.

LCS und Rewriting eine Konzeptbeschreibung zu bestimmen, deren zugehörige Klasse die speziellste ist, von der diese Bausteine Instanzen sind. Der grundlegenden Idee aus Abschnitt 4.1 folgend werden Bausteine und ihre Beschreibungen aber zunächst in Individuen und Assertionen einer ABox übersetzt. Um die speziellste Konzeptbeschreibung zu bestimmen, von denen die entsprechenden ABox-Individuen Instanzen sind, ist daher in einem ersten Schritt noch von den konkreten ABox-Individuen durch Berechnung der zugehörigen MSCs zu abstrahieren. Auf die so erhaltenen Konzeptbeschreibungen kann dann die LCS-Operation angewendet werden. Nach Definition erhält man damit die speziellste Konzeptbeschreibung, von der alle betrachteten Individuen Instanzen sind. Zur Bewertung durch den Modellierer und für die Übersetzung in Klassenbeschreibungen wird schließlich noch wie im ersten Szenario ein möglichst kleines Rewriting bestimmt. Abbildung 4.4 zeigt dieses erweiterte Szenario.

Die im ersten Szenario auftretenden Probleme (i) und (ii) treten auch im erweiterten Szenario auf und werden hier ebenfalls durch einen nachgeschalteten Rewriting-Schritt gelöst. Das Problem (iii) tritt hier in etwas abgewandelter Form auf: Die

Modellierer werden als Eingabe Bausteine wählen, die aus ihrer Sicht sehr ähnlich sind und in einer Klasse zusammengefasst werden sollten. Häufig wird aber der Fall eintreten, dass die tatsächlichen Beschreibungen der Bausteine bzw. die zugehörigen MSCs doch so unterschiedlich sind, dass der dazu berechnete LCS zu allgemein ist, also nicht eine Klasse an der gewünschten Position in der Hierarchie liefert. Wird der Dienst zur Berechnung einer Klasse zu einer Menge von Bausteinen zur Spezialisierung einer zu groß gewordenen Klasse f eingesetzt, kann der „zu allgemeine LCS" z.B. äquivalent zur Klasse f selbst sein. Wird der Dienst zur Definition einer neuen Klasse aus einer Menge typischer Instanzen (Beispiele) eingesetzt, d.h. bei der bottom-up Konstruktion der Wissensbasis, kann der „zu allgemeine LCS" z.B. eine bereits existierende, sehr allgemeine Klasse sein. Die Möglichkeit, sich bei einem zu allgemeinen Ergebnis durch iteriertes Auswählen einer Menge von Bausteinen und Berechnung des zugehörigen Vorschlags an die gewünschte Beschreibung der neuen Klasse heranzutasten, ist wiederum zu aufwendig.
Stattdessen wird auch hier der in Abbildung 4.3 skizzierte Lösungsansatz umgesetzt: Man berechnet nicht nur den LCS aller MSCs $\mathsf{msc}_{\mathcal{A}}(a_1), \ldots, \mathsf{msc}_{\mathcal{A}}(a_m)$, sondern die durch die LCS zu allen Teilmengen von $\{\mathsf{msc}_{\mathcal{A}}(a_1), \ldots, \mathsf{msc}_{\mathcal{A}}(a_m)\}$ induzierte Subsumtionshierarchie. Diese wird dem Modellierer präsentiert und er kann anhand geeigneter Kriterien Kandidaten wählen, diese begutachten und aus diesen schließlich eine (oder mehrere) neue Klasse(n) definieren. Geeignete Kriterien sind hier beispielsweise die Bedingung, dass ein Kandidat mindestens 50% der Beispiele als Instanzen haben muss und die zugehörige Klasse eine „günstige" Position in der Hierarchie einnimmt. Die Kriterien sollten außerdem für die potentiellen Kandidaten automatisch geprüft werden können, damit automatisch eine Vorauswahl von Kandidaten für den Modellierer bestimmt werden kann.

Zusammenfassend ergibt sich aus den in den Abbildungen 4.2, 4.3 und 4.4 skizzierten Szenarien auf Seiten der Beschreibungslogik die Aufgabe, die Berechnung des LCS, des MSC und minimaler Rewritings sowie die Berechnung von Hierarchien von LCS formal zu untersuchen und geeignete Algorithmen zu entwickeln. Als „Zielsprache" wird dabei jeweils die BL $\mathcal{ALE}$ betrachtet.
Kapitel 5 untersucht LCSs in $\mathcal{ALE}$ und liefert einen Algorithmus zur Berechnung des LCS von n $\mathcal{ALE}$-Konzeptbeschreibungen. Durch diesen lassen sich natürlich auch Konzeptbeschreibungen modulo einer azyklischen $\mathcal{ALE}$-TBox durch vorheriges Auffalten verarbeiten.
Kapitel 6 gibt eine kurze Einführung in die Formale Begriffsanalyse, genauer in die Berechnung sogenannter Begriffsverbände, und stellt einen Algorithmus vor, mit dem sich diese Verbände effizient berechnen lassen. Es wird dann gezeigt, dass die gesuchte LCS-Hierarchie einem speziellen Begriffsverband entspricht und der zuvor eingeführte Algorithmus zur Berechnung dieses speziellen Begriffsverbandes eingesetzt werden kann.
Kapitel 7 betrachtet die Berechnung von MSCs in $\mathcal{ALE}$. Leider hat sich gezeigt, dass das MSC in $\mathcal{ALE}$ im allgemeinen nicht existiert. Ursache hierfür sind zyklische

Abhängigkeiten zwischen den Individuen der ABox. Daher liegt das Hauptaugenmerk des Kapitels auf der Approximation des MSC in $\mathcal{ALE}$. Ziel ist, zu einer festen gegebenen Rollentiefe k das speziellste Konzept mit Tiefe kleiner gleich k zu bestimmen, von dem das Individuum Instanz ist. Wie im Falle des LCS lassen sich auch hier durch Auffalten Konzeptbeschreibungen (in der ABox) modulo einer azyklischen $\mathcal{ALE}$-TBox behandeln.

Kapitel 8 beschäftigt sich schließlich mit dem minimalen Rewriting-Problem. Zunächst wird die Komplexität des zugehörigen Entscheidbarkeitsproblems für $\mathcal{ALE}$ (aber auch für die BLen $\mathcal{FL}_0$, $\mathcal{ALN}$ und $\mathcal{ALC}$) untersucht, um einen Eindruck von der Komplexität des Berechnungsproblems zu erhalten. Aufbauend auf einen nichtdeterministischen Algorithmus zur Berechnung (aller) minimaler Rewritings in $\mathcal{ALE}$ wird dann ein heuristischer Algorithmus vorgestellt, der kleine aber nicht immer minimale Rewritings in $\mathcal{ALE}$ deterministisch berechnet.

Im Anschluss an die theoretischen Kapitel wird in Kapitel 9 auf die Implementierung der zuvor entwickelten Algorithmen, d.h. auf die Realisierung der hier beschriebenen Szenarien, die Anbindung des BL-Systems FaCT an die Modellierungsumgebung ModKit sowie Erfahrungen mit den Nicht- und den Standard-Inferenzdiensten eingegangen.

Kapitel 5

Least Common Subsumer

In diesem und den folgenden Kapiteln werden nun die technischen Resultate zu den in Kapitel 3 eingeführten Nicht-Standard Inferenzen „Berechnung des LCS“ (Kapitel 5 und 6), „Berechnung des MSC“ (Kapitel 7) und „Rewriting von Konzeptbeschreibungen“ (Kapitel 8) vorgestellt.

In Abschnitt 5.1 wird nochmals ein kurzer Überblick über bekannte Ergebnisse zur Berechnung des LCS gegeben. Darüberhinaus wird die zu diesen bekannten Ergebnissen führende Vorgehensweise zur Charakterisierung des LCS herausgearbeitet. Diese wird in Abschnitt 5.2 zunächst durch Übertragung auf die ausdrucksschwache BL $\mathcal{EL}$ verdeutlicht. Die erhaltenen Resultate werden dann in Abschnitt 5.3 auf unsere Zielsprache $\mathcal{ALE}$ erweitert.

5.1 Bekannte Ergebnisse

In Abhängigkeit von der betrachteten BL muss der LCS von zwei oder mehr Konzeptbeschreibungen im allgemeinen nicht existieren. Beispielsweise existiert in der BL $\mathcal{L}_\sqcap$, die nur Konjunktion von Konzeptnamen erlaubt, der LCS zweier verschiedener Konzeptnamen P und Q nicht, da er gleich $\top$ wäre, was aber in $\mathcal{L}_\sqcap$ nicht ausgedrückt werden kann. Wie bereits in Abschnitt 3.4.2 erwähnt, ist der LCS, wenn er in der betrachteten BL stets existiert, eindeutig modulo Äquivalenz. In der Literatur [BKM99a, BK98, FP96, CH94b, CH94a, CBH92] beschränkt man sich meist auf die Betrachtung des Problems, den LCS *zweier* Konzeptbeschreibungen zu berechnen, da der LCS für $n > 2$ Konzeptbeschreibungen durch iteriertes Anwenden der binären LCS-Operation erhalten werden kann (cf den Paragraphen *Least Common Subsumer und Produkte*).

Die ersten Ergebnisse und Algorithmen zum LCS bezogen sich alle auf (Teilsprachen von) Classic [CH94b, CH94a, CBH92]. (Eine Übersicht über alle in Classic verfügbaren Konstruktoren findet sich in [BP94].) Es hat sich aber herausgestellt [BK98, KB99],

$$
\begin{tikzcd}
& C' \arrow[d, dashed, "!\sqsubseteq"] \arrow[ddl, bend right, "\sqsubseteq"'] \arrow[ddr, bend left, "\sqsubseteq"] & \\
& C \arrow[dl, "\sqsubseteq"'] \arrow[dr, "\sqsubseteq"] & \\
C_1 & & C_2
\end{tikzcd}
\qquad
\begin{tikzcd}
& \mathcal{G}_{C'} \arrow[d, dashed, "!\varphi"] \arrow[ddl, bend right, "\varphi_1"'] \arrow[ddr, bend left, "\varphi_2"] & \\
& \mathcal{G}_{C_1} \times \mathcal{G}_{C_2} \arrow[dl, "\pi_1"'] \arrow[dr, "\pi_2"] & \\
\mathcal{G}_{C_1} & & \mathcal{G}_{C_2}
\end{tikzcd}
$$

Abbildung 5.1: Bedingungen an den LCS zweier Konzeptbeschreibungen.

dass die vorgestellten Algorithmen jeweils nur für Fragmente der betrachteten Sprachen korrekt sind. Beispielsweise berücksichtigt keiner der Algorithmen inkonsistente Konzepte, die in $\mathcal{ALN}$ und damit auch in Classic ausgedrückt werden können. Erst weiterführende Untersuchungen haben zu exakten Komplexitätsresultaten und korrekten Algorithmen zur Berechnung des LCS geführt bzw. gezeigt, dass der LCS in bestimmten Teilsprachen von Classic im allgemeinen nicht existiert [BK98, KB99]. Zur Berechnung des LCS in BLen mit Existenzrestriktionen fanden sich vor den Arbeiten am LuFG Theoretische Informatik keine Resultate in der Literatur.

Least common subsumer und Produkte

Es besteht ein enger Zusammenhang zwischen dem LCS und dem aus der Kategorientheorie bekannten Begriff des *Produktes* [Pie91, HS73]. Man betrachte die Kategorie, in der

- Objekte Konzeptbeschreibungen sind und
- ein Morphismus von einem Objekt C_1 zu einem Objekt C_2 genau dann existiert, wenn C_2 von C_1 subsumiert wird.

Bezüglich dieser Kategorie lassen sich die in der Definition 3.15 formulierten Bedingungen an einen LCS C von C_1 und C_2 in Form eines Diagramms wie in Abbildung 5.1 links gezeigt darstellen. Dabei repräsentieren die durchgezogenen und mit $\sqsubseteq$ markierten Pfeile gültige Subsumtionsbeziehungen, wobei die Pfeilspitze auf den Subsumee zeigt. Der gestrichelte und mit '$!\sqsubseteq$' markierte Pfeil bedeutet, dass die Subsumtionsbeziehung gelten soll, wobei die Spitze wiederum auf den Subsumee deutet. Ein solches Diagramm ist auch als *Produktdiagramm* in der Kategorientheorie bekannt [Pie91, HS73].

Insbesondere entspricht also der LCS zweier Konzeptbeschreibungen dem Produkt der zugehörigen Objekte in der oben eingeführten Kategorie. Mit aus der Kategorientheorie bekannten Eigenschaften der Produktoperation folgt, dass sich der LCS von n Konzeptbeschreibungen durch iteriertes Anwenden der binären LCS-Operation bestimmen lässt.

Die Algorithmen zur Berechnung des LCS in [CH94b, FP96, Küs00] basieren alle auf der folgenden Vorgehensweise: ausgehend von strukturellen Charakterisierungen der Subsumtion, d.h. Charakterisierungen, die auf syntaktischen Vergleichen geeigneter Normalformen beruhen, wird der LCS zweier Konzeptbeschreibungen mit Hilfe ihrer Normalformen charakterisiert. Diese Beschreibung liefert dann auf natürliche Weise einen Algorithmus zur Berechnung des LCS. Es sei bereits an dieser Stelle darauf hingewiesen, dass bei allen betrachteten BLen die Schwierigkeit dieses Ansatzes jeweils in der Entwicklung und dem Beweis von Korrektheit und Vollständigkeit der strukturellen Charakterisierung der Subsumtion liegt und nicht so sehr in der anschließenden Charakterisierung des LCS.

Auch für $\mathcal{ALE}$ führt der beschriebene Ansatz zum Ziel. Dabei beruht die strukturelle Charakterisierung der Subsumtion in $\mathcal{ALE}$ auf der Idee, Konzeptbeschreibungen C in sogenannte *Beschreibungsbäume* $\mathcal{G}_C$ zu übersetzen. Subsumtion zwischen Konzeptbeschreibungen wird dann auf die Frage reduziert, ob ein Homomorphismus zwischen den zugehörigen Beschreibungsbäumen existiert. Ausgehend von dieser Charakterisierung folgt dann leicht, dass das Produkt der Bäume, welches sich durch eine geeignete Anpassung des graphentheoretischen Produktes auf Beschreibungsbäume ergibt, den LCS liefert. Abbildung 5.1 zeigt rechts das entsprechende Produktdiagramm. Dabei stehen die Kantenlabel jeweils für einen Homomorphismus vom Beschreibungsbaum des Subsumierers in den Beschreibungsbaum des Subsumees; im Falle des Produktes $\mathcal{G}_{C_1} \times \mathcal{G}_{C_2}$ handelt es sich bei π_i jeweils um die Projektion der $i$ten Komponente, d.h. die Abbildung, die ein Tupel von Knoten jeweils auf die $i$te Komponente abbildet.

5.2 Least Common Subsumer in $\mathcal{EL}$

Die Übertragung des im vorherigen Abschnitts beschriebenen prinzipiellen Ansatzes zur Charakterisierung des LCS auf BLen mit Existenzrestriktionen liefert eine Charakterisierung der Subsumtion mit Hilfe von Bäumen und Homomorphismen. Da diese Charakterisierung und die darauf aufbauende Charakterisierung des LCS für $\mathcal{ALE}$ recht aufwendig wird, sollen sie in diesem Abschnitt zunächst mit Hilfe der ausdrucksschwächeren BL $\mathcal{EL}$ verdeutlicht werden.

5.2.1 Charakterisierung der Subsumtion in $\mathcal{EL}$

Wie bereits erwähnt basiert die strukturelle Charakterisierung der Subsumtion für die hier betrachteten BLen auf der Repräsentation von Konzeptbeschreibungen durch Beschreibungsbäume. Im Falle von $\mathcal{EL}$-Konzeptbeschreibungen ergibt sich der zugehörige $\mathcal{EL}$-Beschreibungsbaum im wesentlichen unmittelbar aus der Termstruktur der Konzeptbeschreibung.

Definition 5.1 ($\mathcal{EL}$-Beschreibungsbäume) *Sei N_C eine Menge von Konzeptnamen und N_R eine Menge von Rollennamen. Ein* $\mathcal{EL}$-Beschreibungsbaum *ist ein beschrifteter Baum der Form $\mathcal{G} = (V, E, v_0, \ell)$ mit*

- *Wurzel $v_0 \in V$,*
- *$E \subseteq V \times N_R \times V$ und*
- *einer Beschriftungsfunktion ℓ, die jedem Knoten $v \in V$ eine Menge $\ell(v) \subseteq N_C$ zuordnet, wobei das leere Label $\top$ entspricht.*

Eine Kante $vrw \in E$ wird im folgenden als $\exists$-Kante *bezeichnet und der Knoten w als* r-Nachfolger *bzw.* direkter Nachfolger *von v. Für $v \in V$ bezeichnet $\mathcal{G}(v)$ den* Unterbaum *von $\mathcal{G}$ mit Wurzel v.*

Formal ist die Übersetzung von Konzeptbeschreibungen in Beschreibungsbäume und umgekehrt induktiv über die Tiefe von Konzeptbeschreibungen bzw. Beschreibungsbäumen erklärt. Für jede der in dieser Arbeit betrachteten BLen $\mathcal{L}$ (s. Tabelle 3.2) ist die *Tiefe depth*(C) einer $\mathcal{L}$-Konzeptbeschreibung C definiert als die Schachtelungstiefe der Quantoren in C, wobei zusätzlich die Tiefe von Zahlenrestriktionen als 1 definiert ist. Die *Tiefe depth*$(\mathcal{G})$ eines $\mathcal{EL}$-Beschreibungsbaumes $\mathcal{G}$ ist definiert als die Länge des längsten Pfades in $\mathcal{G}$.
Im folgenden gehen wir o.B.d.A. davon aus, dass jede $\mathcal{EL}$-Konzeptbeschreibung C in der folgenden Normalform gegeben ist:

$$C = P_1 \sqcap \ldots \sqcap P_n \sqcap \exists r_1.C_1 \sqcap \ldots \sqcap \exists r_m.C_m,$$

wobei $P_i \in N_C \cup \{\top\}$ und jedes C_i ebenfalls in dieser Normalform ist. Liegt die Konzeptbeschreibung in dieser Normalform vor, lässt sich die Übersetzung in einen $\mathcal{EL}$-Beschreibungsbaum $\mathcal{G}(C) := (V, E, v_0, \ell)$ besonders leicht induktiv definieren:

Für $\mathit{depth}(C) = 0$ definiere $V := \{v_0\}$, $E := \emptyset$ und $\ell(v_0) := \{P_1, \ldots, P_n\} \setminus \{\top\}$.

Für $\mathit{depth}(C) > 0$ seien $\mathcal{G}(C_i) = (V_i, E_i, v_{0i}, \ell_i)$ die induktiv definierten Beschreibungsbäume zu C_i, $1 \leq i \leq m$; o.B.d.A. seien die V_i und $\{v_0\}$ paarweise disjunkt. Definiere

- $V := \{v_0\} \cup \bigcup_{1 \leq i \leq m} V_i$,
- $E := \{v_0 r_i v_{0i} \mid 1 \leq i \leq m\} \cup \bigcup_{1 \leq i \leq m} E_i$ und
- $\ell(v) := \begin{cases} \{P_1, \ldots, P_n\} \setminus \{\top\}, & v = v_0 \\ \ell_i(v), & v \in V_i \text{ für ein } i \in \{1, \ldots, m\} \end{cases}$

Entsprechend wird ein $\mathcal{EL}$-Beschreibungsbaum $\mathcal{G} = (V, E, v_0, \ell)$ induktiv in eine $\mathcal{EL}$-Konzeptbeschreibung $C_{\mathcal{G}}$ übersetzt:

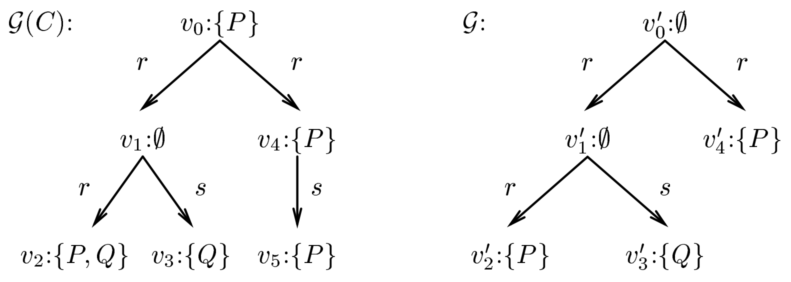


Abbildung 5.2: $\mathcal{EL}$-Beschreibungsbäume.

Für $depth(\mathcal{G}) = 0$ definiere

$$C_{\mathcal{G}} := \begin{cases} \top, & \text{falls } \ell(v_0) = \emptyset \\ P_1 \sqcap \ldots \sqcap P_n, & \text{falls } \ell(v_0) = \{P_1, \ldots, P_n\}, n \geq 1. \end{cases}$$

Für $depth(\mathcal{G}) > 0$ sei $\ell(v_0) = \{P_1, \ldots, P_n\}$ und $\{v_1, \ldots, v_m\}$ die Menge aller direkten Nachfolger von v_0 mit $v_0 r_i v_i \in E$, $1 \leq i \leq m$. Für $1 \leq i \leq m$ sei C_i die zum Unterbaum von $\mathcal{G}$ mit Wurzel v_i induktiv definierte Konzeptbeschreibung. Definiere $C_{\mathcal{G}} := P_1 \sqcap \ldots \sqcap P_n \sqcap \exists r_1.C_1 \sqcap \ldots \sqcap \exists r_m.C_m$.

Beispiel 5.2 *Die Übersetzung der $\mathcal{EL}$-Konzeptbeschreibung $C = P \sqcap \exists r.(\exists r.(P \sqcap Q) \sqcap \exists s.Q) \sqcap \exists r.(P \sqcap \exists s.P)$ ergibt den $\mathcal{EL}$-Beschreibungsbaum $\mathcal{G}(C)$ links in Abbildung 5.2. Der rechts in Abbildung 5.2 dargestellte $\mathcal{EL}$-Beschreibungsbaum $\mathcal{G}$ liefert die $\mathcal{EL}$-Konzeptbeschreibung $C_{\mathcal{G}} = \exists r.(\exists r.P \sqcap \exists s.Q) \sqcap \exists r.P$.*

Die *Semantik* von $\mathcal{EL}$-Beschreibungsbäumen $\mathcal{G}$ bzgl. einer Interpretation $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ ist durch die Semantik der zugehörigen $\mathcal{EL}$-Konzeptbeschreibungen definiert, d.h. $\mathcal{G}^{\mathcal{I}} := C_{\mathcal{G}}^{\mathcal{I}}$. Folglich sind Subsumtion und Äquivalenz zwischen $\mathcal{EL}$-Beschreibungsbäumen erklärt durch: $\mathcal{G} \sqsubseteq \mathcal{G}'$ ($\mathcal{G} \equiv \mathcal{G}'$) genau dann wenn $C_{\mathcal{G}} \sqsubseteq C_{\mathcal{G}'}$ ($C_{\mathcal{G}} \equiv C_{\mathcal{G}'}$).
Durch Induktion über die Tiefe von C bzw. $\mathcal{G}$ sieht man leicht, dass die Übersetzung von $\mathcal{EL}$-Konzeptbeschreibungen in $\mathcal{EL}$-Beschreibungsbäume (und umgekehrt) äquivalenzerhaltend ist, d.h.

$$C \equiv C_{\mathcal{G}(C)} \text{ und } \mathcal{G} \equiv \mathcal{G}(C_{\mathcal{G}}). \tag{5.1}$$

Zur Charakterisierung der Subsumtion zwischen $\mathcal{EL}$-Konzeptbeschreibungen unter Verwendung der zugehörigen Beschreibungsbäume fehlt noch die Definition des Begriffes *Homomorphismus zwischen Beschreibungsbäumen.*

Definition 5.3 (Homomorphismen zwischen $\mathcal{EL}$-Beschreibungsbäumen)
Seien $\mathcal{G} = (V_G, E_G, v_0, \ell_G)$ und $\mathcal{H} = (V_H, E_H, w_0, \ell_H)$ $\mathcal{EL}$-Beschreibungsbäume. Eine Abbildung $\varphi : V_H \longrightarrow V_G$ ist ein Homomorphismus *von $\mathcal{H}$ nach $\mathcal{G}$ genau dann, wenn die folgenden Bedingungen erfüllt sind:*

1. $\varphi(w_0) = v_0$,

2. $\ell_H(v) \subseteq \ell_G(\varphi(v))$ *für alle* $v \in V_H$ *und*

3. $\varphi(v)r\varphi(w) \in E_G$ *für alle* $vrw \in E_H$.

Die Abbildung φ *ist ein* Isomorphismus *von* $\mathcal{H}$ *nach* $\mathcal{G}$ *genau dann, wenn* φ *ein bijektiver Homomorphismus ist und zusätzlich*

4. $\ell_H(v) = \ell_G(\varphi(v))$ *für alle* $v \in V_H$

gilt. In diesem Fall heißt $\mathcal{H}$ isomorph *zu* $\mathcal{G}$, *kurz* $\mathcal{H} \cong \mathcal{G}$.

Wiederum durch Induktion über die Tiefe sieht man leicht, dass jeder $\mathcal{EL}$-Beschreibungsbaum $\mathcal{G}$ isomorph ist zum Beschreibungsbaum der zugehörigen $\mathcal{EL}$-Konzeptbeschreibung $C_\mathcal{G}$, d.h.

$$\mathcal{G} \cong \mathcal{G}(C_\mathcal{G}). \tag{5.2}$$

Diese Eigenschaft wird im Beweis der Korrektheit und Vollständigkeit der Charakterisierung der Subsumtion (und des LCS) benötigt. Das folgende Theorem charakterisiert nun die Subsumtion in $\mathcal{EL}$ unter Verwendung der oben eingeführten Begriffe.

Theorem 5.4 *Seien* C, D *zwei* $\mathcal{EL}$-*Konzeptbeschreibungen und* $\mathcal{G}(C), \mathcal{G}(D)$ *die zugehörigen* $\mathcal{EL}$-*Beschreibungsbäume. Dann gilt* $C \sqsubseteq D$ *genau dann, wenn ein Homomorphismus* φ *von* $\mathcal{G}(D)$ *nach* $\mathcal{G}(C)$ *existiert.*

Theorem 5.4 folgt unmittelbar aus der Verallgemeinerung der Aussage auf $\mathcal{ALE}$ in Theorem 5.21, die in Abschnitt 5.3.1 bewiesen wird.

Beispiel 5.5 (Fortsetzung von Beispiel 5.2) *Für die* $\mathcal{EL}$-*Beschreibungsbäume in Abbildung 5.2 liefert die Abbildung* $\{v_i' \mapsto v_i \mid 0 \leq i \leq 4\}$ *offensichtlich einen Homomorphismus von* $\mathcal{G}$ *nach* $\mathcal{G}(C)$. *Da* $\mathcal{G}$ *isomorph zu* $\mathcal{G}(C_\mathcal{G})$ *ist, existiert also auch ein Homomorphismus von* $\mathcal{G}(C_\mathcal{G})$ *nach* $\mathcal{G}(C)$ *und mit Theorem 5.4 folgt* $C \sqsubseteq C_\mathcal{G}$.

Bevor im nächsten Abschnitt der LCS in $\mathcal{EL}$ charakterisiert wird, betrachten die folgenden Paragraphen noch (i) die Komplexität der Subsumtion in $\mathcal{EL}$ sowie (ii) den Zusammenhang zwischen $\mathcal{EL}$-Konzeptbeschreibungen und *conjunctive queries* [AHV95] sowie *conceptual graphs* [Sow84].

Komplexität der Subsumtion in $\mathcal{EL}$

Aus der Charakterisierung der Subsumtion in Theorem 5.4 ergibt sich intuitiv folgender Entscheidungsalgorithmus für das Subsumtionsproblem $C \sqsubseteq D$ in $\mathcal{EL}$:

1. Übersetze C und D in die zugehörigen $\mathcal{EL}$-Beschreibungsbäume $\mathcal{G}(C)$ und $\mathcal{G}(D)$.

2. Entscheide, ob ein Homomorphismus von $\mathcal{G}(D)$ nach $\mathcal{G}(C)$ existiert. Falls ja, gib "ja" zurück, sonst "nein".

Offensichtlich sind $\mathcal{EL}$-Konzeptbeschreibungen in polynomieller Zeit in die zugehörigen $\mathcal{EL}$-Beschreibungsbäume übersetzbar. In [Rey77] wurde ein Algorithmus vorgestellt, der in polynomieller Zeit entscheidet, ob ein Homomorphismus zwischen zwei (nicht beschrifteten) Bäumen existiert. Abbildung 5.3 zeigt die Anpassung dieses Algorithmus an $\mathcal{EL}$-Beschreibungsbäume. Die grundlegende Idee besteht darin, eine Abbildung δ zu definieren, die jedem Knoten aus $\mathcal{G}$ eine Menge von Knoten aus $\mathcal{H}$ zuordnet, welche potentielle Urbilder dieses Knotens für einen Homomorphismus sind. Dazu wird $\mathcal{H}$ nur einmal in post order, d.h. von den Blättern zur Wurzel, traversiert, woraus folgt, dass der Algorithmus stets polynomielle Laufzeit bzgl. der Größe der Eingabe hat (s. z.B. [AHU87] für einen Algorithmus zur Ausgabe der Knoten in post order). Nun lässt sich per Induktion über die Tiefe von $\mathcal{H}$ genau dann ein Homomorphismus von $\mathcal{H}$ nach $\mathcal{G}$ definieren, wenn die Wurzel w_0 von $\mathcal{H}$ im Bild $\delta(v_0)$ der Wurzel v_0 von $\mathcal{G}$ enthalten ist.[1] Damit folgt

Satz 5.6 *Für $\mathcal{EL}$-Konzeptbeschreibungen C, D lässt sich das Subsumtionsproblem $C \sqsubseteq D$ in polynomieller Zeit (bzgl. der Größe von C und D) entscheiden.*

Ein Vergleich mit *conjunctive queries* und *conceptual graphs*

Theorem 5.4 lässt sich als Spezialfall der Charakterisierung der Subsumtion zwischen *simple conceptual graphs* [CM92] bzw. zwischen *conjunctive queries* [AHV95] auffassen. Im ersten Fall beruht die Charakterisierung auf Homomorphismen zwischen Graphen und im zweiten Fall auf Homomorphismen zwischen Mengen von Variablen und Konstanten.

[1] Ein formaler Beweis zur Vollständigkeit und Korrektheit des Algorithmus ergibt sich leicht aus den technischen Resultaten in [BMT98] zur Arbeit [BMT99]. Dort wird unter Verwendung einer Erweiterung des hier angegebenen Algorithmus gezeigt, dass auch das Subsumtionsproblem für die BL $\mathcal{ELIRO}_1$, die $\mathcal{EL}$ um *inverse Rollen, Rollenkonjunktion* und *Konstanten* erweitert, polynomiell entscheidbar ist. $\mathcal{O}$ steht hier für den aus Classic bekannten Konstruktor ONE-OF, $\mathcal{O}_1$ für die Einschränkung dieses Konstruktors auf unäre Mengen von Konstanten.

Eingabe: $\mathcal{EL}$-Beschreibungsbäume $\mathcal{H} = (V_H, E_H, w_0, \ell_H)$, $\mathcal{G} = (V_G, E_G, v_0, \ell_G)$.
Ausgabe: "ja", falls ein Homomorphismus von $\mathcal{H}$ nach $\mathcal{G}$ existiert;
"nein" sonst.

Algorithmus: $\mathsf{hom?}(\mathcal{H}, \mathcal{G})$
Sei $\{v_1, \ldots, v_n\}$ eine post-order Sortierung von V_H.
Definiere eine Abbildung $\delta : V_G \longrightarrow \mathcal{P}(V_H)$ wie folgt:
Initialisiere δ mit $\delta(v) := \emptyset$ für alle $v \in V_G$;
Für $1 \leq i \leq n$
Für $w \in V_G$
Falls $\ell_H(v_i) \subseteq \ell_G(w) \wedge$
$\forall v_i r v \in E_H : \exists w' \in V_G : (v \in \delta(w') \wedge w r w' \in E_G)$
dann $\delta(w) := \delta(w) \cup \{v_i\}$;
Falls $w_0 \in \delta(v_0)$, dann "ja", sonst "nein".

Abbildung 5.3: Entscheidungsalgorithmus zur Existenz eines Homomorphismus zwischen $\mathcal{EL}$-Beschreibungsbäumen.

Simple Conceptual Graphs. In [BMT99] wird die Beziehung zwischen Beschreibungslogiken und der Klasse sogenannter *simple conceptual graphs* (SGs) untersucht. SGs entsprechen dem konjunktiven, positiven, existentiellen Fragment der Prädikatenlogik 1. Stufe (PL1), d.h. der Menge der existentiell quantifizierten Konjunktionen von Atomen (s. [Sow84] für grundlegende Definitionen zu *conceptual graphs*). Zwischen der BL $\mathcal{ELIRO}_1$ und der Klasse von SGs, die Bäume sind, gibt es einen 1–1-Zusammenhang. Genauer gesagt können $\mathcal{ELIRO}_1$-Konzeptbeschreibungen in äquivalente SGs, die Bäume sind, übersetzt werden und umgekehrt. Dabei ist Äquivalenz über die Übersetzung von $\mathcal{ELIRO}_1$-Konzeptbeschreibungen bzw. SGs in PL1-Formeln erklärt (für die Übersetzung in PL1-Formeln siehe z.B. [Bor96] für BL-Konzeptbeschreibungen und [Sow84, BMT98] für SGs). Auf Grund dieser Äquivalenz kann das Subsumtionsproblem in $\mathcal{ELIRO}_1$ auf das Subsumtionsproblem für SGs reduziert werden.

In [CM92, MC92] charakterisieren die Autoren Subsumtion $G \sqsubseteq H$ zwischen SGs durch Homomorphismen vom Subsumierer H auf den Subsumee G, wobei G in einer bestimmten Normalform sein muss. Für beliebige SGs ist das Subsumtionsproblem NP-vollständig [CM92], für die Subklasse der Bäume aber polynomiell entscheidbar [MC92]. Folglich ist ein Algorithmus, der $\mathcal{ELIRO}_1$-Konzeptbeschreibungen C, D zunächst in äquivalente SGs G_C, G_D übersetzt und dann $G_C \sqsubseteq G_D$ entscheidet, ein polynomielles Entscheidungsverfahren für $C \sqsubseteq D$. Insbesondere ergibt sich der oben beschriebene Algorithmus für $\mathcal{EL}$ als Spezialfall.

In [CMS98] wird dargelegt, dass es auch einen 1–1-Zusammenhang zwischen SGs und *conjunctive queries* (CQs) gibt und somit Subsumtion zwischen SGs auf Subsumtion (engl. *containment*) zwischen CQs reduzieren lässt und umgekehrt. Folglich kann Subsumtion in $\mathcal{EL}$ auch als Spezialfall der Subsumtion zwischen CQs gesehen werden.

Conjunctive Queries. Eine *conjunctive query* q ist definiert als endliche Konjunktion atomarer Formeln der Form $R(x_1, \ldots, x_n)$, wobei x_i Variablen oder Konstanten sein können (s. z.B. [AHV95] für grundlegende Definitionen). Für eine gegebene Menge $\{y_1, \ldots, y_m\}$ ausgezeichneter Variablen in q und eine Interpretation $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ ist die Semantik von q definiert als $\{(\alpha_1, \ldots, \alpha_m) \in \Delta^m \mid \mathcal{I} \models q[y_1/\alpha_1, \ldots, y_m/\alpha_m]\}$, wobei freie Variablen in q existentiell quantifiziert werden.
Man sieht leicht, dass eine $\mathcal{EL}$-Konzeptbeschreibung C stets in eine äquivalente CQ q_C übersetzt werden kann, die keine Konstanten, nur unäre und binäre Prädikate sowie x_0 als einzige ausgezeichnete Variable enthält. Beispielsweise liefert die in Beispiel 5.2 eingeführte Konzeptbeschreibung C die (bei ausgezeichneter Variable x_0) äquivalente CQ

$$\begin{array}{l}P(x_0) \wedge \\ r(x_0, x_1) \wedge r(x_1, x_2) \wedge P(x_2) \wedge Q(x_2) \ \wedge s(x_1, x_3) \wedge Q(x_3) \wedge \\ r(x_0, x_4) \wedge P(x_4) \wedge s(x_4, x_5) \wedge P(x_5).\end{array}$$

In [AHV95] wird Subsumtion $p \sqsubseteq q$ zwischen CQs durch Homomorphismen zwischen den Mengen von Variablen und Konstanten in p bzw. q charakterisiert. Tatsächlich gibt es einen offensichtlichen 1–1-Zusammenhang zwischen der Einschränkung der Definition eines Homomorphismus zwischen CQs auf CQs, die $\mathcal{EL}$-Konzeptbeschreibungen entsprechen, und Homomorphismen zwischen $\mathcal{EL}$-Beschreibungsbäumen. Damit ergibt sich Theorem 5.4 auch als direkte Konsequenz aus Theorem 6.2.3 in [AHV95].

5.2.2 Charakterisierung des LCS in $\mathcal{EL}$

Wie oben erwähnt kann der LCS mit Hilfe des Produktes geeigneter Repräsentationen der Konzeptbeschreibungen charakterisiert werden. Im Falle der kleinen BL $\mathcal{EL}$ liefern bereits die $\mathcal{EL}$-Beschreibungsbäume die geeignete Repräsentation. Das Produkt zweier solcher Bäume ist nun wie folgt definiert:

Definition 5.7 (Produkt von $\mathcal{EL}$-Beschreibungsbäumen)
Seien $\mathcal{G} = (V_G, E_G, v_0, \ell_G)$ und $\mathcal{H} = (V_H, E_H, w_0, \ell_H)$ $\mathcal{EL}$-Beschreibungsbäume. Das Produkt $\mathcal{G} \times \mathcal{H} = (V, E, (v_0, w_0), \ell)$ von $\mathcal{G}$ und $\mathcal{H}$ *ist induktiv definiert durch:*

- $\ell(v_0, w_0) := \ell_C(v_0) \cap \ell_D(w_0)$,
- *zu jedem Paar (v, w) mit $v_0 r v \in E_G$ und $w_0 r w \in E_H$ ist $(v_0, w_0) r (v, w) \in E$,*
- *zu jedem direkten Nachfolger (v, w) von (v_0, w_0) enthält $\mathcal{G} \times \mathcal{H}$ das induktiv definierte Produkt $\mathcal{G}(v) \times \mathcal{H}(w)$ als Unterbaum, wobei (v, w) die Wurzel dieses Unterbaumes ist.*

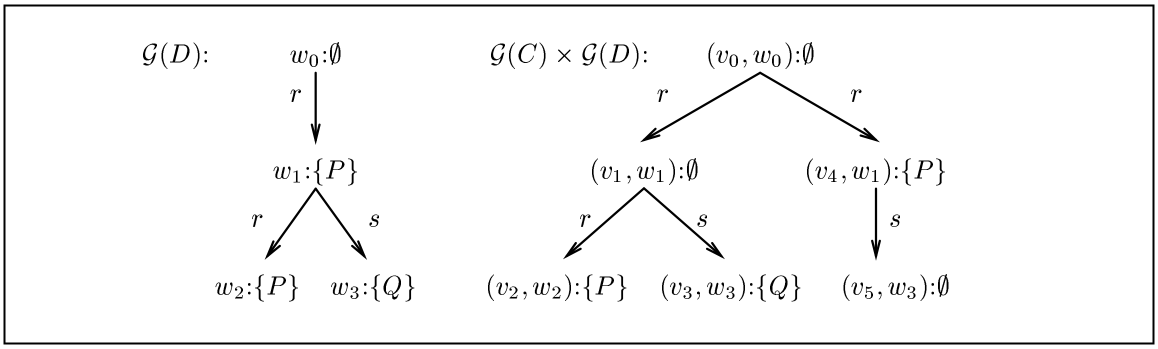


Abbildung 5.4: Ein Beispiel für das Produkt zweier $\mathcal{EL}$-Beschreibungsbäume.

Beispiel 5.8 *Abbildung 5.4 zeigt links den $\mathcal{EL}$-Beschreibungsbaum $\mathcal{G}(D)$ zur Konzeptbeschreibung $D = \exists r.(P \sqcap \exists r.P \sqcap \exists s.Q)$. Rechts ist das Produkt $\mathcal{G}(C) \times \mathcal{G}(D)$ des Baumes $\mathcal{G}(C)$ aus Abbildung 5.2 und $\mathcal{G}(D)$ abgebildet.*

Es folgt die Charakterisierung des LCS zweier $\mathcal{EL}$-Konzeptbeschreibungen, wobei von „dem“ LCS gesprochen werden kann, da er per Definition bis auf Äquivalenz eindeutig bestimmt ist.

Theorem 5.9 *Seien C, D $\mathcal{EL}$-Konzeptbeschreibungen und $\mathcal{G}(C)$, $\mathcal{G}(D)$ die zugehörigen $\mathcal{EL}$-Beschreibungsbäume. Der LCS von C und D ist gegeben durch die Konzeptbeschreibung $C_{\mathcal{G}(C)\times\mathcal{G}(D)}$ zum Produkt $\mathcal{G}(C) \times \mathcal{G}(D)$ von $\mathcal{G}(C)$ und $\mathcal{G}(D)$.*

Beweis: Seien $\mathcal{G}(C) = (V_C, E_C, v_0, \ell_C)$, $\mathcal{G}(D) = (V_D, E_D, w_0, \ell_D)$ und $\mathcal{G}(C) \times \mathcal{G}(D) = (V, E, (v_0, w_0), \ell)$. Folgende Punkte sind zu zeigen:

1. $C \sqsubseteq C_{\mathcal{G}(C)\times\mathcal{G}(D)}$,
2. $D \sqsubseteq C_{\mathcal{G}(C)\times\mathcal{G}(D)}$ und
3. für jedes C' mit $C \sqsubseteq C'$ und $D \sqsubseteq C'$ gilt $C_{\mathcal{G}(C)\times\mathcal{G}(D)} \sqsubseteq C'$.

Man sieht leicht, dass für $i \in \{1, 2\}$ die Projektion π_i mit $\pi_i(v_1, v_2) := v_i$ einen Homomorphismus von $\mathcal{G}(C) \times \mathcal{G}(D)$ nach $\mathcal{G}(C)$ (für $i = 1$) bzw. $\mathcal{G}(D)$ (für $i = 2$) liefert. Mit der Rückrichtung von Theorem 5.4 und Eigenschaft (5.1) folgt $C \sqsubseteq C_{\mathcal{G}(C)\times\mathcal{G}(D)}$ bzw. $D \sqsubseteq C_{\mathcal{G}(C)\times\mathcal{G}(D)}$.
Es bleibt 3. zu zeigen. Sei C' ein beliebiger Common Subsumer von C und D und $\mathcal{G}(C') = (V', E', v_0', \ell')$ der zugehörige $\mathcal{EL}$-Beschreibungsbaum. Gemäß Theorem 5.4 existiert ein Homomorphismus φ_1 von $\mathcal{G}(C')$ nach $\mathcal{G}(C)$ und ein Homomorphismus φ_2 von $\mathcal{G}(C')$ nach $\mathcal{G}(D)$.

Behauptung: Das Produkt $\varphi := \langle \varphi_1, \varphi_2 \rangle$ von φ_1 und φ_2 definiert durch $\varphi(v') := (\varphi_1(v'), \varphi_2(v'))$ für $v' \in V'$ ist ein Homomorphismus von $\mathcal{G}(C')$ nach $\mathcal{G}(C) \times \mathcal{G}(D)$.

Dann folgt wiederum mit Theorem 5.4 und Eigenschaft (5.2) $C_{\mathcal{G}(C)\times\mathcal{G}(D)} \sqsubseteq C'$, was den Beweis abschließt.

Zum Beweis der Behauptung ist zu zeigen, dass (i) φ wohl-definiert ist, d.h. für alle $v' \in V'$ gilt $\varphi(v') \in V$; und (ii) φ die Bedingungen 1.–3. aus Definition 5.3 erfüllt.

Der Punkt (i) kann durch Induktion nach der Länge $\lambda(v')$ des eindeutigen (!) Pfades von v_0' nach v' in $\mathcal{G}(C')$ gezeigt werden.

Für $\lambda(v') = 0$ erhält man $v' = v_0'$, also $\varphi(v') = \varphi(v_0') = (\varphi_1(v_0'), \varphi_2(v_0')) = (v_0, w_0) \in V$. Für $\lambda(v') > 0$ existiert ein eindeutiger (!) Vorgänger v'' in $\mathcal{G}(C')$ mit $v''rv' \in E'$ für ein $r \in N_R$. Es ist $\lambda(v'') < \lambda(v')$ und per Induktion folgt $\varphi(v'') = (\varphi_1(v''), \varphi_2(v'')) \in V$. Da φ_1, φ_2 Homomorphismen sind, ist $\varphi_1(v'')r\varphi_1(v') \in E_C$ und $\varphi_2(v'')r\varphi_2(v') \in E_D$. Per Definition des Produktes von $\mathcal{EL}$-Beschreibungsbäumen ist $(\varphi_1(v'), \varphi_2(v'))$ ein r-Nachfolger von $(\varphi_1(v''), \varphi_2(v''))$ in $\mathcal{G}(C) \times \mathcal{G}(D)$, also $(\varphi_1(v'), \varphi_2(v')) \in V$.

Zum Beweis von Punkt (ii) werden die Bedingungen aus Definition 5.3 betrachtet:

1. Da φ_1, φ_2 Homomorphismen sind, ist $\varphi_1(v_0') = v_0$ und $\varphi_2(v_0') = w_0$, also $\varphi(v_0') = (v_0, w_0)$.

2. Aus der Definition des Produktes von $\mathcal{EL}$-Beschreibungsbäumen folgt $\ell(v, w) = \ell_C(v) \cap \ell_D(w)$ für alle Knoten $(v, w) \in V$. Außerdem gilt $\ell'(v') \subseteq \ell_C(\varphi_1(v'))$ und $\ell'(v') \subseteq \ell_D(\varphi_2(v'))$. Damit ist $\ell'(v') \subseteq \ell_C(\varphi_1(v')) \cap \ell_D(\varphi_2(v')) = \ell(\varphi_1(v'), \varphi_2(v'))$.

3. Sei $v'rw' \in E'$. Da φ_1, φ_2 Homomorphismen sind, folgt $\varphi_1(v')r\varphi_1(w') \in E_C$ und $\varphi_2(v')r\varphi_2(w') \in E_D$. Mit (i) folgt $(\varphi_1(v'), \varphi_2(v')) \in V$ und mit Definition 5.7 folgt $(\varphi_1(v'), \varphi_2(v'))r(\varphi_1(w'), \varphi_2(w')) \in E$. □

Mit Theorem 5.9 folgt für die Konzeptbeschreibungen C aus Beispiel 5.2 und D aus Beispiel 5.8, dass der LCS von C und D gegeben ist durch

$$C_{\mathcal{G}(C)\times\mathcal{G}(D)} = \exists r.(\exists r.P \sqcap \exists s.Q) \sqcap \exists r.(P \sqcap \exists s.\top).$$

Die *Größe* eines $\mathcal{EL}$-Beschreibungsbaumes $\mathcal{G} = (V, E, v_0, \ell)$ ist definiert als die Summe der Anzahl der Kanten und der Größen der Knotenlabel, d.h.

$$|\mathcal{G}| := |E| + \sum_{v \in V} |\ell(v)|.$$

Nach Definition von $\mathcal{G}(C)$ und der Definition der Größe $|C|$ (s. Definition 3.17) ist die Größe des $\mathcal{EL}$-Beschreibungsbaumes $\mathcal{G}(C)$ stets kleiner oder gleich der Größe von C, d.h.

$$|\mathcal{G}(C)| \leq |C|. \tag{5.3}$$

Die Größe von $\mathcal{G}(C)$ kann kleiner sein als die Größe von C, da Mehrfachvorkommen atomarer Konzeptnamen in einer Konjunktion in $|\mathcal{G}(C)|$ nicht mitgezählt, wohl aber in $|C|$. Umgekehrt ist die Größe der $\mathcal{EL}$-Konzeptbeschreibung $C_\mathcal{G}$ stets gleich der Größe des $\mathcal{EL}$-Beschreibungsbaumes $\mathcal{G}$, d.h.

$$|C_\mathcal{G}| = |\mathcal{G}|. \tag{5.4}$$

Da die Größe des Produktes zweier Bäume offensichtlich durch das Produkt der Größen der Eingabebäume beschränkt werden kann, erhält man

Satz 5.10 *Die Größe der LCS zweier $\mathcal{EL}$-Konzeptbeschreibungen C, D ist polynomiell in der Größe der Konzeptbeschreibungen C und D und der LCS kann in polynomieller Zeit berechnet werden.*

In den in Abschnitt 3.4 genannten Anwendungen des LCS ist man aber im allgemeinen an der Berechnung des LCS von $n > 2$ Konzeptbeschreibungen $C_1, \ldots, C_n$ interessiert. Dieser LCS ergibt sich aus dem Produkt $\mathcal{G}(C_1) \times \cdots \times \mathcal{G}(C_n)$ der zugehörigen Beschreibungsbäume. Dieses kann wie üblich durch iterierte Berechnung des binären Produktes bestimmt werden. Leider zeigt nun das folgende Beispiel, dass bereits für die ausdrucksschwache BL $\mathcal{EL}$ die Größe dieses LCS *nicht* polynomiell beschränkt werden kann.

Beispiel 5.11 *Für $n \geq 0$ definiere*

$$D_n := \begin{cases} \top, & n = 0 \\ \exists r.(P \sqcap Q \sqcap D_{n-1}), & n > 0 \end{cases}$$

und für $n \geq 1$ und $1 \leq i \leq n$ definiere

$$C_i^n := \begin{cases} \exists r.(P \sqcap D_{n-1}) \sqcap \exists r.(Q \sqcap D_{n-1}), & i = 1 \\ \exists r.(P \sqcap Q \sqcap C_{i-1}^{n-1}), & 1 < i \leq n. \end{cases}$$

Die Idee hinter der Definition der Konzepte D_n und C_i^n ist die folgende: Der Beschreibungsbaum $\mathcal{G}(D_n)$ zu D_n ist ein einzelner Pfad der Länge n, wobei die Wurzel mit der leeren Menge und alle anderen Knoten mit der Menge $\{P, Q\}$ beschriftet sind. Der Beschreibungsbaum $\mathcal{G}(C_i^n)$ zu C_i^n ist ein Baum der Tiefe n der folgenden Form: von der Wurzel führt ein einzelner Pfad der Länge $i - 1$ zu einem Knoten, der genau zwei r-Nachfolger v_1, v_2 hat, wobei v_1 mit $\{P\}$ beschriftet ist und v_2 mit $\{Q\}$ und beide Wurzeln eines einzelnen Pfades der Länge $n - i$ sind, dessen Knoten alle mit $\{P, Q\}$ beschriftet sind. Alle Kanten in diesem Baum sind mit r beschriftet. Abbildung 5.5 zeigt die entsprechenden Bäume für $n = 3$. Das Produkt der Bäume $\mathcal{G}(C_1^n), \ldots, \mathcal{G}(C_n^n)$ bildet dann einen binären Baum der Tiefe n, hat also eine Größe exponentiell in n.

Lemma 5.12 *Für $n \in \mathbb{N}$ sei C_i^n wie in Beispiel 5.11 definiert. Die Größe von C_i^n ist linear in n und die Größe des LCS von $C_1^n, \ldots, C_n^n$ ist exponentiell in n.*

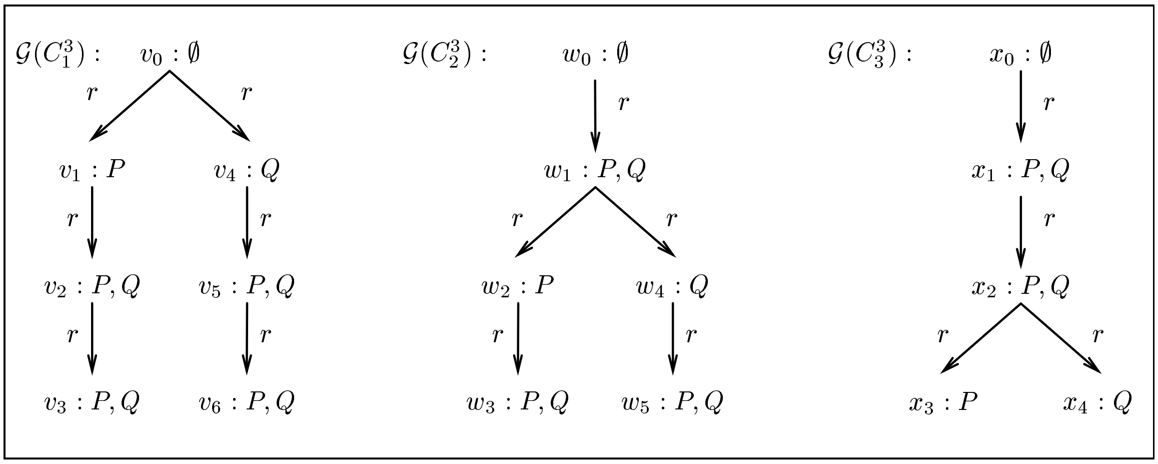


Abbildung 5.5: Die $\mathcal{EL}$-Beschreibungsbäume zu C_1^3, C_2^3 und C_3^3.

Beweis: Offensichtlich kann die Größe von C_i^n linear beschränkt werden. Nach Theorem 5.9 ergibt sich der LCS von $C_1^n, \ldots, C_n^n$ aus dem Produkt der Bäume $\mathcal{G}(C_1^n), \ldots,$ $\mathcal{G}(C_n^n)$. Man sieht leicht, dass das Produkt $\mathcal{G}(C_1^n) \times \cdots \times \mathcal{G}(C_n^n)$ isomorph ist zum vollen binären Baum $\mathcal{B}_n$ der Tiefe n, wobei

- jede Kante mit r und die Wurzel mit der leeren Menge beschriftet ist und
- jeder andere Knoten genau zwei Nachfolger hat, einer mit $\{P\}$ und der andere mit $\{Q\}$ beschriftet.

Als Beispiel betrachte man den Baum zu $\mathcal{G}(C_1^3) \times \mathcal{G}(C_2^3) \times \mathcal{G}(C_3^3)$ in Abbildung 5.6. Es ist also $\mathsf{lcs}(C_1^n, \ldots, C_n^n) \equiv C_{\mathcal{B}_n}$. Da die Größe von $\mathcal{B}_n$ (und damit auch von $C_{\mathcal{B}_n}$) exponentiell in n ist, bleibt zu zeigen, dass es keine $\mathcal{EL}$-Konzeptbeschreibung C gibt mit $C \equiv C_{\mathcal{B}_n}$ und $|C| < |C_{\mathcal{B}_n}|$, wobei mit Eigenschaft (5.3) und (5.4) o.B.d.A. $|C| = |\mathcal{G}(C)|$ sei.

Angenommen, $C \equiv C_{\mathcal{B}_n}$ und $|C| \leq |C_{\mathcal{B}_n}|$. Mit Theorem 5.4 und Eigenschaft (5.2) folgt aus $C \sqsubseteq C_{\mathcal{B}_n}$, dass ein Homomorphismus φ von $\mathcal{B}_n = (V, E, v_0, \ell)$ nach $\mathcal{G}(C) = (V_C, E_C, w_0, \ell_C)$ existiert. Wir zeigen, dass φ injektiv ist. Angenommen, es existieren Knoten $v_1 \neq v_2$ in $\mathcal{B}_n$ mit $\varphi(v_1) = \varphi(v_2)$. Dann können v_1 und v_2 insbesondere so gewählt werden, dass sie einen gemeinsamen direkten Vorgänger v haben, d.h. $\{vrv_1, vrv_2\} \subseteq E$. Da φ Homomorphismus ist, folgt $\ell(v_1) \cup \ell(v_2) = \{P, Q\} \subseteq \ell_C(\varphi(v_1))$. Sei die Länge des Pfades von w_0 nach $\varphi(v)$ in $\mathcal{G}(C)$ gleich $i - 1$. Dann gibt es keinen Homomorphismus von $\mathcal{G}(C)$ nach $\mathcal{G}(C_i^n)$: Durch die Definition von C_i^n ist der Knoten v_i^n in $\mathcal{G}(C_i^n)$, auf den $\varphi(v)$ abgebildet wird, eindeutig bestimmt. Wiederum nach Definition von C_i^n hat v_i^n genau zwei r-Nachfolger v' und v'', die mit P bzw. Q beschriftet sind, wobei $\ell_C(\varphi(v_1)) \not\subseteq \ell_i^n(v')$ und $\ell_C(\varphi(v_1)) \not\subseteq \ell_i^n(v'')$. Also kann $\mathcal{G}(C)$ nicht homomorph auf $\mathcal{G}(C_i^n)$ abgebildet werden, was wegen $C_{\mathcal{B}_n} \sqsupseteq C_i^n$ im Widerspruch zur Annahme $C \equiv C_{\mathcal{B}_n}$ steht.

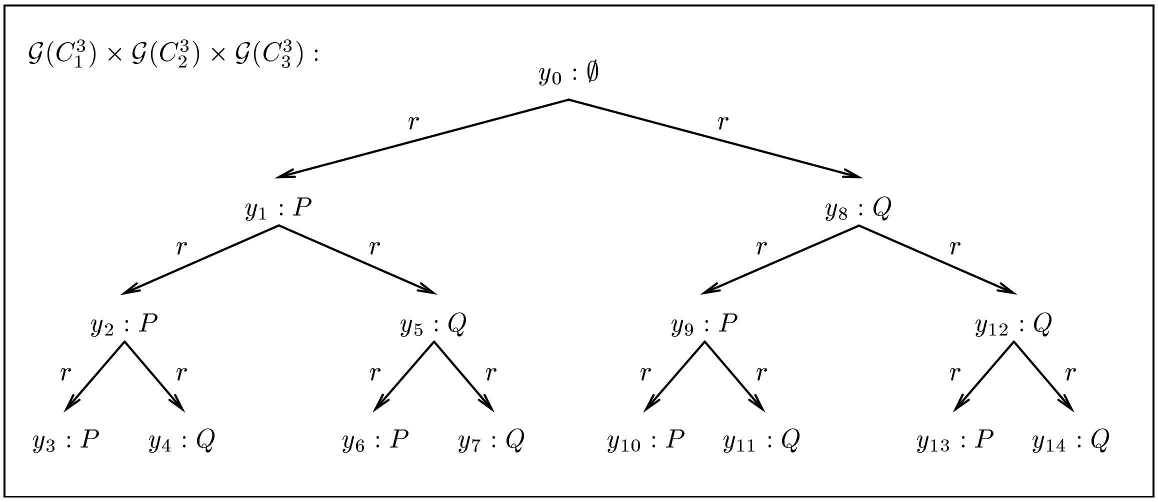


Abbildung 5.6: Das Produkt der $\mathcal{EL}$-Beschreibungsbäume zu C_1^3, C_2^3 und C_3^3.

Aus der Existenz eines injektiven Homomorphismus von $\mathcal{B}_n$ nach $\mathcal{G}(C)$ folgt $|\mathcal{B}_n| \leq |\mathcal{G}(C)|$ und mit Eigenschaft (5.4) $|C_{\mathcal{B}_n}| \leq |C|$. □

Mit Lemma 5.12 folgt unmittelbar

Satz 5.13 *Die Größe des LCS von n $\mathcal{EL}$-Konzeptbeschreibungen, deren Größe jeweils linear in n ist, kann exponentiell in n wachsen.*

5.3 Least Common Subsumer in $\mathcal{ALE}$

Dieser Abschnitt erweitert die Ergebnisse zum LCS in $\mathcal{EL}$ auf $\mathcal{ALE}$. Dazu werden zunächst die Definitionen *Beschreibungsbaum* und *Homomorphismus* an die $\mathcal{ALE}$-Syntax angepasst. Diese Anpassung alleine reicht allerdings nicht aus, um eine korrekte und vollständige Charakterisierung der Subsumtion und die gewünschte Charakterisierung des LCS zu erhalten. Probleme ergeben sich im wesentlichen aus der Interaktion von Existenz- und Werterestriktionen sowie aus Inkonsistenzen. Die im folgenden präsentierte Lösung basiert auf der Idee, die Konzeptbeschreibungen zunächst in eine geeignete Normalform zu bringen und erst dann in Beschreibungsbäume zu übersetzen. Dabei können die zu betrachtenden Normalformen zwar exponentiell groß bezüglich der Größe der Ausgangskonzepte sein. Allerdings wird sich zeigen, dass sich ein solcher exponentieller Zuwachs bei der Berechnung des LCS zweier $\mathcal{ALE}$-Konzeptbeschreibungen im allgemeinen nicht vermeiden lässt.

5.3.1 Charakterisierung der Subsumtion in $\mathcal{ALE}$

Die Definition von $\mathcal{ALE}$-Beschreibungsbäumen ergibt sich auf natürliche Weise durch eine Erweiterung der Definition von $\mathcal{EL}$-Beschreibungsbäumen um Kantenlabel der

Form $\forall r$ und Knotenlabel, die neben Konzeptnamen auch negierte Konzeptnamen und $\bot$ enthalten dürfen.

Definition 5.14 ($\mathcal{ALE}$-Beschreibungsbäume) *Sei N_C eine Menge von Konzeptnamen und N_R eine Menge von Rollennamen. Ein $\mathcal{ALE}$-*Beschreibungsbaum *ist ein beschrifteter Baum der Form $\mathcal{G} = (V, E, v_0, \ell)$ mit*

- *Wurzel $v_0 \in V$,*
- *$E \subseteq V \times N_R \times V \cup V \times \{\forall r \mid r \in N_R\} \times V$ und*
- *einer Beschriftungsfunktion ℓ, die jedem Knoten $v \in V$ eine Menge $\ell(v) \subseteq N_C \cup \{\neg P \mid P \in N_C\} \cup \{\bot\}$ zuordnet, wobei das leere Label $\top$ entspricht.*

Für $v \in V$ bezeichnet $\mathcal{G}(v)$ den Unterbaum *von $\mathcal{G}$ mit Wurzel v.*

Analog zur Notation für $\exists$-*Kanten* wird $v\forall rw \in E$ als $\forall$-*Kante* und w als $\forall r$-*Nachfolger* bzw. *direkter Nachfolger von v* bezeichnet. Wie für $\mathcal{EL}$ ist die Übersetzung von Konzeptbeschreibungen in Beschreibungsbäume und umgekehrt induktiv erklärt. Dabei gehen wir wieder o.B.d.A. davon aus, dass $\mathcal{ALE}$-Konzeptbeschreibungen stets in folgender Normalform vorliegen:

$$C = Q_1 \sqcap \ldots \sqcap Q_n \sqcap \forall s_1.C'_1 \sqcap \ldots \sqcap \forall s_k.C'_k \sqcap \exists r_1.C_1 \sqcap \ldots \sqcap \exists r_m.C_m,$$

wobei $Q_i \in N_C \cup \{\neg P \mid P \in N_C\} \cup \{\top, \bot\}$ und die C_i und C'_j ebenfalls in dieser Normalform sind. Eine solche Konzeptbeschreibung wird wie folgt in einen $\mathcal{ALE}$-Beschreibungsbaum $\mathcal{G}(C) = (V, E, v_0, \ell)$ übersetzt:

Für *depth*$(C) = 0$ definiere $V := \{v_0\}$, $E := \emptyset$ und $\ell(v_0) := \{Q_1, \ldots, Q_n\} \setminus \{\top\}$.

Für *depth*$(C) > 0$ seien $\mathcal{G}(C_i) = (V_i, E_i, v_{0i}, \ell_i)$ und $\mathcal{G}(C'_j) = (V'_j, E'_j, v'_{0j}, \ell'_j)$ die induktiv definierten Beschreibungsbäume zu C_i, $1 \le i \le m$, und C'_j, $1 \le j \le k$. Wiederum seien die V_i, V'_j und $\{v_0\}$ o.B.d.A. paarweise disjunkt. Definiere

- $V := \{v_0\} \cup \bigcup_{1 \le i \le m} V_i \cup \bigcup_{1 \le j \le k} V'_j$,
- $E := \{v_0 r_i v_{0i} \mid 1 \le i \le m\} \cup \{v_0 \forall s_j v'_{0j} \mid 1 \le j \le k\} \cup \bigcup_{1 \le i \le m} E_i \cup \bigcup_{1 \le j \le k} E'_j$ und
- $\ell(v) := \begin{cases} \{Q_1, \ldots, Q_n\} \setminus \{\top\}, & v = v_0 \\ \ell_i(v), & v \in V_i \text{ für ein } i \in \{1, \ldots, m\} \\ \ell'_j(v), & v \in V'_j \text{ für ein } j \in \{1, \ldots, k\} \end{cases}$

Die Übersetzung eines $\mathcal{ALE}$-Beschreibungsbaumes $\mathcal{G} = (V, E, v_0, \ell)$ in eine $\mathcal{ALE}$-Konzeptbeschreibung $C_{\mathcal{G}}$ erfolgt ebenfalls induktiv:

Für $depth(\mathcal{G}) = 0$ definiere

$$C_\mathcal{G} := \begin{cases} \top & \text{, falls } \ell(v_0) = \emptyset \\ Q_1 \sqcap \ldots \sqcap Q_n & \text{, falls } \ell(v_0) = \{Q_1, \ldots, Q_n\}, n \geq 1. \end{cases}$$

Für $depth(\mathcal{G}) > 0$ sei $\ell(v_0) = \{Q_1, \ldots, Q_n\}$, $\{v_1, \ldots, v_m\}$ die Menge aller direkten Nachfolger von v_0 mit $v_0 r_i v_i \in E$, $1 \leq i \leq m$, und $\{v'_1, \ldots, v'_k\}$ die Menge aller direkten Nachfolger von v_0 mit $v_0 \forall s_j v'_j \in E$, $1 \leq j \leq k$. Für $1 \leq i \leq m$ $(1 \leq j \leq k)$ sei C_i (C'_j) die zum Unterbaum von $\mathcal{G}$ mit Wurzel v_i (v'_j) induktiv definierte Konzeptbeschreibung. Definiere $C_\mathcal{G} := Q_1 \sqcap \ldots \sqcap Q_n \sqcap \exists r_1.C_1 \sqcap \ldots \sqcap \exists r_m.C_m \sqcap \forall s_j.C'_j \sqcap \ldots \sqcap \forall s_k.C'_k$.

Wie für $\mathcal{EL}$ ist auch für $\mathcal{ALE}$-Beschreibungsbäume $\mathcal{G}$ die *Semantik* bzgl. einer Interpretation $\mathcal{I}$ durch die Semantik der zugehörigen $\mathcal{ALE}$-Konzeptbeschreibung $C_\mathcal{G}$ definiert und es gilt

$$C \equiv C_{\mathcal{G}(C)} \text{ und } \mathcal{G} \equiv \mathcal{G}(C_\mathcal{G}). \tag{5.5}$$

Die Erweiterung des Begriffes Homomorphismus auf $\mathcal{ALE}$-Beschreibungsbäume erfordert offensichtlich die Berücksichtigung von Kanten mit Label der Form $\forall r$, d.h. die 3. Bedingung in Definition 5.3 muss ersetzt werden durch

3.' $\varphi(v) r \varphi(w) \in E_G$ für alle $vrw \in E_H$ und $\varphi(v) \forall r \varphi(w) \in E_G$ für alle $v \forall rw \in E_H$.

Diese einfache Erweiterung reicht aber nicht aus, um analog zu Theorem 5.4 eine korrekte und *vollständige* Charakterisierung der Subsumtion in $\mathcal{ALE}$ zu erhalten. Dies verdeutlichen die folgenden Beispiele.

Beispiel 5.15 *Betrachte die $\mathcal{ALE}$-Konzeptbeschreibungen $C = P \sqcap Q$ und $D = P \sqcap \forall r.\top$. Es gilt $\forall r.\top \equiv \top$ und damit $D \equiv P$. Insbesondere gilt also $C \sqsubseteq D$. Außerdem ist*

$$\begin{array}{lcll} \mathcal{G}(C) & = & (\{v_0\}, \emptyset, v_0, \ell_C) & \textit{mit } \ell_C(v_0) = \{P, Q\} \textit{ und} \\ \mathcal{G}(D) & = & (\{w_0, w_1\}, \{w_0 \forall r w_1\}, w_0, \ell_D) & \textit{mit } \ell_D(w_0) = \{P\} \textit{ und } \ell_D(w_1) = \emptyset. \end{array}$$

Offensichtlich existiert wegen $E_C = \emptyset$ und $E_D \neq \emptyset$ kein Homomorphismus von $\mathcal{G}_D$ nach $\mathcal{G}_C$ (der den Bedingungen 1 und 2 aus Definition 5.3 und obiger Bedingung 3′ genügt).

Die Ursache für das Problem im obigen Beispiel liegt darin, dass die Kante $w_0 \forall r w_1$ in $\mathcal{G}_D$ in dem Sinne redundant ist, dass sie eine Werterestriktion repräsentiert, die äquivalent zu $\top$ ist. Zur Lösung dieses Problems sind derartige redundante Kanten aus dem Beschreibungsbaum des Subsumierers zu eliminieren. Dies könnte zum einen nachträglich geschehen, d.h. nach der Übersetzung des möglichen Subsumierers D in $\mathcal{G}(D)$. Hier wird aber der alternative Ansatz verfolgt, die Konzeptbeschreibungen D

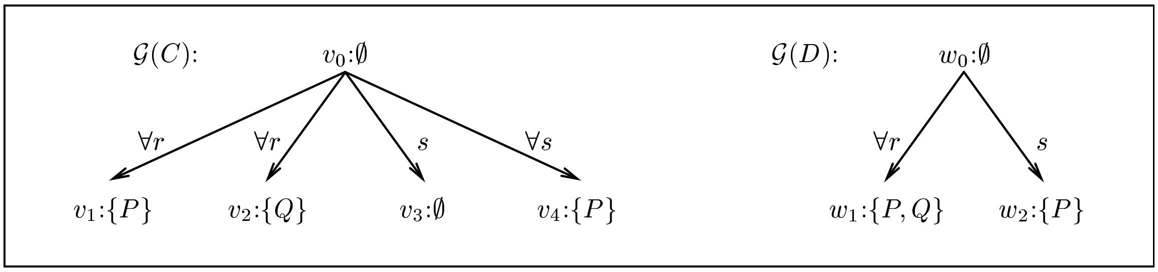


Abbildung 5.7: $\mathcal{ALE}$-Beschreibungsbäume: Interaktionen zwischen Werte- und Existenzrestriktionen.

vor der Übersetzung durch Anwenden äquivalenzerhaltender Normalisierungsregeln in eine geeignete Normalform zu bringen. Hierbei bedeutet „geeignet“, dass der zur Normalform erhaltene Beschreibungsbaum keine solchen redundanten Kanten mehr enthält. Für das in Beispiel 5.15 skizzierte Problem leisten die Regeln

$$\begin{aligned} \forall r.\top &\longrightarrow \top \\ E \sqcap \top &\longrightarrow E \end{aligned}$$

das Gewünschte, wobei die zweite Regel modulo Kommutativität und Assoziativität der Konjunktion zu lesen ist, d.h. auch $\top \sqcap E$ wird zu E normalisiert. Die erschöpfende Anwendung dieser Regeln auf eine $\mathcal{ALE}$-Konzeptbeschreibung D liefert eine äquivalente $\mathcal{ALE}$-Konzeptbeschreibung, die sogenannte $\top$-*Normalform* von D.
Um eine vollständige Charakterisierung der Subsumtion zu erhalten, muss aber nicht nur der Subsumierer, sondern auch der Subsumee geeignet normalisiert werden.

Beispiel 5.16 *Abbildung 5.7 zeigt die Beschreibungsbäume zu den Konzeptbeschreibungen*

$$\begin{aligned} C &= \forall r.P \sqcap \forall r.Q \sqcap \exists s.\top \sqcap \forall s.P \text{ und} \\ D &= \forall r.(P \sqcap Q) \sqcap \exists s.P. \end{aligned}$$

Es gilt $C \sqsubseteq D$, aber offensichtlich existiert kein Homomorphismus von $\mathcal{G}(D)$ nach $\mathcal{G}(C)$:

- *w_1 muss wegen des Kantenlabels auf v_1 oder v_2 abgebildet werden, aber für $i = 1, 2$ gilt $\ell_D(w_1) \not\subseteq \ell_C(v_i)$;*
- *w_2 muss wegen des Kantenlabels auf v_3 abgebildet werden, aber es gilt $\ell_D(w_2) \not\subseteq \ell_C(v_3)$.*

Die Ursache für das in Beispiel 5.16 aufgezeigte Problem liegt in der Interaktion zwischen Werte- und Existenzrestriktionen in $\mathcal{ALE}$-Konzeptbeschreibungen sowie Wertrestriktionen untereinander: treten zu einem Rollennamen r in einer Konjunktion

sowohl Werte- als auch Existenzrestriktionen auf, so gelten alle Konzeptbeschreibungen in den Werterestriktionen implizit auch in den Existenzrestriktionen. Im Beispiel ist jeder s-Nachfolger einer Instanz von C auch Instanz von P. Um eine vollständige Charakterisierung der Subsumtion zu erhalten, sind diese impliziten Subsumtionsbeziehungen explizit zu machen. Desweiteren zeigt das Beispiel, dass zusätzlich alle Werterestriktionen zu einem Rollennamen r in *einer* Werterestriktion der Form $\forall r.C'$ zusammenzufassen sind.
Formal drücken die folgenden Äquivalenzen diese Zusammenhänge aus:

$$\begin{aligned} \forall r.E \sqcap \forall r.F &\equiv \forall r.(E \sqcap F); \\ \forall r.E \sqcap \exists r.F &\equiv \forall r.E \sqcap \exists r.(E \sqcap F). \end{aligned}$$

Liest man diese Äquivalenzen wiederum von links nach rechts als Termersetzungsregeln (modulo Kommutativität und Assoziativität der Konjunktion), so liefert die erschöpfende Anwendung dieser Regeln eine äquivalente Konzeptbeschreibung, wobei die zweite Regel auf jedes Paar von Existenz- und Werterestriktion nur einmal angewendet werden darf. Die erzeugte Konzeptbeschreibung enthält zum einen in jeder Konjunktion zu jedem Rollennamen höchstens eine Werterestriktion, d.h. alle Werterestriktionen zu einem Rollennamen wurden in einer Werterestriktion zusammengefasst. Zum anderen wurden alle impliziten Folgerungen, die sich aus Konjunktionen von Werte- und Existenzrestriktionen ergeben, explizit gemacht.
Ein letztes Problem bei der Charakterisierung der Subsumtion in $\mathcal{ALE}$ ergibt sich aus Inkonsistenzen, die in $\mathcal{ALE}$-Konzeptbeschreibungen ausgedrückt werden können. Ausgehend von den Äquivalenzen

$$\begin{aligned} P \sqcap \neg P &\equiv \bot \\ \exists r.\bot &\equiv \bot \\ E \sqcap \bot &\equiv \bot \end{aligned}$$

sind derartige Inkonsistenzen im Subsumee wieder durch geeignete Normalisierungsregeln explizit zu machen. Zusätzlich ist zur Lösung dieses Problems allerdings auch noch die Definition des Homomorphismus geeignet zu modifizieren.

Beispiel 5.17 *Man sieht leicht, dass die $\mathcal{ALE}$-Konzeptbeschreibung*

$$C = \forall r.\bot \sqcap \exists s.(P \sqcap \exists r.Q)$$

von der $\mathcal{ALE}$-Konzeptbeschreibung

$$D = \forall r.(\exists r.P \sqcap \exists r.\neg P) \sqcap \exists s.\exists r.Q$$

subsumiert wird. Betrachtet man allerdings die zugehörigen $\mathcal{ALE}$-Beschreibungsbäume (s. Abbildung 5.8), so erkennt man, dass keine Abbildung von $\mathcal{G}(D)$ auf $\mathcal{G}(C)$ existiert, die die Bedingungen 1 und 2 aus Definition 5.3 und obige Bedingung $3'$ erfüllt.

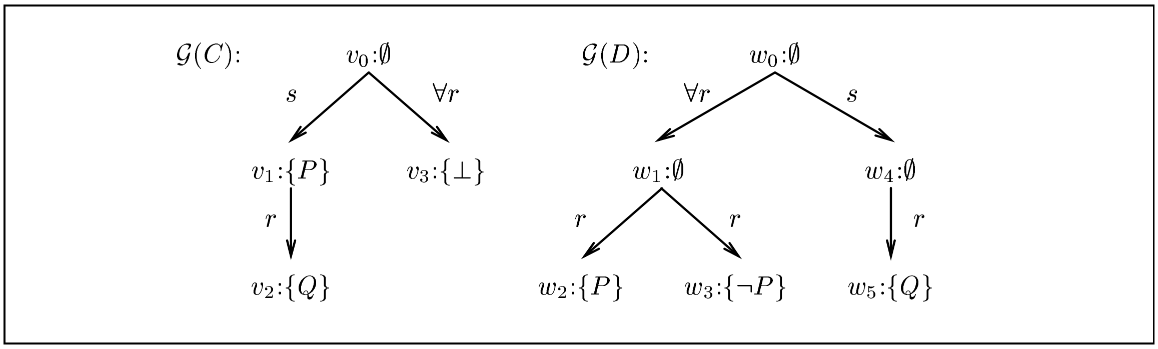


Abbildung 5.8: $\mathcal{ALE}$-Beschreibungsbäume mit Inkonsistenzen.

Die folgende Definition fasst die Normalisierungsregeln für $\mathcal{ALE}$-Konzeptbeschreibungen zusammen und führt die zugehörigen Normalformen ein.

Definition 5.18 ($\mathcal{ALE}$-Normalisierungsregeln und -Normalformen) *Seien E und F zwei $\mathcal{ALE}$-Konzeptbeschreibungen, $P \in N_C$ ein Konzeptname und $r \in N_R$ ein Rollenname. Die $\mathcal{ALE}$-*Normalisierungsregeln *sind wie folgt definiert:*

$$\begin{aligned}
\forall r.\top &\longrightarrow \top && (5.6)\\
E \sqcap \top &\longrightarrow E && (5.7)\\
P \sqcap \neg P &\longrightarrow \bot && (5.8)\\
\exists r.\bot &\longrightarrow \bot && (5.9)\\
E \sqcap \bot &\longrightarrow \bot && (5.10)\\
\forall r.E \sqcap \forall r.F &\longrightarrow \forall r.(E \sqcap F) && (5.11)\\
\forall r.E \sqcap \exists r.F &\longrightarrow \forall r.E \sqcap \exists r.(E \sqcap F), \textit{ falls } F \not\sqsubseteq E. && (5.12)
\end{aligned}$$

*Eine $\mathcal{ALE}$-Konzeptbeschreibung ist $\mathcal{ALE}$-*normalisiert*, wenn keine der Regeln auf C anwendbar ist; C ist* $\top$-normalisiert*, wenn die Regeln (5.6) und (5.7) nicht auf C anwendbar sind. Die $\mathcal{ALE}$-*Normalform *zu C ist definiert als die Konzeptbeschreibung, die man aus C durch erschöpfende Anwendung aller Regeln erhält. Die* $\top$-Normalform *zu C ist definiert als die Konzeptbeschreibung, die man aus C durch erschöpfende Anwendung der Regeln (5.6) und (5.7) erhält. Für $\mathcal{ALE}$- und $\top$-Normalform sind die Regeln jeweils modulo Kommutativität und Assoziativität der Konjunktion anzuwenden.*

*Der $\mathcal{ALE}$-Beschreibungsbaum zur $\mathcal{ALE}$-Normalform von C wird mit $\mathcal{G}_C$ bezeichnet und der $\mathcal{ALE}$-Beschreibungsbaum zur $\top$-Normalform von C mit $\mathcal{G}_C^\top$. Ein $\mathcal{ALE}$-Beschreibungsbaum $\mathcal{G}$ heißt $\mathcal{ALE}$-*normalisiert *bzw.* $\top$-normalisiert*, wenn $C_\mathcal{G}$ $\mathcal{ALE}$-normalisiert bzw. $\top$-normalisiert ist.*

Die Definition der Normalformen macht Sinn, da durch die Einschränkung der Anwendung von Regel (5.12) die erschöpfende Anwendung der Regeln zum einen stets nach

endlich vielen Schritten terminiert und zum anderen ein eindeutiges Ergebnis liefert, d.h. die $\mathcal{ALE}$-Normalform zu einer $\mathcal{ALE}$-Konzeptbeschreibung ist eindeutig (modulo Kommutativität und Assoziativität der Konjunktion). Desweiteren sind die Regeln äquivalenzerhaltend, sodass sowohl $\mathcal{ALE}$- als auch $\top$-Normalform von C äquivalent zu C sind.
Wie man leicht sieht, ist keine der Normalisierungsregeln auf die Konzepteschreibungen C, D aus Beispiel 5.17 anwendbar ist, d.h. C und D sind in $\mathcal{ALE}$-Normalform. Dass dennoch kein Homomorphismus von $\mathcal{G}(D)$ nach $\mathcal{G}(C)$ existiert, obwohl $C \sqsubseteq D$ gilt, liegt daran, dass die Bedingungen $1, 2$ und $3'$ die für alle $\mathcal{ALE}$-Konzeptbeschreibungen E gültige Äquivalenz $\bot \sqsubseteq E$ nicht berücksichtigen. Intuitiv entspricht dem Zusammenhang $\bot \sqsubseteq E$ auf Seiten der Beschreibungsbäume eine Abbildung, die den Baum zu E ganz auf den einzelnen Knoten, der $\bot$ repräsentiert, abbildet. Formal können Inkonsistenzen in $\mathcal{ALE}$-Beschreibungsbäumen durch folgenden Homomorphiebegriff geeignet behandelt werden:

Definition 5.19 (Homomorphismen zwischen $\mathcal{ALE}$-Beschreibungsbäumen)
Seien $\mathcal{G} = (V_G, E_G, v_0, \ell_G)$ und $\mathcal{H} = (V_H, E_H, w_0, \ell_H)$ $\mathcal{ALE}$-Beschreibungsbäume. Eine Abbildung $\varphi : V_H \longrightarrow V_G$ ist ein Homomorphismus *von $\mathcal{H}$ nach $\mathcal{G}$ genau dann, wenn die folgenden Bedingungen erfüllt sind:*

1. $\varphi(w_0) = v_0$,
2. *für alle $v \in V_H$ gilt $\ell_H(v) \subseteq \ell_G(\varphi(v))$ oder $\bot \in \ell_G(\varphi(v))$,*
3. *für alle $v\rho w \in E_H$ mit $\rho = r$ oder $\rho = \forall r$ für ein $r \in N_R$ gilt $\varphi(v)\rho\varphi(w) \in E_G$; oder $\varphi(v) = \varphi(w)$ und $\bot \in \ell_G(\varphi(v))$.*

Die Abbildung φ ist ein Isomorphismus *von $\mathcal{H}$ nach $\mathcal{G}$ genau dann, wenn φ ein bijektiver Homomorphismus ist und zusätzlich*

4. $\ell_H(v) = \ell_G(\varphi(v))$ *für alle $v \in V_H$*

gilt. In diesem Fall heißt $\mathcal{H}$ isomorph *zu $\mathcal{G}$, kurz $\mathcal{H} \cong \mathcal{G}$.*

Für die $\mathcal{ALE}$-Konzeptbeschreibungen C und D aus Beispiel 5.17 zeigt Abbildung 5.9 einen Homomorphismus φ von $\mathcal{G}(D)$ nach $\mathcal{G}(C)$, der alle Knoten im Unterbaum mit Wurzel w_1 in $\mathcal{G}(D)$ auf den Knoten v_3 mit Label $\ell_C(v_3) = \{\bot\}$ in $\mathcal{G}(C)$ abbildet.
Wiederum wie für $\mathcal{EL}$ (vgl. Eigenschaft (5.2)) zeigt man durch Induktion über die Tiefe, dass jeder $\mathcal{ALE}$-Beschreibungsbaum $\mathcal{G}$ isomorph ist zum Beschreibungsbaum der zugehörigen $\mathcal{ALE}$-Konzeptbeschreibung $C_\mathcal{G}$, d.h.

$$\mathcal{G} \cong \mathcal{G}(C_\mathcal{G}). \tag{5.13}$$

Bevor nun Subsumtion in $\mathcal{ALE}$ mit Hilfe der soeben eingeführten Begriffe charakterisiert wird, soll an dieser Stelle noch kurz auf den Zusammenhang zwischen den $\mathcal{ALE}$-Normalisierungsregeln und tableaux-basierten Subsumtionsalgorithmen sowie auf die Komplexität der $\mathcal{ALE}$-Normalform eingegangen werden.

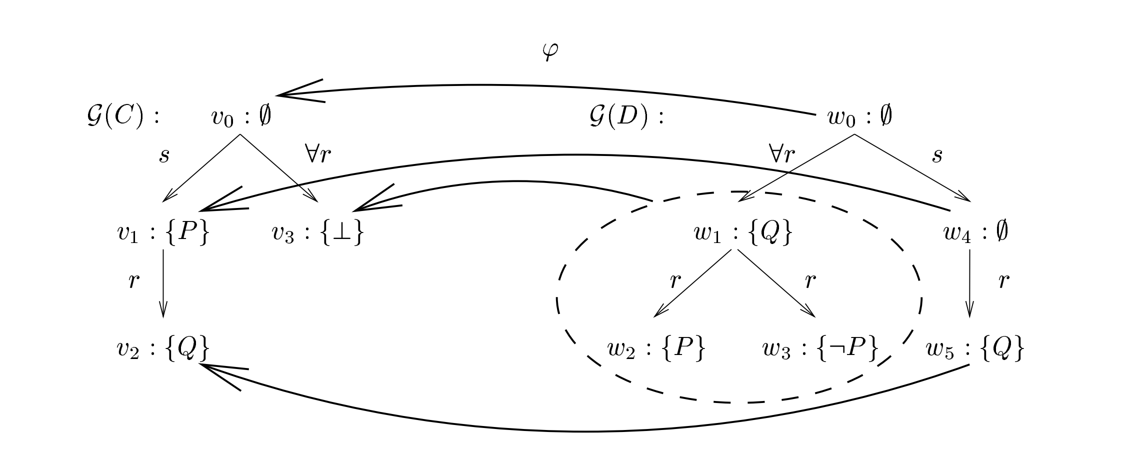


Abbildung 5.9: Ein Homomorphismus zwischen $\mathcal{ALE}$-Beschreibungsbäumen.

Normalisierungsregeln vs. Tableau-Algorithmen.

Es existiert ein enger Zusammenhang zwischen den Normalisierungsregeln aus Definition 5.18 und den sogenannten *Vervollständigungsregeln* in tableaux-basierten Subsumtionsalgorithmen, wie sie z.B. in [DLN+92] für $\mathcal{ALE}$ oder in [SS91] für $\mathcal{ALC}$ eingeführt wurden. So basieren beispielsweise Regel (5.12) in Definition 5.18 und die $\forall$-Regel in [DLN+92, SS91], d.h. die Vervollständigungsregel für Werterestriktionen, auf der gleichen Idee: Wissen, das implizit durch Konjunktionen der Form $\forall r.E \sqcap \exists r.F$ gegeben ist, wird durch *propagieren* von E in die Existenzrestriktion, d.h. durch ableiten der Konjunktion $\forall r.E \sqcap \exists r.(E \sqcap F)$, explizit gemacht. In [DLN+92] wird gezeigt, dass erschöpfendes Anwenden der $\forall$-Regel (im Zusammenspiel mit der Vervollständigungsregel für Existenzrestriktionen) zu einem exponentiell großen Tableaux führen kann. Zugleich zeigen die Autoren, dass das Subsumtionsproblem in $\mathcal{ALE}$ (und auch schon in $\mathcal{FLE}$) NP-vollständig ist. Wie wir im folgenden noch sehen werden, entspricht diesem Komplexitätsresultat bei der hier gegebenen strukturellen Charakterisierung der Subsumtion für $\mathcal{ALE}$ die Tatsache, dass die $\mathcal{ALE}$-Normalform zu einer $\mathcal{ALE}$-Konzeptbeschreibung C exponentiell groß in der Größe von C sein kann.

Beispiel 5.20 *Betrachte die Sequenz* $\{C_1, C_2, C_3, \ldots\}$ *von* $\mathcal{ALE}$*-Konzeptbeschreibungen mit*

$$C_n := \begin{cases} \exists r.P \sqcap \exists r.Q, & n = 1 \\ \exists r.P \sqcap \exists r.Q \sqcap \forall r.C_{n-1}, & n > 1. \end{cases}$$

Man sieht leicht, dass die Größe von C_n *linear in* n *ist. Abbildung 5.10 zeigt den* $\mathcal{ALE}$*-Beschreibungsbaum zur* $\mathcal{ALE}$*-Normalform von* C_2*. Offensichtlich wird durch Anwenden von Regel (5.12) auf* C_2 *die Anzahl der Existenzrestriktionen verdoppelt.*

Durch eine Induktion nach n *erhält man die allgemeine Aussage, dass die* $\mathcal{ALE}$*-Normalform von* C_n *exponentiell groß in* n *ist.*

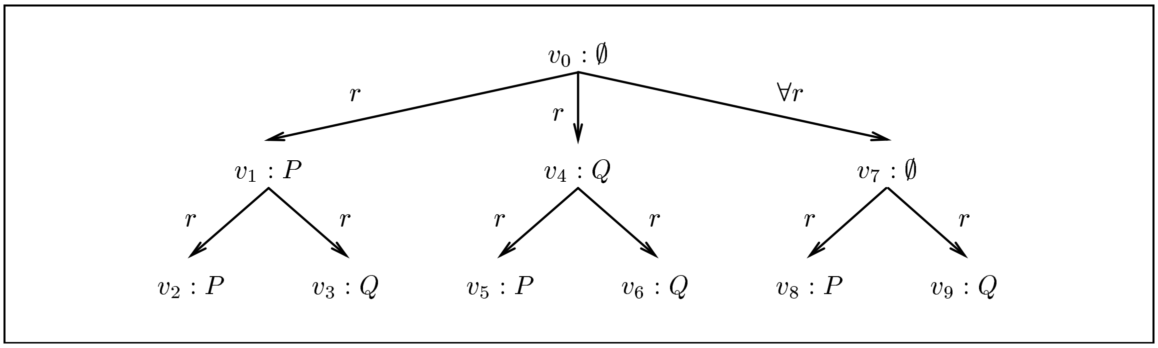


Abbildung 5.10: $\mathcal{ALE}$-Beschreibungsbaum zur $\mathcal{ALE}$-Normalform von C_2.

Die Charakterisierung der Subsumtion in $\mathcal{ALE}$

Mit den oben eingeführten Notationen kann nun die Charakterisierung der Subsumtion von $\mathcal{EL}$ auf $\mathcal{ALE}$ übertragen werden.

Theorem 5.21 *Seien C, D zwei $\mathcal{ALE}$-Konzeptbeschreibungen, $\mathcal{G}_C$ der $\mathcal{ALE}$-Beschreibungsbaum zur $\mathcal{ALE}$-Normalform von C und $\mathcal{G}_D^\top$ der $\mathcal{ALE}$-Beschreibungsbaum zur $\top$-Normalform von D. Es gilt $C \sqsubseteq D$ genau dann, wenn ein Homomorphismus φ von $\mathcal{G}_D^\top$ nach $\mathcal{G}_C$ existiert.*

Der Beweis von Theorem 5.21 ist wie folgt gegliedert: Zunächst wird zum Beweis der Rückrichtung eine allgemeinere Aussage bewiesen (cf Lemma 5.22). Zum Beweis der Hinrichtung werden wir durch Induktion nach der Tiefe von D einen Homomorphismus von $\mathcal{G}_D^\top$ nach $\mathcal{G}_C$ definieren. Der zweite Teil des Beweises ist wesentlich aufwendiger als der erste, da im Induktionsschritt die Voraussetzung $C' \sqsubseteq D'$ nachzuweisen ist (für Konzeptbeschreibungen C', D' mit $depth(C') < depth(C)$, die sich aus Unterbäumen in $\mathcal{G}_C$ bzw. $\mathcal{G}_D^\top$ ergeben). Dieser Nachweis erfolgt jeweils durch Kontraposition unter Verwendung eines sogenannten *kanonischen Modells*.

Beweis der Rückrichtung in Theorem 5.21

Das folgende Lemma formuliert eine Verallgemeinerung der Rückrichtung in Theorem 5.21.

Lemma 5.22 *Seien $\mathcal{G}, \mathcal{H}$ zwei $\mathcal{ALE}$-Beschreibungsbäume. Existiert ein Homomorphismus von $\mathcal{H}$ nach $\mathcal{G}$, so gilt $C_\mathcal{G} \sqsubseteq C_\mathcal{H}$.*

Wegen $C \equiv C_{\mathcal{G}_C}$ und $D \equiv C_{\mathcal{G}_D^\top}$ folgt aus der Existenz eines Homomorphismus von $\mathcal{G}_D^\top$ nach $\mathcal{G}_C$ mit Lemma 5.22 $C \sqsubseteq D$, also die Rückrichtung in Theorem 5.21.

Beweis von Lemma 5.22 durch Induktion über $depth(\mathcal{H})$:
Sei $\mathcal{G} = (V_G, E_G, v_0, \ell_G)$ und $\mathcal{H} = (V_H, E_H, w_0, \ell_H)$, φ ein Homomorphismus von $\mathcal{H}$

nach $\mathcal{G}$ sowie $\mathcal{I} = (\Delta, \cdot^\mathcal{I})$ eine beliebige Interpretation. Gilt $C_\mathcal{G} \equiv \bot$, so ist $C_\mathcal{G}^\mathcal{I} = \emptyset$ und damit $C_\mathcal{G}^\mathcal{I} \subseteq C_\mathcal{H}^\mathcal{I}$. Sonst zeige, dass $x_0 \in C_\mathcal{G}^\mathcal{I}$ auch $x_0 \in C_\mathcal{H}^\mathcal{I}$ impliziert, woraus $C_\mathcal{G}^\mathcal{I} \subseteq C_\mathcal{H}^\mathcal{I}$ und schließlich $C_\mathcal{G} \sqsubseteq C_\mathcal{H}$ folgt. Sei also $x_0 \in C_\mathcal{G}^\mathcal{I}$.

depth$(\mathcal{H}) = 0$: Dann ist nach Konstruktion $C_\mathcal{H} = \bigsqcap_{Q \in \ell_H(w_0)} Q$. Aus $x_0 \in C_\mathcal{G}^\mathcal{I}$ folgt

1. $C_\mathcal{G} \not\equiv \bot$ und damit $\bot \notin \ell_G(v_0)$;
2. $x_0 \in Q^\mathcal{I}$ für alle $Q \in \ell_G(v_0)$.

Da φ Homomorphismus von $\mathcal{H}$ nach $\mathcal{G}$ ist, folgt mit (1) $\ell_H(w_0) \subseteq \ell_G(v_0)$. Also impliziert (2) $x_0 \in Q^\mathcal{I}$ für alle $Q \in \ell_H(w_0)$ und damit $x_0 \in C_\mathcal{H}^\mathcal{I}$.

depth$(\mathcal{H}) > 0$: Dann ist nach Konstruktion

$$C_\mathcal{H} = \bigsqcap_{Q \in \ell_H(w_0)} Q \sqcap \bigsqcap_{w_0 r w \in E_H} \exists r.C_{\mathcal{H}(w)} \sqcap \bigsqcap_{w_0 \forall r w \in E_H} \forall r.C_{\mathcal{H}(w)}.$$

Es ist zu zeigen, dass x_0 Instanz jedes Konjunktes ist.

Wie im Fall *depth*$(\mathcal{H}) = 0$ folgt $x_0 \in Q^\mathcal{I}$ für alle $Q \in \ell_H(w_0)$.

Sei $w_0 r w_1 \in E_H$. Es ist $\bot \notin \ell_G(v_0)$, da sonst $C_\mathcal{G} \equiv \bot$ wäre. Also folgt mit Bedingung 3 in Definition 5.19, dass $v_0 r \varphi(w_1) \in E_G$. Man sieht leicht, dass die Einschränkung von φ auf die Knoten im Unterbaum $\mathcal{H}(w_1)$ einen Homomorphismus von $\mathcal{H}(w_1)$ nach $\mathcal{G}(\varphi(w_1))$ liefert. Per Induktion folgt $C_{\mathcal{G}(\varphi(w_1))} \sqsubseteq C_{\mathcal{H}(w_1)}$. Wegen $v_0 r \varphi(w_1) \in E_G$ existiert nach Konstruktion eine Existenzrestriktion der Form $\exists r.C_{\mathcal{G}(\varphi(w_1))}$ auf Toplevel von $C_\mathcal{G}$. Aus $x_0 \in C_\mathcal{G}^\mathcal{I}$ folgt $x_0 \in (\exists r.C_{\mathcal{G}(\varphi(w_1))})^\mathcal{I}$, d.h. es existiert ein $x_1 \in \Delta$ mit $(x_0, x_1) \in r^\mathcal{I}$ und $x_1 \in C_{\mathcal{G}(\varphi(w_1))})^\mathcal{I}$. Schließlich folgt mit $C_{\mathcal{G}(\varphi(w_1))})^\mathcal{I} \subseteq C_{\mathcal{H}(w_1)}^\mathcal{I}$ auch $x_0 \in (\exists r.C_{\mathcal{H}(w_1)})^\mathcal{I}$.

Sei $w_0 \forall r w_1 \in E_H$. Wie oben erhält man $v_0 \forall r.\varphi(w_1) \in E_G$ und $C_{\mathcal{G}(\varphi(w_1))} \sqsubseteq C_{\mathcal{H}(w_1)}$. Nach Konstruktion ist $\forall r.C_{\mathcal{G}(\varphi(w_1))}$ eine Werterestriktion auf Toplevel von $C_\mathcal{G}$. Mit $x_0 \in C_\mathcal{G}^\mathcal{I}$ folgt also, dass $(x_0, x_1) \in r^\mathcal{I}$ stets $x_1 \in C_{\mathcal{G}(\varphi(w_1))}^\mathcal{I}$ impliziert. Wegen $C_{\mathcal{G}(\varphi(w_1))})^\mathcal{I} \subseteq C_{\mathcal{H}(w_1)}^\mathcal{I}$ folgt also $x_0 \in (\forall r.C_{\mathcal{H}(w_1)})^\mathcal{I}$. □

Beweis der Hinrichtung in Theorem 5.21

Für den eigentlichen Beweis, d.h. die Definition eines Homomorphismus von $\mathcal{G}_D^\top$ nach $\mathcal{G}_C$ unter der Voraussetzung $C \sqsubseteq D$, sind noch einige Vorbereitungen nötig.
Das folgende Lemma fasst zunächst nützliche Eigenschaften der $\mathcal{ALE}$- und der $\top$-Normalform (bzw. der zugehörigen Beschreibungsbäume) zusammen. Diese ergeben sich unmittelbar aus den $\mathcal{ALE}$-Normalisierungsregeln (s. Definition 5.18).

Lemma 5.23 *Sei C eine $\mathcal{ALE}$-Konzeptbeschreibung, $\mathcal{G}_C = (V_C, E_C, v_0, \ell_C)$ der $\mathcal{ALE}$-Beschreibungsbaum zur $\mathcal{ALE}$-Normalform von C und $\mathcal{G}_C^\top = (V_C^\top, E_C^\top, v_0^\top, \ell_C^\top)$ der Baum zur $\top$-Normalform von C. Dann gilt:*

1. *Für alle $v \in V_C$ ist $\mathcal{G}_C(v)$ $\mathcal{ALE}$-normalisiert und für alle $v \in V_C^\top$ ist $\mathcal{G}_C^\top(v)$ $\top$-normalisiert.*

2. *Für alle $v \in V_C$ und alle Rollennamen $r \in N_R$ existiert höchstens eine Kante der Form $v\forall rw$ in E_C.*

3. *$\{vrw, v\forall rw'\} \subseteq E_C$ impliziert $C_{\mathcal{G}_C(w)} \sqsubseteq C_{\mathcal{G}_C(w')}$.*

4. *Falls $\bot \in \ell_C(v)$, dann*

   (a) *$\ell_C(v) = \{\bot\}$ und*

   (b) *$v = v_0$ und $E_C = \emptyset$; oder v ist ein Blatt in $\mathcal{G}_C$ und es existiert $v' \in V_C$ mit $v'\forall rv \in E_C$ für ein $r \in N_R$.*

5. *Falls $v\forall rw \in E_C^\top$, dann ist $C_{\mathcal{G}_C^\top(w)} \not\equiv \top$.*

Im nächsten Schritt wird die sogenannte *kanonische Interpretation* zu einem $\mathcal{ALE}$-Beschreibungsbaum $\mathcal{G}$ definiert. Für einen Baum $\mathcal{G}_C$ zur $\mathcal{ALE}$-Normalform einer konsistenten Konzeptbeschreibung C liefert sie ein Modell von C (cf Lemma 5.25).

Definition 5.24 (Kanonische Interpretation) *Sei $\mathcal{G} = (V, E, v_0, \ell)$ ein $\mathcal{ALE}$-Beschreibungsbaum. Die* kanonische Interpretation *$\mathcal{I}(\mathcal{G})$ ist definiert durch $\mathcal{I}(\mathcal{G}) := (\Delta_{\mathcal{I}(\mathcal{G})}, \cdot^{\mathcal{I}(\mathcal{G})})$ mit*

- $\Delta_{\mathcal{I}(\mathcal{G})} := \{v \in V \mid$ *existieren $v_1, \ldots, v_n$ in V und Rollennamen $r_1, \ldots, r_n$ mit $v_{i-1}r_iv_i \in E$ für alle $1 \le i \le n$ und $v_n = v\}$,*
- *$P^{\mathcal{I}(\mathcal{G})} := \{v \in \Delta_{\mathcal{I}(\mathcal{G})} \mid P \in \ell(v)\}$ für alle $P \in N_C$ und*
- *$r^{\mathcal{I}(\mathcal{G})} := \{(v, w) \in \Delta_{\mathcal{I}(\mathcal{G})} \times \Delta_{\mathcal{I}(\mathcal{G})} \mid vrw \in E\}$ für alle $r \in N_R$.*

Lemma 5.25 *Sei C eine $\mathcal{ALE}$-Konzeptbeschreibung, $\mathcal{G}_C = (V_C, E_C, v_0, \ell_C)$ der $\mathcal{ALE}$-Beschreibungsbaum zur $\mathcal{ALE}$-Normalform von C und $\mathcal{I}(\mathcal{G}_C)$ die kanonische Interpretation von $\mathcal{G}_C$. Ist $\bot \notin \ell_C(v_0)$, so ist $v_0 \in C^{\mathcal{I}(\mathcal{G}_C)}$.*

Beweis: Die Aussage des Lemmas ergibt sich für $\mathcal{G} = \mathcal{G}_C$ unmittelbar aus folgender

Behauptung: Ist $\mathcal{G} = (V, E, v_0, \ell)$ $\mathcal{ALE}$-normalisiert, so gilt für alle $v \in V$:
Ist $\bot \notin \ell(v)$, so ist $v \in C^{\mathcal{I}}_{\mathcal{G}(v)}(\mathcal{G})$.

Beweis der Behauptung durch Induktion nach $depth(\mathcal{G}(v))$:

$depth(\mathcal{G}(v)) = 0$**:** Sei $\bot \notin \ell(v)$. Zeige $v \in Q^{\mathcal{I}(\mathcal{G})}$ für alle $Q \in \ell(v)$. Dies impliziert $v \in C^{\mathcal{I}(\mathcal{G})}_{\mathcal{G}(v)}$. Sei also $Q \in \ell(v)$.

1. $Q \in N_C$. Nach Definition ist $v \in Q^{\mathcal{I}(\mathcal{G})}$.
2. $Q = \neg P$ für ein $P \in N_C$. Es ist $P \notin \ell(v)$; sonst wäre $\mathcal{G}$ nicht $\mathcal{ALE}$-normalisiert, da Regel (5.8) auf $C_{\mathcal{G}(v)}$ anwendbar wäre. Also ist $v \notin P^{\mathcal{I}(\mathcal{G})}$ und damit $v \in (\neg P)^{\mathcal{I}(\mathcal{G})}$.

depth$(\mathcal{G}(v)) > 0$**:** $C_{\mathcal{G}(v)}$ ist von der Form

$$C_{\mathcal{G}(v)} = \bigsqcap_{Q \in \ell(v)} Q \sqcap \bigsqcap_{vrw \in E} \exists r.C_{\mathcal{G}(w)} \sqcap \bigsqcap_{v\forall rw \in E} \forall r.C_{\mathcal{G}(w)}.$$

Zeige, dass v bzgl. $\mathcal{I}(\mathcal{G})$ Instanz aller Konjunkte auf Toplevel von $C_{\mathcal{G}(v)}$ ist. Damit folgt $v \in C_{\mathcal{G}(v)}^{\mathcal{I}(\mathcal{G})}$.

Wie im Fall $depth(\mathcal{G}(v)) = 0$ folgt $v \in Q^{\mathcal{I}(\mathcal{G})}$ für alle $Q \in \ell(v)$.

Sei $vrw \in E$. Es ist $\bot \notin \ell(w)$; sonst wäre $\mathcal{G}$ nicht $\mathcal{ALE}$-normalisiert. Also folgt per Induktion $w \in C_{\mathcal{G}(w)}^{\mathcal{I}(\mathcal{G})}$. Nach Definition ist $(v, w) \in r^{\mathcal{I}(\mathcal{G})}$ und somit $v \in (\exists r.C_{\mathcal{G}(w)})^{\mathcal{I}(\mathcal{G})}$.

Sei $v\forall rw \in E$ und $(v, w') \in r^{\mathcal{I}(\mathcal{G})}$. Zu zeigen: $w' \in C_{\mathcal{G}(w)}^{\mathcal{I}(\mathcal{G})}$. Aus der Definition von $\mathcal{I}(\mathcal{G})$ und $(v, w') \in r^{\mathcal{I}(\mathcal{G})}$ folgt $vrw' \in E$ und $w' \in V$. Es ist $\bot \notin \ell(w)$ und $\bot \notin \ell(w')$; sonst wäre $\mathcal{G}$ wegen Regel (5.9)–(5.12) nicht $\mathcal{ALE}$-normalisiert. Per Induktion folgt $w' \in C_{\mathcal{G}(w')}^{\mathcal{I}(\mathcal{G})}$. Schließlich folgt mit Lemma 5.23 $C_{\mathcal{G}(w')} \sqsubseteq C_{\mathcal{G}(w)}$ und somit $w' \in C_{\mathcal{G}(w)}^{\mathcal{I}(\mathcal{G})}$. Da $(v, w') \in r^{\mathcal{I}(\mathcal{G})}$ beliebig gewählt war, folgt $v \in (\forall r.C_{\mathcal{G}(w)})^{\mathcal{I}(\mathcal{G})}$. □

Als Konsequenz aus Lemma 5.23 und Lemma 5.25 folgt

Lemma 5.26 *Eine $\mathcal{ALE}$-Konzeptbeschreibung C ist inkonsistent genau dann, wenn die $\mathcal{ALE}$-Normalform von C gleich $\bot$ und damit $\mathcal{G}_C = (\{v_0\}, \emptyset, v_0, \ell)$ mit $\ell(v_0) = \{\bot\}$ ist.*

Beweis: Angenommen, $\ell(v_0) = \{\bot\}$. Dann folgt $C_{\mathcal{G}_C} \equiv \bot$ und mit $C \equiv C_{\mathcal{G}_C}$ auch $C \equiv \bot$.

Sei umgekehrt $C \equiv \bot$. Angenommen, $\bot \notin \ell(v_0)$. Dann folgt mit Lemma 5.25, dass $\mathcal{I}(\mathcal{G}_C)$ ein Modell von C ist im Widerspruch zu $C \equiv \bot$. Also ist $\bot \in \ell(v_0)$ und mit Lemma 5.23 folgt $\ell(v_0) = \{\bot\}$ und $E = \emptyset$. □

Im folgenden wird nun die Rückrichtung von Theorem 5.21 bewiesen. Seien dazu C, D $\mathcal{ALE}$-Konzeptbeschreibungen mit $C \sqsubseteq D$ sowie $\mathcal{G}_C = (V_C, E_C, v_0, \ell_C)$ und $\mathcal{G}_D^\top = (V_D, E_D, w_0, \ell_D)$ die entsprechenden $\mathcal{ALE}$-Beschreibungsbäume. Wir zeigen durch Induktion nach *depth*(D), dass ein Homomorphismus φ von $\mathcal{G}_D^\top$ nach $\mathcal{G}_C$ existiert.

Induktionsanfang: $depth(D) = 0$
Ist $C \equiv \bot$, so folgt mit Lemma 5.26 $\ell(v_0) = \{\bot\}$ und $\varphi : \{w_0 \mapsto v_0\}$ liefert einen Homomorphismus von $\mathcal{G}_D^\top$ nach $\mathcal{G}_C$.

Andernfalls ist $D = Q_1 \sqcap \ldots \sqcap Q_n$. Wiederum wird φ definiert durch $\varphi(w_0) := v_0$. Es bleibt zu zeigen $\{Q_1, \ldots, Q_n\} \subseteq \ell_C(v_0)$. Angenommen, $Q_i \notin \ell_C(v_0)$ für ein $i \in \{1, \ldots, n\}$.

1. Ist $Q_i \in N_C$, so folgt für die kanonische Interpretation $\mathcal{I}(\mathcal{G}_C)$ mit Lemma 5.25 $v_0 \in C^{\mathcal{I}(\mathcal{G}_C)}$, aber wegen $v_0 \notin Q_i^{\mathcal{I}(\mathcal{G}_C)}$ ist $v_0 \notin D^{\mathcal{I}(\mathcal{G}_C)}$ im Widerspruch zu $C \sqsubseteq D$.

2. Für $Q_i = \neg P$, $P \in N_C$, definiere eine Erweiterung $\mathcal{J} = (\Delta_\mathcal{J}, \cdot^\mathcal{J})$ von $\mathcal{I}(\mathcal{G}_C) = (\Delta_{\mathcal{I}(\mathcal{G}_C)}, \cdot^{\mathcal{I}(\mathcal{G}_C)})$ wie folgt: $\Delta_\mathcal{J} := \Delta_{\mathcal{I}(\mathcal{G}_C)}$ und $\cdot^\mathcal{J}$ wird definiert wie $\cdot^{\mathcal{I}(\mathcal{G}_C)}$ bis auf $P^\mathcal{J} := P^{\mathcal{I}(\mathcal{G}_C)} \cup \{v_0\}$. Da $\neg P \notin \ell_C(v_0)$, folgt wie im Beweis von Lemma 5.25 $v_0 \in C^\mathcal{J}$, aber $v_0 \notin D^\mathcal{J}$ im Widerspruch zu $C \sqsubseteq D$.

3. Für $Q_i = \bot$ folgt $D \equiv \bot$ und wegen $C \sqsubseteq D$ auch $C \equiv \bot$ im Widerspruch zur Annahme $C \not\equiv \bot$.

Also ist φ ein Homomorphismus von $\mathcal{G}_D^\top$ nach $\mathcal{G}_C$.

Induktionsschritt: $depth(D) > 0$
Ist $C \equiv \bot$, so folgt wie oben, dass φ definiert durch $\varphi(w) := v_0$ für alle $w \in V_D$ einen Homomorphismus von $\mathcal{G}_D^\top$ nach $\mathcal{G}_C$ liefert.

Andernfalls erhält man einen Homomorphismus φ von $\mathcal{G}_D^\top$ nach $\mathcal{G}_C$ wie folgt: Zu jedem direkten Nachfolger w von w_0 in $\mathcal{G}_D^\top$, wird ein $v \in V_C$ bestimmt mit $C_{\mathcal{G}_C(v)} \sqsubseteq C_{\mathcal{G}_D^\top(w)}$. Mit Lemma 5.23 folgt, dass $C_{\mathcal{G}_C(v)}$ $\mathcal{ALE}$-normalisiert und $C_{\mathcal{G}_D^\top(w)}$ $\top$-normalisiert ist. Per Induktion existiert dann ein Homomorphismus φ'_w von $\mathcal{G}(C_{\mathcal{G}_D^\top(w)})$ nach $\mathcal{G}(C_{\mathcal{G}_C(v)})$. Mit Eigenschaft (5.13) folgt dann auch die Existenz eines Homomorphismus φ_w von $\mathcal{G}_D^\top(w)$ nach $\mathcal{G}_C(v)$. Alle auf diese Weise erhaltenen Homomorphismen lassen sich dann zu einem Homomorphismus φ von $\mathcal{G}_D^\top$ nach $\mathcal{G}_C$ integrieren.

Sei also $C \not\equiv \bot$. Wie im Induktionsanfang folgt $\ell_D(w_0) \subseteq \ell_C(v_0)$. Sei $w_0 r w \in E_D$. Dann gilt:

1. $C_{\mathcal{G}_D^\top(w)} \not\equiv \bot$:
   Andernfalls wäre $D \equiv \bot$ und damit auch $C \equiv \bot$ im Widerspruch zur Annahme $C \not\equiv \bot$.

2. Es existiert $v \in V_C$ mit $v_0 r v \in E_C$:
   Angenommen, es existiert kein $v \in V_C$ mit $v_0 r v \in E_C$. Dann ist $v_0 \in C^{\mathcal{I}(\mathcal{G}_C)}$, aber $v_0 \notin D^{\mathcal{I}(\mathcal{G}_C)}$ im Widerspruch zu $C \sqsubseteq D$.

3. Es existiert ein $v \in V_C$ mit $v_0rv \in E_C$ *und* $C_{\mathcal{G}_C(v)} \sqsubseteq C_{\mathcal{G}_D^\top(w)}$:
   Sei $\{v_1, \ldots, v_m\}$ die Menge aller r-Nachfolger von v_0 in $\mathcal{G}_C$, d.h. $v_0rv_i \in E_C$ für alle $1 \leq i \leq m$. Angenommen, $C_{\mathcal{G}_C(v_i)} \not\sqsubseteq C_{\mathcal{G}_D^\top(w)}$ für alle $1 \leq i \leq m$. Dann existieren Interpretationen $\mathcal{I}_i = (\Delta_{\mathcal{I}_i}, \cdot^{\mathcal{I}_i})$ mit

   - $V_C, \Delta_{\mathcal{I}_1}, \ldots, \Delta_{\mathcal{I}_m}$ sind paarweise verschieden und
   - für alle $1 \leq i \leq m$ existiert $x_i \in \Delta_i$ mit $x_i \in C^{\mathcal{I}_i}_{\mathcal{G}_C(v_i)} \setminus C^{\mathcal{I}_i}_{\mathcal{G}_D^\top(w)}$.

   Modifiziere die kanonische Interpretation $\mathcal{I}(\mathcal{G}_C) = (\Delta_{\mathcal{I}(\mathcal{G}_C)}, \cdot^{\mathcal{I}(\mathcal{G}_C)})$ zu $\mathcal{J} = (\Delta_\mathcal{J}, \cdot^\mathcal{J})$ wie folgt:

$$
\begin{aligned}
\Delta_\mathcal{J} &:= (V_C \setminus \{v_1, \ldots, v_m\}) \cup \bigcup_{1 \leq i \leq m} \Delta_{\mathcal{I}_i}, \\
P^\mathcal{J} &:= (P^{\mathcal{I}(\mathcal{G}_C)} \cap \Delta_\mathcal{J}) \cup \bigcup_{1 \leq i \leq m} P^{\mathcal{I}_i}, \\
s^\mathcal{J} &:= (s^{\mathcal{I}(\mathcal{G}_C)} \cap \Delta_\mathcal{J} \times \Delta_\mathcal{J}) \cup \bigcup_{1 \leq i \leq m} s^{\mathcal{I}_i}, \text{ für } s \neq r, \\
r^\mathcal{J} &:= (s^{\mathcal{I}(\mathcal{G}_C)} \cap \Delta_\mathcal{J} \times \Delta_\mathcal{J}) \cup \{(v_0, x_i) \mid 1 \leq i \leq m\} \cup \bigcup_{1 \leq i \leq m} s^{\mathcal{I}_i}.
\end{aligned}
$$

   Dann gilt $v_0 \in C^\mathcal{J}_{\mathcal{G}(C)}$, denn v_0 ist Instanz aller Konjunkte auf Toplevel von $C_{\mathcal{G}_C}$: Aus dem Beweis von Lemma 5.25 folgt, dass $v_0 \in Q^\mathcal{J}$ für alle $Q \in \ell_C(v_0)$ und v_0 Instanz aller Existenz- und Werterestriktionen auf Toplevel von $C_{\mathcal{G}_C}$ der Form $\exists s.C'/\forall s.C''$ mit $s \neq r$ ist. Bleibt zu zeigen, dass $v_0 \in \exists r.C_{\mathcal{G}_C(v_i)}$ für alle $1 \leq i \leq m$ und $v_0 \in \forall r.C'$, wobei $\forall r.C'$ die (!) Werterestriktion zu r auf Toplevel von $C_{\mathcal{G}_C}$ sei. Nach Konstruktion ist, für alle $1 \leq i \leq m$, $(v_0, x_i) \in r^\mathcal{J}$ und $x_i \in C^\mathcal{J}_{\mathcal{G}_C(v_i)} = C^{\mathcal{I}_i}_{\mathcal{G}_C(v_i)}$, also $v_0 \in \exists r.C^\mathcal{J}_{\mathcal{G}_C(v_i)}$. Mit Lemma 5.23 folgt $C_{\mathcal{G}_C(v_i)} \sqsubseteq C'$ für alle $1 \leq i \leq m$, also gilt $x_i \in C'^\mathcal{J}$. Nach Konstruktion bildet die Menge $\{x_1, \ldots, x_m\}$ die Menge aller r-Nachfolger von v_0 in $\mathcal{J}$, also gilt auch $v_0 \in \forall r.C'$.

   Um einen Widerspruch zur Voraussetzung $C \sqsubseteq D$ abzuleiten, bleibt zu zeigen, dass $v_0 \notin D^\mathcal{J}$. Nach Konstruktion ist keiner der r-Nachfolger von v_0 in $\mathcal{J}$ Instanz von $C_{\mathcal{G}_D^\top(w)}$, also ist $v_0 \notin \exists r.C_{\mathcal{G}_D^\top(w)}$ und damit $v_0 \notin D^\mathcal{J}$.

Damit ist gezeigt, dass zu w ein Knoten $v \in V_C$ existiert mit $v_0rv \in E_C$ und $C_{\mathcal{G}_C(v)} \sqsubseteq C_{\mathcal{G}_D^\top(w)}$. Wie oben beschrieben folgt nun per Induktion und Eigenschaft (5.13) die Existenz eines Homomorphismus φ_w von $\mathcal{G}_D^\top(w)$ nach $\mathcal{G}_C(v)$.

Es bleiben die $\forall r$-Nachfolger von w_0 zu betrachten. Sei $w_0 \forall r w \in E_D$. Dann gilt:

1. $C_{\mathcal{G}_D^\top(w)} \not\equiv \top$:
   Folgt unmittelbar aus Lemma 5.23.

2. Es existiert ein $v \in V_C$ mit $v_0 \forall rv \in E_C$:
Andernfalls lässt sich die kanonische Interpretation $\mathcal{I}(\mathcal{G}_C) = (\Delta_{\mathcal{I}(\mathcal{G}_C)}, \cdot^{\mathcal{I}(\mathcal{G}_C)})$ zu einer Interpretation $\mathcal{J} = (\Delta_{\mathcal{J}}, \cdot^{\mathcal{J}})$ erweitern, mit $C^{\mathcal{J}} \not\subseteq D^{\mathcal{J}}$ im Widerspruch zu $C \sqsubseteq D$. Wegen 1. existiert eine Interpretation $\mathcal{I} = (\Delta_{\mathcal{I}}, \cdot^{\mathcal{I}})$ mit $x \in \Delta_{\mathcal{I}}$ und $x \notin C^{\mathcal{I}}_{\mathcal{G}_D^{\top}(w)}$, wobei $\Delta_{\mathcal{I}(\mathcal{G}_C)}$ und $\Delta_{\mathcal{I}}$ o.B.d.A. disjunkt seien. Definiere
$$\begin{array}{rcl} \Delta_{\mathcal{J}} & := & \Delta_{\mathcal{I}(\mathcal{G}_C)} \cup \Delta_{\mathcal{I}}, \\ P^{\mathcal{J}} & := & P^{\mathcal{I}(\mathcal{G}_C)} \cup P^{\mathcal{I}}, \\ s^{\mathcal{J}} & := & \left\{ \begin{array}{ll} s^{\mathcal{I}(\mathcal{G}_C)} \cup s^{\mathcal{I}} & \text{, für } s \neq r, \\ s^{\mathcal{I}(\mathcal{G}_C)} \cup \{(v_0, x)\} \cup s^{\mathcal{I}} & \text{, für } s = r. \end{array} \right. \end{array}$$
Da nach Annahme keine Werterestriktion der Form $\forall r.C'$ auf Toplevel von $C_{\mathcal{G}_C}$ existiert, folgt mit Lemma 5.25, dass $v_0 \in C^{\mathcal{J}}$. Nach Konstruktion ist aber $v_0 \notin (\forall r.C_{\mathcal{G}_D(w)})^{\mathcal{J}}$, also $v_0 \notin D^{\mathcal{J}}$.

3. Es existiert $v \in V_C$ mit $v_0 \forall rv \in E_C$ *und* $C_{\mathcal{G}_C(v)} \sqsubseteq C_{\mathcal{G}_D^{\top}(w)}$:
Mit 2. und Lemma 5.23 existiert genau ein $v \in V_C$ mit $v_0 \forall rv \in E_C$. Angenommen, $C_{\mathcal{G}_C(v)} \not\sqsubseteq C_{\mathcal{G}_D^{\top}(w)}$. Dann existiert eine Interpretation $\mathcal{I} = (\Delta_{\mathcal{I}}, \cdot^{\mathcal{I}})$ und $x \in \Delta_{\mathcal{I}}$ mit $x \in C^{\mathcal{I}}_{\mathcal{G}_C(v)}$ und $x \notin C^{\mathcal{I}}_{\mathcal{G}_D^{\top}}$, wobei $\Delta_{\mathcal{I}(\mathcal{G}_C)}$ und $\Delta_{\mathcal{I}}$ o.B.d.A. disjunkt seien. Definiere $\mathcal{J}$ wie in 2. Dann gilt wiederum $v_0 \in C^{\mathcal{J}}$ aber $v_0 \notin D^{\mathcal{J}}$ im Widerspruch zu $C \sqsubseteq D$.

Damit ist gezeigt, dass zu w ein Knoten $v \in V_C$ existiert mit $v_0 \forall rv \in E_C$ und $C_{\mathcal{G}_C(v)} \sqsubseteq C_{\mathcal{G}_D^{\top}(w)}$. Wiederum folgt per Induktion und Eigenschaft (5.13), dass ein Homomorphismus φ_w von $\mathcal{G}_D^{\top}(w)$ nach $\mathcal{G}_C(v)$ existiert.
Unter Verwendung der soeben definierten Homomorphismen kann $\varphi : V_D \longrightarrow V_C$ wie folgt definiert werden:
$$\varphi := \{w_0 \mapsto v_0\} \cup \bigcup_{w_0 rw \in E_D} \varphi_w \cup \bigcup_{w_0 \forall rw \in E_D} \varphi_w.$$
Nach Konstruktion ist φ ein Homomorphismus von $\mathcal{G}_D^{\top}$ nach $\mathcal{G}_C$. Dies schließt den Beweis zu Theorem 5.21 ab. □

5.3.2 Charakterisierung des LCS in $\mathcal{ALE}$

Nachdem im vorhergehenden Abschnitt die Charakterisierung der Subsumtion von $\mathcal{EL}$ auf $\mathcal{ALE}$ übertragen wurde, soll nun auch die Charakterisierung des LCS in $\mathcal{EL}$ aus Theorem 5.9 auf $\mathcal{ALE}$ übertragen werden. Im ersten Schritt wird dazu die Definition des Produktes angepasst. Dabei sind zum einen $\forall$-Kanten zu berücksichtigen. Zum anderen sind Inkonsistenzen geeignet zu behandeln, genauer gesagt, (Unter-) Bäume, deren Wurzel $\bot$ im Label enthält. Solch einem Baum $\mathcal{G}_\bot$ entspricht auf Seiten der

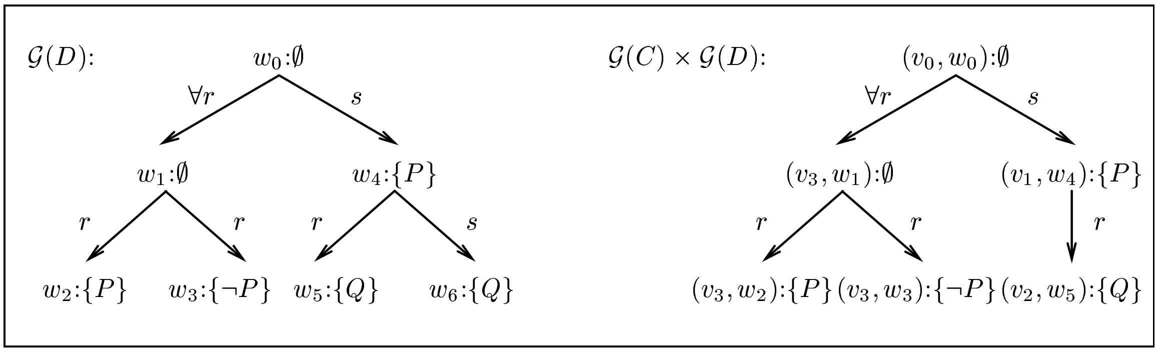


Abbildung 5.11: Das Produkt zweier $\mathcal{ALE}$-Beschreibungsbäume.

Konzeptbeschreibungen $\bot$. Da $\bot \sqsubseteq C$ für alle Konzeptbeschreibungen C gilt, ist $\mathsf{lcs}(\bot, C) \equiv C$. Folglich muss das Produkt von $\mathcal{G}_\bot$ und $\mathcal{G}_C$ so definiert werden, dass $C_{\mathcal{G}_\bot \times \mathcal{G}_C} \equiv C$.

Definition 5.27 *Seien $\mathcal{G} = (V_G, E_G, v_0, \ell_G)$ und $\mathcal{H} = (V_H, E_H, w_0, \ell_H)$ zwei $\mathcal{ALE}$-Beschreibungsbäume. Das Produkt $\mathcal{G} \times \mathcal{H} = (V, E, (v_0, w_0), \ell)$ ist induktiv definiert durch:*

1. *Ist $\bot \in \ell_G(v_0)$, so ist $\mathcal{G} \times \mathcal{H}$ definiert als der Beschreibungsbaum, den man aus $\mathcal{H}$ erhält, indem man jeden Knoten $w \in V_H$ durch (v_0, w) ersetzt.*

2. *Ist $\bot \in \ell_H(w_0)$, so ist $\mathcal{G} \times \mathcal{H}$ definiert als der Beschreibungsbaum, den man aus $\mathcal{G}$ erhält, indem man jeden Knoten $v \in V_G$ durch (v, w_0) ersetzt.*

3. *Sonst ist $\mathcal{G} \times \mathcal{H}$ induktiv definiert durch:*

   - $\ell(v_0, w_0) := \ell_C(v_0) \cap \ell_D(w_0)$,
   - *zu jedem Paar (v, w) mit $v_0 r v \in E_G$ und $w_0 r w \in E_H$ bzw. $v_0 \forall r v \in E_G$ und $w_0 \forall r w \in E_H$ ist $(v_0, w_0) r (v, w) \in E$ bzw. $(v_0, w_0) \forall r (v, w) \in E$,*
   - *zu jedem Nachfolger (v, w) von (v_0, w_0) enthält $\mathcal{G} \times \mathcal{H}$ das induktiv definierte Produkt $\mathcal{G}(v) \times \mathcal{H}(w)$ als Unterbaum.*

Als Beispiel betrachte man Abbildung 5.11: links ist der $\mathcal{ALE}$-Beschreibungsbaum zu $D = \forall r.(\exists r.P \sqcap \exists r.\neg P) \sqcap \exists s.(P \sqcap \exists r.Q \sqcap \exists s.Q)$ abgebildet und rechts das Produkt von $\mathcal{G}(D)$ und dem $\mathcal{ALE}$-Beschreibungsbaum $\mathcal{G}(C)$ aus Abbildung 5.8.

Theorem 5.28 *Seien C, D $\mathcal{ALE}$-Konzeptbeschreibungen und $\mathcal{G}_C$, $\mathcal{G}_D$ die $\mathcal{ALE}$-Beschreibungsbäume zu den $\mathcal{ALE}$-Normalformen von C und D. Der LCS von C und D ist gegeben durch die Konzeptbeschreibung $C_{\mathcal{G}_C \times \mathcal{G}_D}$ zum Produkt von $\mathcal{G}_C$ und $\mathcal{G}_D$.*

Beweis (analog zum Beweis von Theorem 5.9):
Seien $\mathcal{G}_C = (V_C, E_C, v_0, \ell_C)$, $\mathcal{G}_D = (V_D, E_D, w_0, \ell_D)$ und $\mathcal{G}_C \times \mathcal{G}_D = (V, E, (v_0, w_0), \ell)$. Es ist zu zeigen:

1. $C \sqsubseteq C_{\mathcal{G}_C \times \mathcal{G}_D}$,
2. $D \sqsubseteq C_{\mathcal{G}_C \times \mathcal{G}_D}$ und
3. für jedes C' mit $C \sqsubseteq C'$ und $D \sqsubseteq C'$ gilt $C_{\mathcal{G}(C) \times \mathcal{G}(D)} \sqsubseteq C'$.

Wie im Falle von $\mathcal{EL}$-Beschreibungsbäumen liefern auch für $\mathcal{ALE}$-Beschreibungsbäume die Projektionen π_1 und π_2 Homomorphismen von $\mathcal{G}_C \times \mathcal{G}_D$ nach $\mathcal{G}_C$ (für $i = 1$) bzw. nach $\mathcal{G}_D$ (für $i = 2$). Mit Lemma 5.22 und Eigenschaft 5.5 folgt $C \sqsubseteq C_{\mathcal{G}_C \times \mathcal{G}_D}$ bzw. $D \sqsubseteq C_{\mathcal{G}_C \times \mathcal{G}_D}$.
Sei C' ein gemeinsamer Subsumierer von C und D und $\mathcal{G}_{C'}^{\top} = (V', E', v_0', \ell')$ der $\mathcal{ALE}$-Beschreibungsbaum zur $\top$-Normalform von C'. Mit Theorem 5.21 folgt, dass Homomorphismen φ_1, φ_2 von $\mathcal{G}_{C'}^{\top}$ nach $\mathcal{G}_C$ bzw. $\mathcal{G}_D$ existieren. Wiederum wird eine Abbildung φ von $\mathcal{G}_{C'}^{\top}$ nach $\mathcal{G}_C \times \mathcal{G}_D$ definiert durch $\varphi(v') := (\varphi_1(v'), \varphi_2(v'))$ für alle $v' \in V'$. Der Beweis zur Wohldefiniertheit von φ aus dem Beweis von Theorem 5.9 lässt sich leicht übertragen. Es bleibt zu zeigen, dass φ den Bedingungen aus Definition 5.19 genügt.

Ad 1: Da φ_1, φ_2 Homomorphismen sind, ist $\varphi_1(v_0') = v_0$ und $\varphi_2(v_0') = w_0$, also $\varphi(v_0') = (v_0, w_0)$.

Ad 2: Für alle $v' \in V'$ ist zu zeigen: $\ell'(v') \subseteq \ell(\varphi_1(v'), \varphi_2(v'))$ oder $\bot \in \ell(\varphi_1(v'), \varphi_2(v'))$. Es sind mehrere Fälle zu unterscheiden.

1. $\ell'(v') \subseteq \ell_C(\varphi_1(v'))$ und $\ell'(v') \subseteq \ell_D(\varphi_2(v'))$.
   Dann gilt offensichtlich $\ell'(v') \subseteq \ell_C(\varphi_1(v')) \cap \ell_D(\varphi_2(v')) = \ell(\varphi(v'))$.
2. $\ell'(v') \subseteq \ell_C(\varphi_1(v'))$ und $\bot \in \ell_D(\varphi_2(v'))$.
   Nach Definition 5.27 ist dann $\ell(\varphi_1(v'), \varphi_2(v')) = \ell_C(\varphi_1(v'))$, und damit $\ell'(v') \subseteq \ell(\varphi(v'))$.
3. $\bot \in \ell_C(\varphi_1(v'))$ und $\ell'(v') \subseteq \ell_D(\varphi_2(v'))$.
   Analog zu 2.
4. $\bot \in \ell_C(\varphi_1(v'))$ und $\bot \in \ell_D(\varphi_2(v'))$.
   Nach Konstruktion des Produktes ist dann auch $\bot \in \ell(\varphi_1(v'), \varphi_2(v'))$.

Ad 3: Sei $v'\rho w' \in E'$ mit $\rho = r$ oder $\rho = \forall r$ für ein $r \in N_R$. Dann ist zu zeigen: $\varphi(v')\rho\varphi(w') \in E$; oder $\varphi(v') = \varphi(w')$ und $\bot \in \ell(\varphi(v'))$. Es sind die folgenden Fälle zu betrachten:

1. $\varphi_1(v')\rho\varphi_1(w') \in E_C$ und $\varphi_2(v')\rho\varphi_2(w') \in E_D$,

2. $\varphi_1(v')\rho\varphi_1(w') \in E_C$ und $\varphi_2(v') = \varphi_2(w')$ und $\bot \in \ell_D(\varphi_2(v'))$,
3. $\varphi_1(v') = \varphi_1(w')$ und $\bot \in \ell_C(\varphi_1(v'))$ und $\varphi_2(v')\rho\varphi_2(w') \in E_D$,
4. $\varphi_1(v') = \varphi_1(w')$ und $\bot \in \ell_C(\varphi_1(v'))$ und $\varphi_2(v') = \varphi_2(w')$ und $\bot \in \ell_D(\varphi_2(v'))$.

In allen vier Fällen folgt $(\varphi_1(v'), \varphi_2(v')) \in V$ aus der Wohldefiniertheit von φ. In den ersten drei Fällen folgt dann nach Konstruktion des Produktes jeweils $(\varphi_1(v'), \varphi_2(v'))\rho(\varphi_1(w'), \varphi_2(w')) \in E$. Im vierten Fall gilt $\varphi(v') = \varphi(w')$ und nach Konstruktion des Produktes ist $\bot \in \ell(\varphi_1(v'), \varphi_2(v'))$. □

Bevor im folgenden Paragraphen die Komplexität des LCS zweier $\mathcal{ALE}$-Konzeptbeschreibungen untersucht wird, verdeutlichen die folgenden Beispiele noch die Notwendigkeit, zur Bestimmung des LCS aus dem Produkt die Beschreibungsbäume der *$\mathcal{ALE}$-Normalformen* zur Berechnung des Produktes zu verwenden.

Beispiel 5.29 *Betrachte nochmals die Beschreibungsbäume $\mathcal{G}(C)$ und $\mathcal{G}(D)$ aus Abbildung 5.7. Die Konzeptbeschreibung $C_{\mathcal{G}(C)\times\mathcal{G}(D)} = \forall r.P \sqcap \forall r.Q \sqcap \exists s.\top$ ist zwar ein Common Subsumer von C und D, aber wegen $C \sqsubseteq D$ ist $\mathsf{lcs}(C, D) \equiv D$ und somit wegen $D \sqsubset C_{\mathcal{G}(C)\times\mathcal{G}(D)}$ nicht der LCS von C und D. Man sieht aber leicht, dass $C_{\mathcal{G}_C\times\mathcal{G}_D} \not\equiv D$.*

Beispiel 5.30 *Dieses besonders einfache Beispiel zeigt, dass auch die „Normalisierung von Inkonsistenzen" notwendig ist, um den LCS aus dem Produkt der Bäume zu erhalten. Sei $C = P \sqcap \neg P$ und $D = Q$. Dann ist offensichtlich $C_{\mathcal{G}(C)\times\mathcal{G}(D)} = \top \sqsupset Q = \mathsf{lcs}(C, D)$. Hier liefert die Anwendung der Normalisierungsregeln $\bot$ als Normalform zu C und damit $C_{\mathcal{G}_C\times\mathcal{G}_D} = Q$.*

Komplexität des LCS in $\mathcal{ALE}$

Wie bereits erwähnt, ist der $\mathcal{ALE}$-Beschreibungsbaum zur $\mathcal{ALE}$-Normalform einer Konzeptbeschreibung ggf. exponentiell groß. Dies führt dazu, dass in $\mathcal{ALE}$ anders als in $\mathcal{EL}$ bereits der LCS *zweier* $\mathcal{ALE}$-Konzeptbeschreibungen exponentiell groß in der Größe der Ausgangskonzepte sein kann.

Beispiel 5.31 (Fortsetzung der Beispiele 5.11 und 5.20) *Betrachte nochmals für $n \geq 1$ die Konzeptbeschreibung D_n aus Beispiel 5.11 und die Konzeptbeschreibung C_n aus Beispiel 5.20. Man sieht leicht, dass das Produkt des Baumes zur $\mathcal{ALE}$-Normalform von C_n und des Baumes zu D_n wiederum dem vollen binären Baum $\mathcal{B}_n$ der Tiefe n (wie im Beweis zu Lemma 5.12 definiert) entspricht.*

Ähnlich wie im Beweis von Lemma 5.12 zeigt man, dass keine $\mathcal{ALE}$-Konzeptbeschreibung existiert, die zum einen äquivalent zu und zum anderen kleiner als $C_{\mathcal{B}_n}$ ist. Da sowohl C_n als auch D_n linear groß in n sind und außerdem $\mathcal{B}_n$ exponentiell groß in n ist, folgt

Satz 5.32 *Der LCS zweier $\mathcal{ALE}$-Konzeptbeschreibungen C und D kann exponentiell groß bzgl. der Größe von C und D sein.*

Durch Induktion nach der Tiefe von C lässt sich leicht zeigen, dass die Größe des Baumes zur $\mathcal{ALE}$-Normalform von C exponentiell beschränkt werden kann. Da die Größe des LCS von n Konzeptbeschreibungen $C_1, \ldots, C_n$ durch das Produkt der Größen der zugehörigen $\mathcal{ALE}$-Beschreibungsbäume $\mathcal{G}_{C_i}$ beschränkt werden kann, folgt trivialerweise, dass der LCS von n $\mathcal{ALE}$-Konzeptbeschreibungen höchstens exponentiell groß in der Größe der Eingabekonzepte sein kann.

Kapitel 6

Formale Begriffsanalyse und Least Common Subsumer

Bei der Berechnung des LCS zu einer Menge $\mathcal{E}$ von Konzeptbeschreibungen hängt die Qualität des Ergebnisses offensichtlich sehr stark von der Wahl dieser Konzeptbeschreibungen ab. Einerseits kann das Ergebnis zu allgemein (z.B. $\top$) sein, wenn die gewählten Ausgangskonzepte zu unterschiedlich sind. Andererseits kann das Ergebnis auch zu speziell (z.B. gleich einem der Ausgangskonzepte) sein, wenn sich die Ausgangskonzepte zu ähnlich sind.

Den in Abschnitt 4.2 vorgestellten Szenarien zur Unterstützung der bottom-up Konstruktion bzw. Erweiterung der Wissensbasis in unserer Prozesstechnikanwendung folgend besteht unser Lösungsansatz für dieses Problem darin, nicht nur den LCS der *vollständigen Menge* $\mathcal{E}$ zu berechnen, sondern die LCSs zu allen *Teilmengen* von $\mathcal{E}$. Diese LCSs werden dann in einer Subsumtionshierarchie angeordnet, aus der mit Hilfe geeigneter Kriterien Kandidaten bestimmt werden, die dem Anwender zur Inspektion (und möglichen Weiterverarbeitung) angeboten werden. Im schlimmsten Fall kann die resultierende Hierarchie (im folgenden als *LCS-Hierarchie* bezeichnet) natürlich exponentiell groß bzgl. der Mächtigkeit von $\mathcal{E}$ sein. Die durch die Kriterien bestimmte tatsächliche Menge von Kandidaten sollte aber jeweils überschaubar und für den Anwender handhabbar sein.

Für die Realisierung dieses Ansatzes kommt eine naive Berechnung der LCS-Hierarchie nicht in Frage. Grund hierfür ist zum einen, dass die Zahl der zu berücksichtigenden Teilmengen von Konzeptbeschreibungen exponentiell groß ist. Zum anderen sind sowohl der Subsumtionstest als auch die Berechnung des LCS für die in der Anwendung eingesetzte BL $\mathcal{ALE}$ „teure“ Operationen. Daher ist das Ziel, vollständige Information über die Struktur der Hierarchie der LCS zu erhalten,

1. ohne die LCS zu allen Teilmengen tatsächlich zu berechnen und
2. ohne alle möglichen Subsumtionsbeziehungen zwischen diesen LCS explizit zu testen.

Genau dieses Ziel lässt sich durch den Einsatz von Methoden aus der Formalen Begriffsanalyse [GW99] erreichen.

In der Formalen Begriffsanalyse werden *Begriffe* eines Anwendungsbereichs durch ein Paar *(Extension, Intension)* beschrieben, d.h. durch eine Menge von *Objekten* und eine Menge von *Merkmalen*.[1] Die Beziehungen zwischen Merkmalen einer Menge $\mathcal{M}$ und Objekten einer Menge $\mathcal{O}$ werden in einem *Kontext* durch eine Inzidenzrelation $I \subseteq \mathcal{M} \times \mathcal{O}$ festgelegt. Sind sowohl $\mathcal{M}$ als auch $\mathcal{O}$ endlich, kann man sich I als eine Kreuztabelle vorstellen. Ein Begriff ist nun ein Paar $(A, B) \in \mathcal{O} \times \mathcal{M}$, sodass (i) B jedes Merkmal enthält, das alle Objekte aus A haben und (ii) A alle Objekte enthält, die alle Merkmale aus B haben. Anschaulich entsprechen Begriffe im endlichen Fall maximalen Rechtecken in der Kreuztabelle des Kontextes. Begriffe bilden bzgl. Unterbegriffsbeziehung (d.h. Mengeninklusion der Extension) einen vollständigen Verband. Im endlichen Fall, d.h. bei endlichen Mengen $\mathcal{O}$ und $\mathcal{M}$, kann dieser Begriffsverband trivialerweise effektiv berechnet werden [GW99]. Im Falle einer endlichen Merkmalsmenge $\mathcal{M}$ und einer unendlichen Objektmenge $\mathcal{O}$ lässt er sich mit Hilfe des *Merkmalsexplorationsalgorithmus* unter zusätzlicher Verwendung eines Experten (ein menschlicher Benutzer oder ein Programm), der Vermutungen über Zusammenhänge zwischen Begriffe bestätigt oder widerlegt, effizient berechnen [Gan91, GW99].

Im folgenden wird gezeigt, dass sich ein zu diesem Algorithmus dualer Algorithmus zur *Objektexploration* für die Berechnung der LCS-Hierarchie verwenden lässt. Genauer: Für eine gegebene Menge $\{C_1, \ldots, C_n\}$ von Konzeptbeschreibungen wird ein formaler Kontext definiert, dessen Begriffsverband isomorph ist zu der durch die Menge $\{C_1, \ldots, C_n\}$ induzierten LCS-Hierarchie. Folglich kann die LCS-Hierarchie durch den Objektexplorationsalgorithmus berechnet werden. Allerdings benötigt er für diese Berechnung noch einen Experten, der im vorliegenden Fall durch den Algorithmus zur Berechnung des LCS sowie einen Subsumtionsalgorithmus für $\mathcal{ALE}$ realisiert werden kann. Obwohl nun im worst case eine exponentielle Anzahl von Aufrufen des Experten möglich ist, zeigen Erfahrungen aus Anwendungen der Formalen Begriffsanalyse [SW00], dass sich der Explorationsalgorithmus in der Praxis deutlich besser verhält und somit für unser Ziel einen vielversprechenden Ansatz liefert.

Eine Alternative zur Berechnung der LCS-Hierarchie

Ein alternativer Lösungsansatz zur Berechnung „guter“ LCS, d.h. von LCS, die einen aus Sicht des Modellierers guten Kandidaten für die gesuchte neue Klasse liefern, besteht darin, in einem ersten Schritt die Menge $\mathcal{E}$ in *Cluster* $\mathcal{E}_1, \ldots, \mathcal{E}_n$ möglichst ähnlicher Konzeptbeschreibungen zu unterteilen. In einem zweiten Schritt werden als Kandidaten dann die LCS der Mengen $\mathcal{E}_1, \ldots, \mathcal{E}_n$ berechnet. Offensichtlich ist

[1] Alle in diesem Absatz verwendeten Begriffe aus der Formalen Begriffsanalyse werden in Abschnitt 6.1 formal eingeführt; hier werden sie informell verwendet, um einen Überblick über Aufbau und Ziele des Kapitels zu geben.

hierbei die Wahl des Ähnlichkeitsmaßes, durch das die Cluster bestimmt werden, der entscheidende Faktor für die Qualität des Ergebnisses. Warum dieser Ansatz im Rahmen der vorliegenden Arbeit nicht verfolgt wurde, lässt sich knapp wie folgt begründen: Auf Seiten der BL würde ein intuitiver Ansatz zum Clustern auf dem oben beschriebenen Ansatz aufbauen und die Ähnlichkeit zweier Konzeptbeschreibungen an der Aussagekraft ihres LCS, d.h. an seiner Lage in der Subsumtionshierarchie, festmachen. Auf Seiten der Anwendung haben Erfahrungen mit Ähnlichkeitsmaßen für Bausteine bzw. Klassen gezeigt, dass solche Maße stets geeignete Heuristiken einbeziehen müssen, um akzeptable Ergebnisse zu erzielen. Ist man also am Einsatz des Clusterns von Konzeptbeschreibungen in der Prozesstechnik interessiert, müssen diese Heuristiken im Ähnlichkeitsmaß für Konzeptbeschreibungen berücksichtigt werden. Folglich würde das Clustern deutlich über den oben beschriebenen Ansatz zur Bestimmung der Kandidaten hinausgehen und dabei einen wesentlich größeren Overhead erzeugen. Dies erschien in der betrachteten Anwendungssituation und bei den mit obigem Ansatz erzielten Ergebnissen (cf Kapitel 9) nicht gerechtfertigt.

6.1 Formale Begriffsanalyse

Dieser Abschnitt führt nur die Begriffe aus der Formalen Begriffsanalyse ein, die für die Beschreibung des Explorationsalgorithmus unbedingt notwendig sind. Für eine umfassende Einführung in die Formale Begriffsanalyse und ihre Anwendungen sei auf [GW99, SW00] verwiesen.

Definition 6.1 (Kontext) *Ein* formaler Kontext *ist ein Tripel $\mathcal{K} = (\mathcal{O}, \mathcal{M}, I)$, wobei*

- *$\mathcal{O}$ eine Menge von Objekten ist,*
- *$\mathcal{M}$ eine Menge von Merkmalen und*
- *$I \subseteq \mathcal{O} \times \mathcal{M}$ eine Relation, die Objekte o mit Merkmalen m verbindet; ist o mit m durch I verbunden, so* hat *das Objekt o das Merkmal m.*

Sei $\mathcal{K} = (\mathcal{O}, \mathcal{M}, I)$ ein formaler Kontext. Für eine Menge $A \subseteq \mathcal{O}$ von Objekten ist die *Intension* A' von A definiert als die Menge aller Merkmale, die alle Objekte in A haben, d.h.

$$A' := \{m \in \mathcal{M} \mid \text{ für alle } a \in A \text{ gilt } (a, m) \in I\}.$$

Entsprechend ist für eine Menge $B \subseteq \mathcal{M}$ von Merkmalen die *Extension* B' von B definiert als die Menge aller Objekte, die alle Merkmale aus B haben, d.h.

$$B' := \{o \in \mathcal{O} \mid \text{ für alle } b \in B \text{ gilt } (o, b) \in I\}.$$

Man sieht leicht, dass für Mengen $A_1 \subseteq A_2 \subseteq \mathcal{O}$ und $B_1 \subseteq B_2 \subseteq \mathcal{M}$ gilt:

- $A_2' \subseteq A_1'$ und $B_2' \subseteq B_1'$ und
- $A_1 \subseteq A_1''$ und $A_1' = A_1'''$ sowie $B_1 \subseteq B_1''$ und $B_1' = B_1'''$.

Ein *formaler Begriff* ist ein Paar (A, B) bestehend aus einer Extension $A \subseteq \mathcal{O}$ und einer Intension $B \subseteq \mathcal{M}$ mit $A' = B$ und $B' = A$. Formale Begriffe können durch folgende Ordnung hierarchisch geordnet werden:

$$(A_1, B_1) \leq (A_2, B_2) \text{ genau dann, wenn } A_1 \subseteq A_2.$$

Tatsächlich induziert diese Ordnung einen vollständigen Verband, den sogenannten *Begriffsverband* des Kontextes. Gegeben einen Kontext in einer konkreten Anwendung, besteht der erste Schritt zur Analyse dieses Kontextes meist in der Berechnung des Begriffsverbandes.

Ein naiver Ansatz zur Berechnung des Begriffsverbandes für einen endlichen Kontext basiert auf den Aussagen im folgenden Lemma. Der Beweis des Lemmas folgt leicht aus den oben gegebenen Definitionen.

Lemma 6.2 *Alle formalen Begriffe eines Kontextes sind von der Form (A'', A') für eine geeignete Menge $A \subseteq \mathcal{O}$ und jedes Paar dieser Form ist ein formaler Begriff. Desweiteren gilt $(A_1'', A_1') \leq (A_2'', A_2')$ genau dann, wenn $A_2' \subseteq A_1'$.*

Folglich lässt sich für einen *endlichen* Kontext der Begriffsverband im Prinzip dadurch berechnen, dass man alle Teilmengen A von $\mathcal{O}$ aufzählt und jeweils den Formalen Begriff (A'', A') berechnet. Dieser naive Ansatz ist natürlich in Anwendungen mit sehr großen Mengen von Objekten sehr ineffizient und in Anwendungen mit unendlicher Objektmenge nicht möglich. Allerdings stehen diesen großen/unendlichen Objektmengen meist relativ kleine Mengen von Merkmalen gegenüber. In solchen Situationen hat sich der Merkmalsexplorationsalgorithmus nach Ganter [Gan91, Stu96b, GW99] als ein effizienter Ansatz zur Berechnung des Begriffsverbandes herausgestellt.

Es wird sich zeigen, dass wir uns bei unserer Anwendung, d.h. der Berechnung der LCS-Hierarchie, in einer *dualen* Situation befinden: die Menge der Merkmale wird die *unendliche* Menge aller Konzeptbeschreibungen der betrachteten BL sein, wohingegen die Menge der Objekte die *endliche* Menge der gegebenen Konzeptbeschreibungen $C_1, \ldots, C_n$ ist, zu der die LCS-Hierarchie zu berechnen ist. Folglich sind sowohl der Algorithmus als auch die zugehörigen grundlegenden Definitionen auf die duale Situation zu übertragen.

Alternativ hätte die Möglichkeit bestanden, statt des durch die Objekte $C_1, \ldots, C_n$ induzierten Kontextes den entsprechenden dualen Kontext zu betrachten und auf diesen den üblichen Merkmalsexplorationsalgorithmus anzuwenden. (Den dualen Kontext erhält man durch Transponieren der von I induzierten Matrix.) Zum einen wäre aber die Betrachtung des dualen Kontextes weniger intuitiv. Zum anderen verursacht die Verwendung der dualen Version des Algorithmus keine zusätzliche, beweistechnische Arbeit, da die Korrektheit des dualen Algorithmus unmittelbar aus der Korrektheit des üblichen Algorithmus und der Tatsache folgt, dass die Ergebnisse des dualen Algorithmus mit denen des üblichen Algorithmus auf dem dualen Kontext übereinstimmen.

Im nächsten Abschnitt wird also der zum aus der Literatur bekannten Algorithmus zur Merkmalsexploration duale Algorithmus zur Objektexploration, im folgenden nur *Objektexploration* genannt, vorgestellt.

6.2 Objektexploration

Um die duale Version von Ganter's Algorithmus spezifizieren zu können, werden noch einige Definitionen benötigt. Die wichtigste ist dabei die der *Implikation zwischen Objekten.* Eine solche Implikation $A_1 \to A_2$ ist in einem Kontext erfüllt, falls jedes Merkmal, das alle Objekte in A_1 haben, auch alle Objekte in A_2 haben.

Definition 6.3 (Implikation) *Sei $\mathcal{K} = (\mathcal{O}, \mathcal{M}, I)$ ein formaler Kontext und seien A_1, A_2 Teilmengen von $\mathcal{O}$. Die* Objektimplikation $A_1 \to A_2$ *gilt in $\mathcal{K}$ ($\mathcal{K} \models A_1 \to A_2$) genau dann, wenn $A_1' \subseteq A_2'$. Ein Merkmal m* verletzt *die Implikation $A_1 \to A_2$ genau dann, wenn $m \in A_1' \setminus A_2'$.*

Man sieht leicht, dass die Implikation $A_1 \to A_2$ genau dann in $\mathcal{K}$ erfüllt ist, wenn $A_2 \subseteq A_1''$. Insbesondere gilt also für jede Menge A von Objekten die Implikation $A \to (A'' \setminus A)$. Die Menge aller in $\mathcal{K}$ gültigen Implikationen wird mit $Imp_{\mathcal{O}}(\mathcal{K})$ bezeichnet. Da diese Menge sehr groß sein kann, ist man an „kleinen“ Erzeugendensystemen der Menge $Imp_{\mathcal{O}}(\mathcal{K})$ interessiert, d.h. an Teilmengen von $Imp_{\mathcal{O}}(\mathcal{K})$, aus denen alle in $\mathcal{K}$ gültigen Implikationen abgeleitet werden können.

Definition 6.4 (Implikationshülle, Basis) *Sei $\mathcal{J}$ eine Menge von Objektimplikationen und A eine Teilmenge von $\mathcal{O}$. Die* Implikationshülle $\mathcal{J}(A)$ *von A bzgl. $\mathcal{J}$ ist definiert als die kleinste Teilmenge H von $\mathcal{O}$ mit*

- *$A \subseteq H$ und*
- *$A_1 \to A_2 \in \mathcal{J}$ und $A_1 \subseteq H$ impliziert $A_2 \subseteq H$.*

Die Menge aller durch $\mathcal{J}$ induzierten Objektimplikationen *enthält genau die Implikationen $A_1 \to A_2$, für die $A_2 \subseteq \mathcal{J}(A_1)$ gilt. Sie wird mit* $\mathrm{Cons}(\mathcal{J})$ *bezeichnet. Eine Menge von Implikationen $\mathcal{J}$ heißt* Basis von $Imp_{\mathcal{O}}(\mathcal{K})$ *genau dann, wenn $Imp_{\mathcal{O}}(\mathcal{K}) = Cons(\mathcal{J})$ und es keine echte Teilmenge von $\mathcal{J}$ mit dieser Eigenschaft gibt.*

Man kann zeigen [GW99], dass für eine Basis $\mathcal{J}$ von $Imp_{\mathcal{O}}(\mathcal{K})$ und alle Teilmengen A von $\mathcal{O}$ gilt $A'' = \mathcal{J}(A)$. Daraus folgt insbesondere, dass der $\cdot''$-Operator $A \mapsto A''$, der im Explorationsalgorithmus verwendet wird, berechnet werden kann, ohne die möglicherweise unendliche Intension A' von A explizit zu berechnen. Die Implikationshülle $\mathcal{J}(A)$ lässt sich sogar sehr effizient berechnen (cf Bemerkung 6.5). Wir können also bei gegebener Basis $\mathcal{J}$ von $Imp_{\mathcal{O}}(\mathcal{K})$ jede Frage der Form "$A_1 \to A_2 \in Imp_{\mathcal{O}}(\mathcal{K})$?" in linearer Zeit bzgl. der Größe von $\mathcal{J} \cup \{A_1 \to A_2\}$ beantworten.

Bemerkung 6.5 *Die oben eingeführten Notationen können leicht auf aussagenlogische Formeln übertragen werden. Betrachtet man die Objekte als aussagenlogische Variablen* $o_1, \ldots, o_n$, *so repräsentiert die aussagenlogische Formel* $\phi_{A_1 \to A_2} := \bigwedge_{o \in A_1} o \to \bigwedge_{o' \in A_2} o'$ *die Objektimplikation* $A_1 \to A_2$.
Sei nun Γ *die Menge der Formeln zu den Implikationen aus* $\mathcal{J}$. *Dann gilt* $A_1 \to A_2 \in Cons(\mathcal{L})$ *genau dann, wenn* $\phi_{A_1 \to A_2}$ *eine logische Folgerung aus* Γ *ist. Man sieht nun leicht, dass die linearen Entscheidungsverfahren für die Erfüllbarkeit von Mengen aussagenlogischer Hornklauseln [DG84] angewendet werden können, um zu entscheiden, ob* $A_1 \to A_2 \in Cons(\mathcal{J})$. *Damit bildet also eine Basis* $\mathcal{J}$ *von* $Imp_{\mathcal{O}}(\mathcal{K})$ *eine Repräsentation von* $Imp_{\mathcal{O}}(\mathcal{K})$, *die es erlaubt, jede Frage der Form "*$A_1 \to A_2 \in Imp_{\mathcal{O}}(\mathcal{K})$*?" in linearer Zeit bzgl. der Größe von* $\mathcal{J} \cup \{A_1 \to A_2\}$ *zu entscheiden.*
Es sei noch darauf hingewiesen, dass sich diese Komplexitätsaussage auch mit Algorithmen beweisen lässt, die zur Bestimmung funktionaler Abhängigkeiten in relationalen Datenbanken entwickelt wurden (siehe z.B. [Mai83], Abschnitt 4.6).

Es kann mehrere Implikationsbasen für die Menge $Imp_{\mathcal{O}}(\mathcal{K})$ geben, wobei nicht jede von minimaler Größe sein muss. Die *Größe* einer Basis $\mathcal{J}$ ist gegeben durch die Anzahl der Implikationen in $\mathcal{J}$. Eine Basis $\mathcal{J}$ von $Imp_{\mathcal{O}}(\mathcal{K})$ heißt *minimale Basis* genau dann, wenn es keine Basis von $Imp_{\mathcal{O}}(\mathcal{K})$ gibt, deren Größe kleiner ist als die Größe von $\mathcal{J}$. Duquenne hat in [Duq87] eine solche minimale Basis für die Menge der Merkmalsimplikationen beschrieben. Diese sogenannte *Duquenne-Guigues-Basis* wird von Ganters Merkmalsexplorationsalgorithmus als Nebenprodukt berechnet. Im folgenden wird nun die *duale Duquenne-Guigues-Basis* eingeführt, die ebenfalls als Nebenprodukt durch den Objektexplorationsalgorithmus berechnet wird.

Die Definition der dualen Duquenne-Guigues-Basis beruht auf einer Modifikation des durch eine Menge $\mathcal{J}$ von Objektimplikationen induzierten Hüllenoperators $A \mapsto \mathcal{J}(A) = A''$. Für eine Teilmenge A von $\mathcal{O}$ bezeichnet $\mathcal{J}^*(A)$ die *Implikations-Pseudohülle* von A bzgl. $\mathcal{J}$, d.h. die kleinste Teilmenge H von $\mathcal{O}$ mit

- $A \subseteq H$ und
- $A_1 \to A_2 \in \mathcal{J}$ und $A_1 \subset H$ (echte Teilmenge!) impliziert $A_2 \subseteq H$.

Wiederum kann für gegebenes $\mathcal{J}$ die Pseudohülle von $A \subseteq \mathcal{O}$ in linearer Zeit bzgl. der Größe von $\mathcal{J}$ und A berechnet werden (z.B. durch Anpassen des Algorithmus aus

[DG84]). Eine Teilmenge A von $\mathcal{O}$ heißt *pseudo-abgeschlossen* bzgl. des formalen Kontextes $\mathcal{K}$ genau, dann wenn $Imp_{\mathcal{O}}(\mathcal{K})^*(A) = A$ und $A'' \neq A$.

Definition 6.6 (Duale Duquenne-Guigues-Basis) *Die* duale Duquenne-Guigues-Basis *eines formalen Kontextes $\mathcal{K}$ ist definiert als die Menge aller Objektimplikationen $A_1 \to A_2$ für die $A_1 \subseteq \mathcal{O}$ pseudo-abgeschlossen bzgl. $\mathcal{K}$ und $A_2 = A_1'' \setminus A_1$ ist.*

Versucht man ausgehend von dieser Definition die duale Duquenne-Guigues-Basis eines Kontextes zu berechnen, so ergeben sich zwei Probleme:

1. Die Definition von pseudo-abgeschlossen bezieht alle gültigen Implikationen aus $Imp_{\mathcal{O}}(\mathcal{K})$ ein: Um diese alle zu berücksichtigen, bräuchte man aber eine Basis von $Imp_{\mathcal{O}}(\mathcal{K})$ und eine solche soll je gerade erst berechnet werden.

2. Man benötigt den Hüllenoperator $A \mapsto A''$: Bei einer unendlichen Menge von Merkmalen kann dieser aber im allgemeinen nicht durch $A \mapsto A' \mapsto A''$ berechnet werden.

Ganter hat das erste Problem gelöst, indem er die pseudo-abgeschlossenen Mengen in $\mathcal{K}$ bzgl. einer speziellen Ordnung, der sogenannten *lektischen Ordnung* aufzählt. Diese Ordnung stellt sicher, dass es zur Berechnung der Pseudo-Hülle stets ausreicht, den bereits berechneten Teil der Basis zu betrachten.

Definition 6.7 (Lektische Ordnung) *Sei $<$ eine beliebige aber feste lineare Ordnung auf der Menge von Objekten $\mathcal{O} = \{o_1, \ldots, o_n\}$, beispielsweise $o_1 < \ldots < o_n$. Für alle j mit $1 \leq j \leq n$ und alle $A_1, A_2 \subseteq \mathcal{O}$ definiere*

$$A_1 <_j A_2 \quad \textit{genau dann, wenn} \quad o_j \in A_2 \setminus A_1 \textit{ und} \\ A_1 \cap \{o_1, \ldots, o_{j-1}\} = A_2 \cap \{o_1, \ldots, o_{j-1}\}.$$

Die lektische Ordnung $<$ *ist definiert als die Vereinigung aller Relationen $<_j$, $j = 1, \ldots, n$.*

Man sieht leicht, dass die lektische Ordnung eine lineare Ordnung auf der Potenzmenge von $\mathcal{O}$ ist, wobei die lektisch kleinste Menge die leere Menge ist.

Das zweite Problem bekommt man in den Griff, indem man eine aufsteigende Kette von endlichen Subkontexten von $\mathcal{K}$ berechnet. Der Kontext $\mathcal{K}_i = (\mathcal{O}_i, \mathcal{M}_i, \mathcal{S}_i)$ heißt *Subkontext* von $\mathcal{K}$ genau dann, wenn $\mathcal{O}_i = \mathcal{O}$, $\mathcal{M}_i \subseteq \mathcal{M}$ und $\mathcal{S}_i = \mathcal{S} \cap (\mathcal{O} \times \mathcal{M}_i)$. Man berechnet dann den Hüllenoperator $A \mapsto A''$ stets bzgl. des zuletzt berechneten endlichen Subkontextes $\mathcal{K}_i$. Da in diesem Subkontext noch Implikationen gültig sein können, die in $\mathcal{K}$ nicht gültig sind, ist noch sicherzustellen, dass keine falschen Implikationen in die Basis aufgenommen werden. Zu diesem Zweck fragt man einen „Experten", ob eine Implikation $A \to A'' \setminus A$ auch tatsächlich in $\mathcal{K}$ gilt (und nicht nur im aktuellen Subkontext $\mathcal{K}_i$ bzgl. dem A'' berechnet wurde). Gilt eine solche Implikation nicht in $\mathcal{K}$, so muss der Experte ein Gegenbeispiel zurückliefern, d.h. ein Merkmal m aus $\mathcal{M} \setminus \mathcal{M}_i$ welches die Implikation verletzt. Der aktuelle Subkontext $\mathcal{K}_i$ wird dann um dieses Merkmal m erweitert. Daher muss der Experte nicht nur das Merkmal m liefern, sondern auch für alle Objekte angeben, ob sie m haben oder nicht.

Damit sind beide Probleme gelöst und der Objektexplorationsalgorithmus kann spezifiziert werden (s. Abbildung 6.2). Er berechnet die Extensionen aller formalen Konzepte in $\mathcal{K}$ sowie die duale Duquenne-Guigues-Basis von $\mathcal{K}$; den Konzeptverband erhält man durch die Mengeninklusionsbeziehungen zwischen den Extensionen.

6.3 Die Berechnung der LCS-Hierarchie

In diesem Abschnitt gilt es nun, für eine gegebene Menge $\mathcal{O} := \{C_1, \ldots, C_n\}$ von Konzeptbeschreibungen einen Kontext zu definieren, dessen Begriffsverband mit der Subsumtionshierarchie der zu den Teilmengen von $\mathcal{O}$ gehörenden LCS übereinstimmt. Unter Verwendung eines geeigneten Experten soll dieser Begriffsverband dann mit dem oben eingeführten Objektexplorationsalgorithmus berechnet werden können. Dazu bezeichne im folgenden $\mathsf{lcs}(A)$ für Teilmengen $A = \{D_1, \ldots, D_m\}$ von $\mathcal{O}$ den LCS von $D_1, \ldots, D_m$, wobei $\mathsf{lcs}(\emptyset) := \bot$ und $\mathsf{lcs}(\{D\}) := D$.

Definition 6.8 (Induzierter Kontext) *Sei $\mathcal{L}$ eine BL und $\mathcal{O} := \{C_1, \ldots, C_n\}$ eine endliche Menge von $\mathcal{L}$-Konzeptbeschreibungen. Der durch $\mathcal{O}$ induzierte formale Kontext $\mathcal{K}_\mathcal{L}(\mathcal{O}) = (\mathcal{O}, \mathcal{M}, \mathcal{S})$ ist definiert durch:*

$$\begin{aligned} \mathcal{O} &:= \{C_1, \ldots, C_n\}, \\ \mathcal{M} &:= \{D \mid D \text{ ist eine } \mathcal{L}\text{-Konzeptbeschreibung}\}, \\ \mathcal{S} &:= \{(C, D) \mid C \sqsubseteq D\}. \end{aligned}$$

Das folgende Lemma beschreibt den engen Zusammenhang zwischen der Intension einer Menge $A \subseteq \mathcal{O}$ und dem LCS dieser Menge.

Lemma 6.9 *Seien A_1, A_2 Teilmengen von $\mathcal{O}$. Dann gilt:*

1. $A_1' = \{D \in \mathcal{M} \mid \mathsf{lcs}(A_1) \sqsubseteq D\}$.

2. $A_1' \subseteq A_2'$ *genau dann, wenn* $\mathsf{lcs}(A_2) \sqsubseteq \mathsf{lcs}(A_1)$.

3. *Die Implikation $A_1 \to A_2$ ist in $\mathcal{K}_\mathcal{L}(\mathcal{O})$ genau dann gültig, wenn* $\mathsf{lcs}(A_2) \sqsubseteq \mathsf{lcs}(A_1)$.

Initialisierung: Starte mit

- der leeren Menge von Objektimplikationen: $\mathcal{J}_0 := \emptyset$;
- der leeren Menge von Extensionen: $\mathcal{E}_0 := \emptyset$;
- dem leeren Subkontext $\mathcal{K}_0$ von $\mathcal{K}$: $\mathcal{M}_0 := \emptyset$;
- der lektisch kleinsten Menge von Objekten: $A_0 := \emptyset$;

Iteration: Angenommen, $\mathcal{K}_i$, $\mathcal{J}_i$, $\mathcal{E}_i$ und A_i ($i \geq 0$) seien bereits berechnet.
Berechne A_i'' bzgl. $\mathcal{K}_i$.
Frage den Experten, ob die Implikation $A_i \to A_i'' \setminus A_i$ in $\mathcal{K}$ gültig ist.

Falls „nein“:
Sei $m_i \in \mathcal{M}$ das vom Experten gelieferte Gegenbeispiel.
Definiere
$A_{i+1} := A_i$;
$\mathcal{J}_{i+1} := \mathcal{J}_i$;
$\mathcal{K}_{i+1}$ als den Subkontext von $\mathcal{K}$ mit $\mathcal{M}_{i+1} := \mathcal{M}_i \cup \{m_i\}$.

Falls „ja“:
$\mathcal{K}_{i+1} := \mathcal{K}_i$;
Die neue Menge A_{i+1} bestimme wie folgt:

$$(\mathcal{E}_{i+1}, \mathcal{J}_{i+1}) := \begin{cases} (\mathcal{E}_i, \mathcal{J}_i \cup \{A_i \to A_i'' \setminus A_i\}) & \text{, falls } A_i'' \neq A_i, \\ (\mathcal{E}_i \cup \{A_i\}, \mathcal{J}_i) & \text{, falls } A_i'' = A_i. \end{cases}$$

$j := n$;
Teste $A_i <_j \mathcal{J}_{i+1}^*((A_i \cap \{o_1, \ldots, o_{j-1}\}) \cup \{o_j\})$ $(*)$
Setze $j := j - 1$ solange, bis einer der folgenden Fälle gilt:

1. $j = 0$: Dann ist $\mathcal{E}_{i+1}$ die Menge der Extensionen aller Konzepte in $\mathcal{K}$ und $\mathcal{J}_{i+1}$ ist die duale Duquenne-Guigues-Basis von $\mathcal{K}$. Der Algorithmus stoppt.
2. $(*)$ gilt für ein $j > 0$: Dann sei $A_{i+1} := \mathcal{J}_{i+1}^*((A_i \cap \{o_1, \ldots, o_{j-1}\}) \cup \{o_j\})$ und die Iteration wird fortgesetzt.

Abbildung 6.1: Der Objektexplorationsalgorithmus.

Beweis: *Ad (1):* Nach Definition von $\cdot'$ und $\mathcal{K}_\mathcal{L}(\mathcal{O})$ gilt für $A \subseteq \mathcal{O}$

$$A' = \{D \in \mathcal{M} \mid C \sqsubseteq D \text{ für alle } C \in A\}.$$

Zeige

$$\{D \in \mathcal{M} \mid C \sqsubseteq D \text{ für alle } C \in A\} = \{D \in \mathcal{M} \mid \mathsf{lcs}(A_1) \sqsubseteq D\}.$$

Aus $C \sqsubseteq D$ für alle $C \in A$ folgt nach Definition des LCS auch $\mathsf{lcs}(A) \sqsubseteq D$, also folgt „$\subseteq$".

Umgekehrt folgt wegen $C \sqsubseteq \mathsf{lcs}(A)$ für alle $C \in A$ aus $\mathsf{lcs}(A) \sqsubseteq D$ auch $C \sqsubseteq D$ für alle $C \in A$ und damit „$\supseteq$".

Ad (2): „$\Rightarrow$" Mit (1) folgt $\mathsf{lcs}(A_1) \in A_1'$. Mit der Voraussetzung $A_1' \subseteq A_2'$ folgt $\mathsf{lcs}(A_1) \in A_2'$ und wiederum mit (1) folgt $\mathsf{lcs}(A_2) \sqsubseteq \mathsf{lcs}(A_1)$.

„$\Leftarrow$" Sei $D \in A_1'$. Zu zeigen: $D \in A_2'$. Mit (1) folgt $\mathsf{lcs}(A_1) \sqsubseteq D$, woraus mit der Voraussetzung $\mathsf{lcs}(A_2) \sqsubseteq \mathsf{lcs}(A_1)$ auch $\mathsf{lcs}(A_2) \sqsubseteq D$ folgt. (1) liefert wiederum $D \in A_2'$.

Ad (3): Die Implikation $A_1 \to A_2$ gilt in $\mathcal{K}_\mathcal{L}(\mathcal{O})$
gdw für alle $D \in \mathcal{M}$ gilt $A \subseteq D' \Longrightarrow B \subseteq D'$
gdw für alle $D \in \mathcal{M}$ gilt
(für alle $C \in A_1$ gilt $C \sqsubseteq D$) $\Longrightarrow$ (für alle $C \in A_2$ gilt $C \sqsubseteq D$)
gdw für alle $D \in \mathcal{M}$ gilt $\mathsf{lcs}(A_1) \sqsubseteq D \Longrightarrow \mathsf{lcs}(A_2) \sqsubseteq D$
gdw $\mathsf{lcs}(A_2) \sqsubseteq \mathsf{lcs}(A_1)$. □

Aus der dritten Aussage des Lemmas folgt unmittelbar, dass sich aus der dualen Duquenne-Guigues-Basis $\mathcal{J}$ von $\mathcal{K}_\mathcal{L}(\mathcal{O})$ alle Subsumtionsbeziehungen der Form $\mathsf{lcs}(A_2) \sqsubseteq \mathsf{lcs}(A_1)$ für Teilmengen A_1, A_2 aus $\mathcal{O}$ ableiten lassen. Genauer: Bei gegebener Basis $\mathcal{J}$ lassen sich alle Fragen der Form "$\mathsf{lcs}(A_1) \sqsubseteq \mathsf{lcs}(A_2)$?" in linearer Zeit (bzgl. der Größe von $\mathcal{J} \cup \{A_1 \to A_2\}$) beantworten. Die wesentliche Konsequenz aus dem Lemma ist aber, dass der Begriffsverband $\mathcal{K}_\mathcal{L}(\mathcal{O})$ mit der Subsumtionshierarchie der LCS aller Teilmengen von $\mathcal{O}$ übereinstimmt.

Theorem 6.10 *Der Konzeptverband zu $\mathcal{K}_\mathcal{L}(\mathcal{O})$ ist isomorph zur Subsumtionshierarchie der LCS aller Teilmengen von $\mathcal{O}$.*

Beweis: Für den Nachweis der Isomorphie betrachten wir aus technischen Gründen nicht die LCS von Teilmengen $A \subseteq \mathcal{O}$, sondern ihre Äquivalenzklassen $[\mathsf{lcs}(A)]_\equiv = \{D \mid D \text{ ist } \mathcal{L}\text{-Konzeptbeschreibung und } \mathsf{lcs}(A) \equiv D\}$. Die durch die Subsumtion in $\mathcal{L}$ induzierte Ordnung $\sqsubseteq_\equiv$ auf den Äquivalenzklassen $[\mathsf{lcs}(A)]_\equiv$ ist definiert durch $[\mathsf{lcs}(A_1)]_\equiv \sqsubseteq_\equiv [\mathsf{lcs}(A_2)]_\equiv$ genau dann, wenn $\mathsf{lcs}(A_1) \sqsubseteq \mathsf{lcs}(A_2)$.

Wir definieren nun eine Abbildung π von den formalen Konzepten in $\mathcal{K}_\mathcal{L}(\mathcal{O})$ in die Menge der (Äquivalenzklassen von) LCS von Teilmengen von $\mathcal{O}$ durch:

$$\pi(A, B) := [\mathsf{lcs}(A)]_\equiv.$$

Nun gilt für formale Konzepte $(A_1, B_1), (A_2, B_2)$, dass $(A_1, B_1) \leq (A_2, B_2)$ genau dann, wenn $A_1 \subseteq A_2$, was wiederum genau dann gilt, wenn $A'_2 \subseteq A'_1$. Mit Lemma 6.9.2 folgt daraus $\mathsf{lcs}(A_1) \sqsubseteq \mathsf{lcs}(A_2)$. Also ist π ordnungserhaltend (und damit auch injektiv): Es gilt $(A_1, B_1) \leq (A_2, B_2)$ genau dann, wenn $[\mathsf{lcs}(A_1)]_\equiv \sqsubseteq_\equiv [\mathsf{lcs}(A_2)]_\equiv$.

Es bleibt zu zeigen, dass π auch surjektiv ist. Sei dazu A eine beliebige Teilmenge von $\mathcal{O}$. Zu zeigen: $[\mathsf{lcs}(A)]_\equiv$ taucht im Bild von π auf. Mit Lemma 6.2 folgt, dass (A'', A') ein formales Konzept ist. Also genügt es zu zeigen, dass $\mathsf{lcs}(A) \equiv \mathsf{lcs}(A'')$. Offensichtlich impliziert $A \subseteq A''$, dass $\mathsf{lcs}(A) \sqsubseteq \mathsf{lcs}(A'')$ gilt. Umgekehrt sei $C_i \in \mathcal{O}$ beliebig. Dann gilt

$$\begin{array}{lll} C_i \in A'' & \text{gdw} & C_i \sqsubseteq D \text{ für alle } D \in A' \qquad \text{(Def. von } \cdot' \text{ und } \mathcal{K}_\mathcal{L}(\mathcal{O})) \\ & \text{gdw} & C_i \sqsubseteq D \text{ für alle } D \text{ mit } \mathsf{lcs}(A) \sqsubseteq D \qquad \text{(Lemma 6.9)} \\ & \text{gdw} & C_i \sqsubseteq \mathsf{lcs}(A). \qquad \text{(Def. des LCS)} \end{array}$$

Offensichtlich folgt daraus $\mathsf{lcs}(A'') \sqsubseteq \mathsf{lcs}(A)$. □

Um nun den Algorithmus aus Abbildung 6.2 tatsächlich zur Berechnung des Begriffsverbandes anwenden zu können, benötigen wir noch einen „Experten“ für den formalen Kontext $\mathcal{K}_\mathcal{L}(\mathcal{O})$. Dieser Experte muss in der Lage sein,

1. für eine gegebene Implikation $A_1 \to A_2$ zu entscheiden, ob sie in $\mathcal{K}_\mathcal{L}(\mathcal{O})$ gilt,
2. falls eine solche Implikation nicht gilt, ein Gegenbeispiel zu liefern, d.h. eine Konzeptbeschreibung $D \in A'_1 \setminus A'_2$, und
3. für ein Gegenbeispiel die zugehörige Spalte in der Matrix zur Inzidenzrelation zu liefern.

Im nächsten Schritt wird gezeigt, dass ein solcher Experte für alle BLen $\mathcal{L}$ implementiert werden kann, in denen der LCS berechenbar und Subsumtion entscheidbar ist, insbesondere also für $\mathcal{ALE}$.

Lemma 6.11 *Ist für eine BL $\mathcal{L}$ ein Subsumtionsalgorithmus sowie ein Algorithmus zur Berechnung des LCS in $\mathcal{L}$ gegeben, so kann ein Experte für den Kontext $\mathcal{K}_\mathcal{L}(\mathcal{O})$ implementiert werden.*

Beweis: Es ist zu zeigen, dass die oben eingeführten Punkte 1. und 2. unter Verwendung der vorausgesetzten Algorithmen realisiert werden können.

Sei also $A_1 \to A_2$ eine im Objektexplorationsalgorithmus generierte Implikation. Mit Lemma 6.9 folgt, dass $A_1 \to A_2$ in $\mathcal{K}_\mathcal{L}(\mathcal{O})$ genau dann gilt, wenn $\mathsf{lcs}(A_2) \sqsubseteq \mathsf{lcs}(A_1)$. Offensichtlich gilt aber $\mathsf{lcs}(A_2) \sqsubseteq \mathsf{lcs}(A_1)$ genau dann, wenn $C_i \sqsubseteq \mathsf{lcs}(A_1)$ für alle $C_i \in A_2$. Um also die Frage, ob "$A_1 \to A_2$" in $\mathcal{K}_\mathcal{L}(\mathcal{O})$ gilt, zu beantworten, genügt es, $\mathsf{lcs}(A_1)$ zu berechnen und dann mit dem Subsumtionsalgorithmus für alle $C_i \in A_2$

zu testen, ob $C_i \sqsubseteq \mathsf{lcs}(A_1)$ gilt oder nicht. Gilt dies für ein $C_i \in A_2$ nicht, so gilt auch die Implikation in $\mathcal{K}_\mathcal{L}(\mathcal{O})$ nicht.

Für den zweiten Punkt sei $A_1 \to A_2$ eine Implikation, die in $\mathcal{K}_\mathcal{L}(\mathcal{O})$ nicht gilt, d.h. $\mathsf{lcs}(A_2) \not\sqsubseteq \mathsf{lcs}(A_1)$. Dann ist $\mathsf{lcs}(A_1)$ ein Gegenbeispiel, d.h. $\mathsf{lcs}(A_1) \in A_1'$ und $\mathsf{lcs}(A_1) \notin A_2'$. Dies folgt unmittelbar aus

(*i*) $A_i' = \{D \in \mathcal{M} \mid \mathsf{lcs}(A_i) \sqsubseteq D\}$, $i = 1, 2$, (s. Lemma 6.9(1)) und

(*ii*) $\mathsf{lcs}(A_2) \not\sqsubseteq \mathsf{lcs}(A_1)$.

Wie oben bereits erwähnt, benötigt der Explorationsalgorithmus zu diesem Gegenbeispiel auch die zugehörige Spalte in der Matrix zur Inzidenzrelation $\mathcal{S}$. Diese Spalte kann leicht unter Verwendung des gegebenen Subsumtionsalgorithmus berechnet werden, indem man für jedes $C_i \in \mathcal{O}$ entscheidet, ob $C_i \sqsubseteq \mathsf{lcs}(A)$ gilt oder nicht. □

Unter Verwendung des im Beweis von Lemma 6.11 beschriebenen Experten liefert nun der Objektexplorationsalgorithmus aus Abbildung 6.2 angewendet auf eine Menge $\mathcal{O}$ von $\mathcal{L}$-Konzeptbeschreibungen und den induzierten formalen Kontext $\mathcal{K}_\mathcal{L}(\mathcal{O})$ das folgende:

- die Extensionen aller formalen Konzepte in $\mathcal{K}_\mathcal{L}(\mathcal{O})$ und damit den Konzeptverband von $\mathcal{K}_\mathcal{L}(\mathcal{O})$, der gemäß Theorem 6.10 mit der Subsumtionshierarchie der LCS aller Teilmengen von $\mathcal{O}$ übereinstimmt;
- die duale Duquenne-Guigues-Basis zu $\mathcal{K}_\mathcal{L}(\mathcal{O})$, die mit Lemma 6.9(3) eine kompakte Repräsentation dieser Subsumtionshierarchie liefert; und schließlich
- einen endlichen Subkontext von $\mathcal{K}_\mathcal{L}(\mathcal{O})$, dessen formalen Konzepte die gleichen Extensionen haben wie die formalen Konzepte in $\mathcal{K}_\mathcal{L}(\mathcal{O})$ und der die gleichen Hüllen bzgl. $\cdot''$ auf Teilmengen von $\mathcal{O}$ liefert.

Objektexploration in der Anwendung

Die Ergebnisse der vorherigen Abschnitte zeigen, dass sich der Objektexplorationsalgorithmus prinzipiell zur Berechnung der Subsumtionshierarchie der LCS eignet. Insbesondere erfüllt die im Rahmen der Prozesstechnikanwendung eingesetzte BL $\mathcal{ALE}$ die Voraussetzungen von Lemma 6.11, sodass dort die LCS-Hierarchie unter Verwendung der Objektexploration berechnet werden kann. Es bleibt aber noch zu analysieren, ob es sich hierbei auch tatsächlich um einen sinnvollen Ansatz zur Lösung dieser Aufgabe handelt. Ein eher generisches Argument für diesen Ansatz sind die positiven Erfahrungen, die mit dem Ganter-Algorithmus bei der Berechnung von Begriffsverbänden in vielen Anwendungen gemacht wurden [Gan91, Stu96b, GW99]. Das Problem mit diesem generischen Argument ist natürlich, dass man es in der Prozesstechnikanwendung mit einem sehr speziellen Kontext zu tun hat, für den sich durchaus herausstellen könnte, dass die Objektexploration nicht einer der günstigsten Ansätze zur Berechnung des Begriffsverbandes ist. Erste Experimente haben aber vielversprechende Resultate geliefert (cf Kapitel 9).

6.4 Angrenzende Arbeiten

Die Idee, Methoden aus der Formalen Begriffsanalyse im Bereich BL einzusetzen, war bereits für frühere Arbeiten grundlegend. So wurde in [Baa95] der Merkmalsexplorationsalgorithmus eingesetzt, um möglichst kleine Repräsentationen der Subsumtionshierarchie aller Konjunktionen definierter Konzeptnamen zu berechnen. Dieser Ansatz wurde in [Stu96a] auf Konjunktionen und Disjunktionen definierter Konzeptnamen ausgedehnt. Zu dem oben vorgestellten Ansatz bestehen dabei aber zwei signifikante Unterschiede:

1. Der in [Baa95, Stu96a] definierte formale Kontext unterscheidet sich stark von dem für die Berechnung der LCS Hierarchie definierten Kontext: Seine Objekte sind Paare bestehend aus einer endlichen Interpretation $\mathcal{I} = (\Delta_\mathcal{I}, \cdot^\mathcal{I})$ und einem Individuum $\alpha \in \Delta_\mathcal{I}$. Für die Realisierung des Experten im Explorationsalgorithmus musste der Subsumtionsalgorithmus so erweitert werden, dass er solche Objekte als Gegenbeispiele liefern konnte.

2. In den Arbeiten [Baa95, Stu96a] bestand das Ziel ausschließlich in der Berechnung der Duquenne-Guigues Basis und nicht in der Berechnung oder Visualisierung des Konzeptverbandes. Tatsächlich hat eine Implementierung und anschließende Evaluierung des Ansatzes aus [Baa95] im Rahmen einer Diplomarbeit [Nas97] gezeigt, dass (i) die erhaltenen Konzeptverbände meist sehr groß sind und daher die Visualisierung des Konzeptverbandes zur Untersuchung der Subsumtionshierarchie ungeeignet ist, aber (ii) die zugehörigen Duquenne-Guigues Basen ziemlich klein sind und daher schnelle Antworten auf Subsumtionsanfragen erlauben.

Kapitel 7

Most Specific Concepts

Im Unterschied zum LCS existiert das MSC zu einem Individuum a bzgl. einer $\mathcal{ALE}$-ABox $\mathcal{A}$ im allgemeinen nicht (cf Abschnitt 7.1). Grund für die Nicht-Existenz sind zyklische Abhängigkeiten zwischen ABox-Individuen. Dieser Tatsache wird im Rahmen dieser Arbeit Rechnung getragen, indem nur Approximationen des MSC in $\mathcal{ALE}$ untersucht werden. Abschnitt 7.2 erläutert die dabei auftretenden Probleme zunächst für die BL $\mathcal{EL}$: Aufbauend auf eine geeignete Charakterisierung der Instanz in $\mathcal{EL}$ wird ein Algorithmus zur Berechnung einer sogenannten *k-Approximation* des MSC eingeführt, der im azyklischen Fall sogar genau das MSC liefert. Erweitert man $\mathcal{EL}$ um primitive Negation (im folgenden mit $\mathcal{EL}_\neg$ bezeichnet), so liefert dieser Algorithmus leider nicht mehr das Gewünschte. Durch eine Anpassung der Charakterisierung der Instanz auf $\mathcal{EL}_\neg$ kann dann aber auch der Approximationsalgorithmus entsprechend angepasst werden (cf Abschnitt 7.3). Dieser Schritt gelingt jedoch für unsere Zielsprache $\mathcal{ALE}$ nicht mehr: Abschnitt 7.4 weist zwar Existenz und Berechenbarkeit der k-Approximation des MSC in $\mathcal{ALE}$ nach, greift dabei aber auf einen inpraktikablen Algorithmus zurück. Die Entwicklung eines effizienteren Algorithmus wie für $\mathcal{EL}$ und $\mathcal{EL}_\neg$ bleibt bei $\mathcal{ALE}$ ein offenes Problem.

7.1 Ansätze zur Berechnung des MSC

Das MSC zu einem Individuum a bzgl. einer $\mathcal{ALE}$-ABox $\mathcal{A}$ existiert im allgemeinen nicht: zyklische Abhängigkeiten zwischen ABox-Individuen können dazu führen, dass es unendlich viele Konzeptbeschreibungen gibt, von denen a Instanz ist, darunter aber keine speziellste solche Konzeptbeschreibung. Das folgende Beispiel illustriert diese Situation für $\mathcal{ALN}$ und $\mathcal{ALE}$.

Beispiel 7.1 *Betrachte die $\mathcal{ALN}$-ABox $\mathcal{A} = \{a : P, a : (\leq 1\ r), (a, a) : r\}$. In [BK98] wird gezeigt, dass für diese ABox kein MSC von a in $\mathcal{ALN}$ existiert: Man sieht leicht,* *dass a Instanz ist von*

$$C_n := \underbrace{\forall r.\cdots\forall r}_{n\ mal}.(P \sqcap (\leq 1\ r) \sqcap (\geq 1\ r))$$

für jedes $n \geq 0$. Intuitiv ist damit das MSC von a bzgl. $\mathcal{A}$ äquivalent zu der unendlichen Konjunktion $\bigsqcap_{n\geq 0} C_n$. Diese kann aber offensichtlich nicht durch eine äquivalente $\mathcal{ALN}$-Konzeptbeschreibung dargestellt werden, d.h. das MSC von a bzgl. $\mathcal{A}$ existiert nicht in $\mathcal{ALN}$.
Für $\mathcal{ALE}$ und die Teilsprachen $\mathcal{EL}$ und $\mathcal{EL}_\neg$ leistet bereits die ABox $\mathcal{A}' = \{a : P, (a,a) : r\}$ das Gewünschte: Man sieht leicht, dass für

$$C_n := \underbrace{\exists r.\cdots\exists r}_{n\ mal}.P$$

$a \in_{\mathcal{A}'} C_n$ für alle $n \geq 0$. Angenommen, es existiert eine $\mathcal{ALE}$-/$\mathcal{EL}$-/$\mathcal{EL}_\neg$-Konzeptbeschreibung C mit $\mathsf{msc}_{\mathcal{A}'}(a) \equiv C$. Sei $depth(C) = k$ und $\mathcal{G}_C$ der $\mathcal{ALE}$-Beschreibungsbaum zur $\mathcal{ALE}$-Normalform von C. Dann existiert offensichtlich kein Homomorphismus von $\mathcal{G}(C_{k+1})$ nach $\mathcal{G}_C$. Mit Theorem 5.21 folgt $C \not\sqsubseteq C_{k+1}$ im Widerspruch zu $a \in_{\mathcal{A}'} C_{k+1}$ und zur Annahme $C \equiv \mathsf{msc}_{\mathcal{A}'}(a)$.

Für $\mathcal{ALN}$ finden sich in der Literatur zwei Ansätze, mit diesem Problem umzugehen: (1) man begnügt sich mit einer Approximation des MSC oder (2) man erweitert die BL auf zyklische Konzeptbeschreibungen, um das MSC exakt repräsentieren zu können.
Der erste, pragmatische Ansatz wurde z.B. in [CBH92, CH94a, CH94b, FP96] für $\mathcal{ALN}$ verfolgt. Die Idee besteht darin, zunächst die Menge der Kandidaten für eine Approximation des MSC zu beschränken: Statt aller $\mathcal{ALN}$-Konzeptbeschreibungen betrachtet man nur Konzeptbeschreibungen, deren Tiefe kleiner gleich einer beliebigen aber festen Schranke k ist. Aus den Ergebnissen in [BK98] folgt leicht, dass die speziellste Konzeptbeschreibung mit Tiefe kleiner gleich k, von der a bzgl. $\mathcal{A}$ Instanz ist, (bis auf Äquivalenz) eindeutig bestimmt ist.[1] Diese Approximation des MSC kann aber exponentiell groß bzgl. k und der ABox sein. Die Bestimmung der worst-case-Komplexität für ihre Berechnung ist noch ein offenes Problem (wobei stark zu vermuten ist, dass sie stets in exponentieller Zeit berechnet werden kann).
Der zweite Ansatz wird in [BK98] betrachtet. Die Autoren führen eine Charakterisierung des MSC mit *zyklischen* Konzeptbeschreibungen ein. Diese zyklischen Konzeptbeschreibungen werden durch zyklische TBoxen definiert und durch die größte Fixpunkt Semantik (gfp-Semantik) interpretiert (s. [Küs98] für eine automatentheoretische Charakterisierung der gfp-Semantik in $\mathcal{ALN}$). Für die $\mathcal{ALN}$-ABox aus Beispiel 7.1 liefert beispielsweise das durch die zyklische $\mathcal{ALN}$-TBox

$$\mathcal{T} = \{C \doteq P \sqcap (\leq 1\ r) \sqcap (\geq 1\ r) \sqcap \forall r.C\}$$

[1] Eine entsprechende Behauptung wird auch in [CH94b] formuliert, dort aber nicht bewiesen.

definierte Konzept C das MSC von a bzgl. $\mathcal{A}$ [BK98]. Die von Baader und Küsters eingeführte Charakterisierung des MSC basiert auf einer automatentheoretischen Repräsentation von zyklischen Konzeptbeschreibungen. Der aus dieser Charakterisierung abgeleitete Algorithmus berechnet das MSC allerdings nur in doppelt exponentieller Zeit (im worst case). Vermutet wird jedoch, dass das Instanzproblem für zyklische $\mathcal{ALN}$-Konzeptbeschreibungen in PSPACE entschieden und das MSC in exponentieller Zeit berechnet werden kann.

Für $\mathcal{ALE}$ sind zyklische Konzeptbeschreibungen und mögliche Semantiken noch nicht gründlich untersucht. Damit ist offen, ob sich der oben vorgestellte zweite Ansatz auf $\mathcal{ALE}$ übertragen lässt, d.h. ob sich das MSC durch zyklische $\mathcal{ALE}$-Konzeptbeschreibungen unter Verwendung der gfp-Semantik darstellen lässt. Um dennoch zum einen einen Eindruck von den Problemen bei der Berechnung des MSC und zum anderen erste Erfahrungen mit MSCs in der Anwendung zu bekommen, wurde im Rahmen dieser Arbeit der pragmatische Ansatz für $\mathcal{ALE}$ verfolgt, d.h. Approximationen des MSC in $\mathcal{ALE}$ untersucht.

Definition 7.2 *Sei $\mathcal{A}$ eine $\mathcal{L}$-ABox, a ein Individuum in $\mathcal{A}$ und $k \in \mathbb{N}$. Eine $\mathcal{L}$-Konzeptbeschreibung C heißt* k-Approximation von a bzgl. $\mathcal{A}$ *(kurz $C = \mathsf{msc}_{\mathcal{A},k}(a)$), wenn*

1. *$a \in_{\mathcal{A}} C$,*
2. *$depth(C) \leq k$ und*
3. *für alle $\mathcal{L}$-Konzepte D mit $depth(D) \leq k$ und $a \in_{\mathcal{A}} D$ gilt $C \sqsubseteq D$.*

Als das wesentliche Ergebnis für $\mathcal{ALE}$ ist festzuhalten, dass sich bei endlicher Signatur und Beschränkung der Tiefe durch eine Schranke k Existenz und Berechenbarkeit der k-Approximation für a bzgl. $\mathcal{A}$ nachweisen lässt (cf Abschnitt 7.4). Dabei ergibt sich die Existenz unmittelbar aus der Tatsache, dass es bei endlicher Signatur bis auf Äquivalenz nur endlich viele verschiedene $\mathcal{ALE}$-Konzeptbeschreibungen der Tiefe kleiner gleich k gibt[2] und $\mathcal{ALE}$ Konjunktion von Konzepten zuläßt. Allerdings basiert der Nachweis der Berechenbarkeit auf dem nicht praktikablen Algorithmus, der alle diese Konzepte aufzählt und die Konjunktion der bzgl. Subsumtion minimalen Konzepte C zurückgibt, die die Instanzbeziehung $a \in_{\mathcal{A}} C$ erfüllen.

Ein Ziel dieses Kapitels ist daher die Entwicklung eines praktikablen Algorithmus zur Berechnung der k-Approximation. Ein intuitiver Ansatz hierfür besteht in der Übertragung der Vorgehensweise bei der Berechnung des LCS auf die hier betrachtete Situation: Die Charakterisierung des LCS (und damit auch der Algorithmus zu seiner Berechnung) ergibt sich auf natürliche Weise aus der Charakterisierung der

[2] Man beachte, dass diese Aussage für $\mathcal{ALN}$ nicht gilt: trotz endlicher Signatur gibt es unendlich viele paarweise nicht äquivalente Zahlenrestriktionen und damit auch $\mathcal{ALN}$-Konzeptbeschreibungen.

Subsumtion mit Hilfe von Beschreibungsbäumen und Homomorphismen. Dieser Vorgehensweise folgend sucht man also eine Charakterisierung der Instanz, die einen zielgerichteten Algorithmus zur Berechnung der k-Approximation nahelegt.

Der dazu in der vorliegenden Arbeit verfolgte Ansatz ist der folgende: Die Instanzbeziehung $a \in_{\mathcal{A}} C$ wird durch einen Homomorphismus vom Beschreibungsbaum zu C in einen *Beschreibungsgraphen* $\mathcal{G}(\mathcal{A})$ zu $\mathcal{A}$ charakterisiert, wobei die Wurzel von C auf a abzubilden ist. Diese Charakterisierung ist für $\mathcal{ALE}$ korrekt, d.h. analog zu Lemma 5.22 werden wir zeigen, dass $a \in_{\mathcal{A}} C$ gilt, wenn ein solcher Homomorphismus von $\mathcal{G}(C)$ nach $\mathcal{G}(\mathcal{A})$ existiert (cf Abschnitt 7.4). Allerdings ist sie lediglich für $\mathcal{EL}$ auch vollständig (cf Abschnitt 7.2). Tatsächlich ist sie sogar schon für $\mathcal{EL}_\neg$ nicht mehr vollständig. Grund hierfür ist, dass durch die Konzeptassertionen einer $\mathcal{EL}_\neg$-ABox im allgemeinen nicht für alle Individuen $a \in \mathit{Ind}(\mathcal{A})$ und alle $P \in N_C$ festgelegt ist, ob $a \in_{\mathcal{A}} P$ oder $a \in_{\mathcal{A}} \neg P$. Dieses Problem bekommt man aber in den Griff, indem man bei der Charakterisierung statt des $\mathcal{EL}_\neg$-Graphen $\mathcal{G}(\mathcal{A})$ alle sogenannten *primitiven Vervollständigungen* von $\mathcal{G}(\mathcal{A})$ berücksichtigt (cf Abschnitt 7.3). Eine primitive Vervollständigung ist eine Erweiterung von $\mathcal{G}(\mathcal{A})$, in der jedes Knotenlabel jeden Konzeptnamen $P \in N_C$ entweder positiv oder negiert enthält. Die resultierende Charakterisierung der Instanz ist dann zwar für $\mathcal{EL}_\neg$, nicht aber für $\mathcal{ALE}$ vollständig. Jedoch können die für $\mathcal{EL}$ und $\mathcal{EL}_\neg$ gewonnenen Erkenntnisse als Ausgangspunkt für eine korrekte und vollständige Charakterisierung der Instanz in $\mathcal{ALE}$ dienen. Deren vermutlich sehr komplexe Spezifikation hätte aber den Rahmen dieser Arbeit gesprengt und wird daher Gegenstand zukünftiger Arbeiten (insbesondere am Lehr- und Forschungsgebiet Theoretische Informatik) sein.

Wie sich nun die soeben skizzierte Charakterisierung der Instanz zur Berechnung der k-Approximation des MSC einsetzen lässt, ist für die kleine BL $\mathcal{EL}$ noch „naheliegend“: Man

1. nehme den Beschreibungsgraphen $\mathcal{G}_{\mathcal{A}}$,
2. rolle ihn zu einem (im allgemeinen unendlichen) Baum mit Wurzel a ab,
3. schneide diesen auf Tiefe k ab und
4. übersetze diesen endlichen Baum in eine Konzeptbeschreibung.

Unter Verwendung der Charakterisierung der Instanz folgt dann leicht, dass die so erhaltene Konzeptbeschreibung die speziellste mit Tiefe kleiner gleich k ist, von der a Instanz ist. Tritt a in einer azyklischen Zusammenhangskomponente der ABox auf, so ist der durch das Abrollen erhaltene Baum endlich und die zugehörige Konzeptbeschreibung liefert sogar genau das MSC zu a.

Da für die Charakterisierung der Instanz in $\mathcal{EL}_\neg$ alle möglichen Vervollständigungen der ABox zu berücksichtigen sind, gilt entsprechendes für die Charakterisierung der k-Approximation: Abschnitt 7.3 zeigt, dass das Produkt der $\mathcal{EL}_\neg$-Graphen zu allen Vervollständigungen einen Beschreibungsgraphen liefert, für den wieder der oben skizzierte Algorithmus die k-Approximation des MSC in $\mathcal{EL}_\neg$ liefert.

Existenz und Berechenbarkeit der k-Approximation des MSC wird also in $\mathcal{EL}$ und $\mathcal{EL}_\neg$ für beliebige Signaturen bewiesen. Da die für $\mathcal{EL}_\neg$ entwickelte Charakterisierung der Instanz für $\mathcal{ALE}$ nicht vollständig ist, leistet auch die Übertragung des verbesserten Algorithmus auf $\mathcal{ALE}$ leider nicht das Gewünschte. Für die aus einem $\mathcal{ALE}$-Beschreibungsgraphen $\mathcal{G}(\mathcal{A})$ durch abrollen, abschneiden und übersetzen erhaltene $\mathcal{ALE}$-Konzeptbeschreibung zu a kann lediglich gezeigt werden, dass a von diesem Konzept Instanz ist, nicht aber, dass es das speziellste Konzept mit Tiefe kleiner gleich k mit dieser Eigenschaft ist. Für die Anwendung besteht hierbei die Hoffnung, dass bereits diese „Approximation der Approximation“ eine sinnvolle Unterstützung liefern kann.

7.2 Most Specific Concepts in $\mathcal{EL}$

Der erste Unterabschnitt führt zunächst die Charakterisierung der Instanz in $\mathcal{EL}$ ein, die für den Nachweis der Korrektheit des in Unterabschnitt 7.2.2 entwickelten Algorithmus zur Berechnung der k-Approximation des MSC benötigt wird.

7.2.1 Die Charakterisierung der Instanz in $\mathcal{EL}$

Die nun vorgestellte Charakterisierung der Instanz in $\mathcal{EL}$ kann als Erweiterung der Charakterisierung der Subsumtion in $\mathcal{EL}$ aus Theorem 5.4 gesehen werden: Wir werden zeigen, dass sich Instanzbeziehungen durch Homomorphismen zwischen $\mathcal{EL}$-Beschreibungsbäumen und $\mathcal{EL}$-Graphen charakterisieren lassen.

Ein *$\mathcal{EL}$-Graph* $\mathcal{G} = (V, E, \ell)$ ist ein beschrifteter Graph mit $E \subseteq V \times N_R \times V$ und $\ell(v) \subseteq N_C$ für alle $v \in V$. Den Ausgangspunkt für die Definition des $\mathcal{EL}$-Graphen $\mathcal{G}(\mathcal{A})$ zu einer $\mathcal{EL}$-ABox $\mathcal{A}$ bildet die graphische Repräsentation von $\mathcal{A}$, bei der Individuen die Knoten und Rollenassertionen die Kanten bilden. Konzeptassertionen der Form $a : C$ werden wie folgt berücksichtigt: die Konzeptnamen auf Toplevel von C bilden das Label von a und zu jeder Existenzrestriktion $\exists r.C'$ auf Toplevel von C wird der $\mathcal{EL}$-Beschreibungsbaum $\mathcal{G}(C') = (V_{C'}, E_{C'}, v_0', \ell_{C'})$ zusammen mit der Kante arv_0' zum Graphen hinzugenommen. Durch diese Konstruktion ergibt sich beispielsweise für die $\mathcal{EL}$-ABox

$$\mathcal{A} \;=\; \{a : P \sqcap \exists s.(Q \sqcap \exists r.P \sqcap \exists s.\top), b : P \sqcap Q, c : \exists r.P, (a,b) : r, (a,c) : r, (b,c) : s\}$$

der $\mathcal{EL}$-Graph in Abbildung 7.1. Formal wird $\mathcal{G}(\mathcal{A})$ wie folgt definiert:

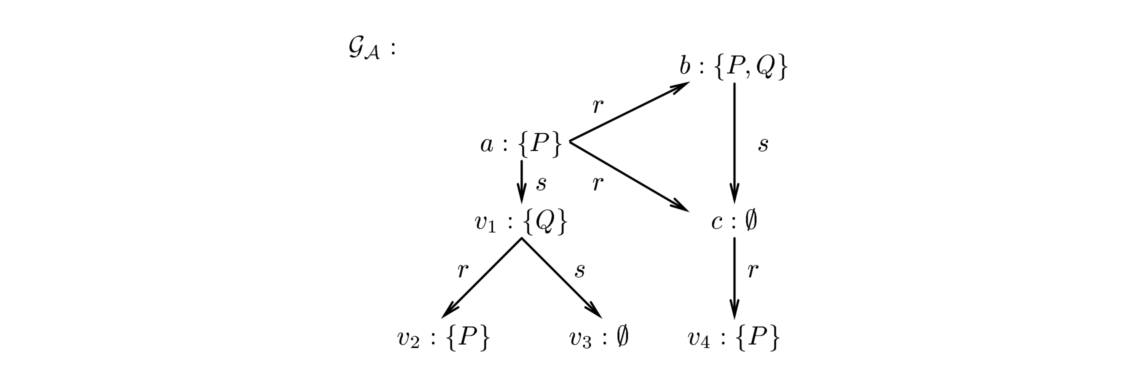


Abbildung 7.1: Der $\mathcal{EL}$-Graph zur $\mathcal{EL}$-ABox $\mathcal{A}$.

Definition 7.3 ($\mathcal{EL}$-Graph einer $\mathcal{EL}$-ABox)
Sei $\mathcal{A}$ eine $\mathcal{EL}$-ABox. Für alle $a \in Ind(\mathcal{A})$ sei $C_a := \bigsqcap_{a:D\in\mathcal{A}} D$, falls eine Konzeptassertion der Form $a : D$ in $\mathcal{A}$ existiert; sonst sei $C_a := \top$. Die $\mathcal{EL}$-Beschreibungsbäume zu C_a, $a \in Ind(\mathcal{A})$, werden mit $\mathcal{G}(C_a) = (V_a, E_a, a, \ell_a)$ bezeichnet, wobei die Mengen V_a o.B.d.A. paarweise disjunkt seien.
Der $\mathcal{EL}$-Graph $\mathcal{G}(\mathcal{A})$ zu $\mathcal{A}$ *ist definiert durch* $\mathcal{G}(\mathcal{A}) := (V, E, \ell)$ *mit*

- $V := \bigcup_{a\in Ind(\mathcal{A})} V_a$,
- $E := \{arb \mid (a, b) : r \in \mathcal{A}\} \cup \bigcup_{a\in Ind(\mathcal{A})} E_a$ *und*
- $\ell(v) := \ell_a(v)$ *für* $v \in V_a$.

Für die Charakterisierung der Instanz bleibt noch der Begriff des Homomorphismus auf $\mathcal{EL}$-Graphen anzupassen: Eine Abbildung $\varphi : V_C \longrightarrow V$ ist ein *Homomorphismus von* $\mathcal{G}(C) = (V_C, E_C, w_0, \ell_C)$ *nach* $\mathcal{G}_\mathcal{A} = (V, E, \ell)$, wenn φ die Bedingungen (2) und (3) aus Definition 5.3 (die Bedingungen an Knotenlabel und Kanten) erfüllt. Die erste Bedingung aus Definition 5.3 (die Bedingung an die Wurzel) geht nun direkt in die Charakterisierung der Instanz ein.

Theorem 7.4 *Seien $\mathcal{A}$ eine $\mathcal{EL}$-ABox mit $\mathcal{G}(\mathcal{A}) = (V, E, \ell)$ und C eine $\mathcal{EL}$-Konzeptbeschreibung mit $\mathcal{G}(C) = (V_C, E_C, w_0, \ell_C)$. Für $a \in Ind(\mathcal{A})$ gilt $a \in_\mathcal{A} C$ genau dann, wenn ein Homomorphismus φ von $\mathcal{G}(C)$ nach $\mathcal{G}(\mathcal{A})$ existiert mit $\varphi(w_0) = a$.*

Beweis der Rückrichtung: Wie bereits in Abschnitt 7.1 erwähnt, ist diese Charakterisierung auch für $\mathcal{ALE}$ korrekt (aber nicht vollständig), d.h. auch für $\mathcal{ALE}$ gilt die Rückrichtung von Theorem 7.4. Lemma 7.28 in Abschnitt 7.4 formuliert die entsprechende Behauptung für $\mathcal{ALE}$. Aus diesem Lemma folgt die Rückrichtung von Theorem 7.4 trivial als Spezialfall.

Beweis der Hinrichtung: Dieser baut auf die durch $\mathcal{G}(\mathcal{A})$ induzierte kanonische Interpretation $\mathcal{I}(\mathcal{G}(\mathcal{A}))$ auf.

Definition 7.5 (Kanonische Interpretation zum $\mathcal{EL}$-Graph)
Sei $\mathcal{G} = (V, E, \ell)$ ein $\mathcal{EL}$-Graph. Die kanonische Interpretation $\mathcal{I}(\mathcal{G})$ *ist definiert durch $\mathcal{I}(\mathcal{G}) := (\Delta_{\mathcal{I}(\mathcal{G})}, \cdot^{\mathcal{I}(\mathcal{G})})$ mit*

- $\Delta_{\mathcal{I}(\mathcal{G})} := V$,
- $P^{\mathcal{I}(\mathcal{G})} := \{v \in V \mid P \in \ell(v)\}$ *für alle* $P \in N_C$ *und*
- $r^{\mathcal{I}(\mathcal{G})} := \{(v, w) \in V \times V \mid (vrw \in E)\}$ *für alle* $r \in N_R$.

Das folgende Lemma formuliert die für den Beweis der Hinrichtung entscheidende Eigenschaft der kanonischen Interpretation.

Lemma 7.6 *Sei $\mathcal{A}$ eine $\mathcal{EL}$-ABox mit $\mathcal{EL}$-Graph $\mathcal{G}(\mathcal{A})$. Dann ist die kanonische Interpretation $\mathcal{I}(\mathcal{G}(\mathcal{A}))$ ein Modell von $\mathcal{A}$.*

Beweis: Zu zeigen: $\mathcal{I}(\mathcal{G}(\mathcal{A}))$ erfüllt jede Assertion in $\mathcal{A}$.
Offensichtlich erfüllt $\mathcal{I}(\mathcal{G}(\mathcal{A}))$ nach Konstruktion jede Rollenassertion aus $\mathcal{A}$.
Sei $a : D$ eine Konzeptassertion aus $\mathcal{A}$ mit $D = P_1 \sqcap \ldots \sqcap P_n \sqcap \exists r_1.D_1 \sqcap \ldots \sqcap \exists r_m.D_m$, $P_i \in N_C$. Zeige

1. $a^{\mathcal{I}(\mathcal{G}(\mathcal{A}))} \in P_i^{\mathcal{I}(\mathcal{G}(\mathcal{A}))}$ für alle $1 \leq i \leq n$ und
2. $a^{\mathcal{I}(\mathcal{G}(\mathcal{A}))} \in (\exists r_j.D_j)^{\mathcal{I}(\mathcal{G}(\mathcal{A}))}$ für alle $1 \leq j \leq m$.

Ad (1): Nach Definition von $\mathcal{G}(\mathcal{A})$ ist $P_i \in \ell(a)$ und nach Definition von $\mathcal{I}(\mathcal{G}(\mathcal{A}))$ ist damit $a^{\mathcal{I}(\mathcal{G}(\mathcal{A}))} \in P_i^{\mathcal{I}(\mathcal{G}(\mathcal{A}))}$ für alle $1 \leq i \leq n$.
Ad (2): Für jedes $1 \leq j \leq m$ existiert nach Definition von $\mathcal{G}(\mathcal{A})$ ein $v_j \in V$ mit $ar_jv_j \in E$ und $D_j \equiv C_{\mathcal{G}(C_a)(v_j)}$. Offensichtlich ist D_j in $\mathcal{ALE}$-Normalform. Damit stimmt $\mathcal{I}(\mathcal{G}(\mathcal{A}))$ eingeschränkt auf die Knoten in $\mathcal{G}(C_a)$ mit der kanonischen Interpretation $\mathcal{I}(\mathcal{G}(C_a))$ von $\mathcal{G}(C_a)$ überein (s. Definition 5.24). Lemma 5.25 impliziert $v_j^{\mathcal{I}(\mathcal{G}(\mathcal{A}))} \in (C_{\mathcal{G}(C_a)(v_j)})^{\mathcal{I}(\mathcal{G}(\mathcal{A}))}$. Zusammen mit $(a^{\mathcal{I}(\mathcal{G}(\mathcal{A}))}, v_j^{\mathcal{I}(\mathcal{G}(\mathcal{A}))}) \in r_j^{\mathcal{I}(\mathcal{G}(\mathcal{A}))}$ folgt daraus $a^{\mathcal{I}(\mathcal{G}(\mathcal{A}))} \in (\exists r_j.D_j)^{\mathcal{I}(\mathcal{G}(\mathcal{A}))}$.
Aus (1) und (2) folgt $a^{\mathcal{I}(\mathcal{G}(\mathcal{A}))} \in D^{\mathcal{I}(\mathcal{G}(\mathcal{A}))}$. □

Mit $a \in_{\mathcal{A}} C$ und $\mathcal{I}(\mathcal{G}(\mathcal{A})) \models \mathcal{A}$ folgt nun die Existenz eines Homomorphismus φ von $\mathcal{G}(C)$ nach $\mathcal{G}(\mathcal{A})$ mit $\varphi(w_0) = a$ unmittelbar aus der folgenden

Behauptung: Sei $v \in V$. Ist $v \in C^{\mathcal{I}(\mathcal{G}(\mathcal{A}))}$, so existiert ein Homomorphismus φ von $\mathcal{G}(C)$ nach $\mathcal{G}(\mathcal{A})$ mit $\varphi(w_0) = v$.

Beweis der Behauptung durch Induktion nach $depth(C)$:

Induktionsanfang: $depth(C) = 0$, d.h. $C = P_1 \sqcap \ldots \sqcap P_n$. Dann ist $\mathcal{G}(C) = (\{w_0\}, \emptyset, w_0, \ell_C)$ mit $\ell_C(w_0) = \{P_1, \ldots, P_n\}$. Definiere φ durch $\varphi(w_0) := v$. Nach Annahme ist $v \in C^{\mathcal{I}(\mathcal{G}(\mathcal{A}))}$. Damit ist nach Definition von $\mathcal{I}(\mathcal{G}(\mathcal{A}))$ auch $\ell_C(w_0) \subseteq \ell(v)$; somit ist φ ein Homomorphismus.

Induktionsschritt: $depth(C) > 0$, d.h. $C = P_1 \sqcap \ldots \sqcap P_n \sqcap \exists r_1.C_1 \sqcap \ldots \sqcap \exists r_m.C_m$. Wie im Induktionsanfang folgt $\{P_1, \ldots, P_n\} \subseteq \ell(v)$. Aus $v \in C^{\mathcal{I}(\mathcal{G}(\mathcal{A}))}$ folgt außerdem, dass für alle $1 \leq i \leq m$ ein $v_i \in V$ existiert mit $(v, v_i) \in r_i^{\mathcal{I}(\mathcal{G}(\mathcal{A}))}$ und $v_i \in C_i^{\mathcal{I}(\mathcal{G}(\mathcal{A}))}$. Sei jeweils w_i der r_i-Nachfolger von w_0 in $\mathcal{G}(C)$ mit $C_i = \mathcal{G}(C)(w_i)$. Per Induktion folgt, dass Homomorphismen $\varphi_{w_1}, \ldots, \varphi_{w_m}$ existieren, die jeweils $\mathcal{G}(C)(w_i)$ in $\mathcal{G}(\mathcal{A})$ homomorph einbetten mit $\varphi_i(w_i) = v_i$. Definiere nun

$$\varphi := \{v_0 \mapsto v\} \cup \bigcup_{w_0 r w \in E_C} \varphi_w.$$

Nach Konstruktion ist φ ein Homomorphismus von $\mathcal{G}(C)$ nach $\mathcal{G}(\mathcal{A})$ mit $\varphi(w_0) = v$. □

Komplexität des Instanzproblems in $\mathcal{EL}$

Ob ein Baum in einen Graphen homomorph eingebettet werden kann, kann in polynomieller Zeit entschieden werden [GJ79, Rey77]. Ein polynomieller Algorithmus, der entscheidet, ob ein Homomorphismus φ von einem $\mathcal{EL}$-Beschreibungsbaum $\mathcal{G}(C)$ in einen $\mathcal{EL}$-Beschreibungsgraphen $\mathcal{G}(\mathcal{A})$ existiert mit $\varphi(w_0) = a$ für ein $a \in Ind(\mathcal{A})$ findet sich z.B. in [BMT98]. Da $\mathcal{G}(\mathcal{A})$ und $\mathcal{G}(C)$ für eine $\mathcal{EL}$-ABox $\mathcal{A}$ und eine $\mathcal{EL}$-Konzeptbeschreibung C in polynomieller Zeit erzeugt werden können, erhalten wir

Satz 7.7 *Das Instanzproblem in $\mathcal{EL}$ ist polynomiell entscheidbar.*

7.2.2 Die k-Approximation des MSC in $\mathcal{EL}$

Das Ziel dieses Abschnitts ist der Nachweis von Existenz und Berechenbarkeit der k-Approximation des MSC in $\mathcal{EL}$. Dazu wird ein Algorithmus spezifiziert, der durch direkte Manipulation des $\mathcal{EL}$-Graphen zur Eingabe-ABox $\mathcal{A}$ diese k-Approximation für Individuen $a \in Ind(\mathcal{A})$ berechnet. Er basiert auf der folgenden Idee: Durch „abrollen“ des $\mathcal{EL}$-Graphen $\mathcal{G}(\mathcal{A})$ erhält man zunächst einen Baum $\mathcal{T}(a, \mathcal{G}(\mathcal{A}))$ mit Wurzel a, endlichem Verzweigungsgrad aber möglicherweise unendlich langen Pfaden (cf Definition 7.8). Durch Abschneiden der längeren Pfade bestimmt man einen $\mathcal{EL}$-Beschreibungsbaum $\mathcal{T}_k(a, \mathcal{G}(\mathcal{A}))$ der Tiefe kleiner gleich k. Mit der Charakterisierung der Instanz folgt dann leicht, dass die $\mathcal{EL}$-Konzeptbeschreibung $C_{\mathcal{T}_k(a,\mathcal{G}(\mathcal{A}))}$ die

k-Approximation von a bzgl. $\mathcal{A}$ ist. Ist $\mathcal{A}$ azyklisch, so ist $\mathcal{T}(a, \mathcal{G}(\mathcal{A}))$ ein endlicher Baum und $C_{\mathcal{T}(a,\mathcal{G}(\mathcal{A}))}$ ist das MSC von a bzgl. $\mathcal{A}$.

Wie aus der Modelltheorie bekannt, erhält man zu einem Graphen mit einem ausgezeichneten Knoten v einen Baum mit Wurzel v, indem man die Menge aller Pfade, die von v ausgehen, als Knotenmenge betrachtet. Für einen $\mathcal{EL}$-Graphen $\mathcal{G} = (V, E, \ell)$ ist $p = v_0 r_1 v_1 r_2 \ldots r_n v_n$ ein *Pfad von* v_0 *nach* v_n der *Länge* $|p| = n$, wenn $v_{i-1} r_i v_i \in E$ für alle $1 \leq i \leq n$. Der Pfad p wird auch als $r_1 \cdots r_n$*-Pfad von* v_0 *nach* v_n bezeichnet und der Knoten v_n als $r_1 \cdots r_n$*-Nachfolger von* v_0, wobei jeder Knoten ε-Nachfolger von sich selber ist. Der Pfad p enthält einen *Zykel*, wenn $v_i = v_j$ für Indizes i, j mit $0 \leq i < j \leq n$. Ein Knoten v ist von v_0 aus *erreichbar*, wenn ein Pfad von v_0 nach v existiert.

Definition 7.8 (Baum zu a bzgl. $\mathcal{G}$ (und k)) *Sei $\mathcal{G} = (V, E, \ell)$ und $a \in V$. Der* Baum $\mathcal{T}(a, \mathcal{G})$ zu a bzgl. $\mathcal{A}$ *ist definiert durch $\mathcal{T}(a, \mathcal{G}) := (V^t, E^t, a, \ell^t)$ mit*

- $V^t := \{a r_1 v_1 r_2 \ldots r_n v_n \mid a r_1 v_1 r_2 \ldots r_n v_n \text{ ist Pfad von } a \text{ nach } v_n \text{ in } \mathcal{G}\}$,
- $E^t := \{prq \mid p, q \in V^t \text{ und } p = a r_1 v_1 r_2 \ldots r_n v_n \text{ und } q = a r_1 v_1 r_2 \ldots r_n v_n r w\}$,
- $\ell^t(p) := \ell(v)$ *für* $p = a r_1 v_1 r_2 \ldots r_n v$.

Für $k \in \mathbb{N}$ ist der Baum $\mathcal{T}_k(a, \mathcal{G})$ zu a bzgl. $\mathcal{G}$ und k *definiert durch $\mathcal{T}_k(a, \mathcal{G}) := (V^t_k, E^t_k, a, \ell^t_k)$ mit*

- $V^t_k := \{p \in V^t \mid |p| \leq k\}$,
- $E^t_k := E^t \cap (V^t_k \times N_R \times V^t_k)$ *und*
- $\ell^t_k(p) := \ell^t(p)$ *für* $p \in V^t_k$.

Da $\mathcal{EL}$-Graphen nach Definition endlich sind, hat $\mathcal{T}(a, \mathcal{G})$ endlichen Verzweigungsgrad, d.h. $\mathcal{T}(a, \mathcal{G})$ ist genau dann unendlich groß, wenn $\mathcal{T}(a, \mathcal{G})$ einen unendlich langen Pfad enthält. Dies ist wiederum genau dann der Fall, wenn in $\mathcal{G}$ ein Pfad existiert, der einen Zykel enthält. Desweiteren ist nach Definition $\mathcal{T}_k(a, \mathcal{G})$ stets ein $\mathcal{EL}$-Beschreibungsbaum der Tiefe kleiner gleich k. Ist $\mathcal{G}$ azyklisch, d.h. in $\mathcal{G}$ existiert kein Pfad, der einen Zykel enthält, so ist $\mathcal{T}(a, \mathcal{G})$ endlich. Die folgende Charakterisierung (der k-Approximation) des MSC verwendet nun diese Bäume, wobei als $\mathcal{EL}$-Graph der $\mathcal{EL}$-Graph einer ABox $\mathcal{A}$ und als Wurzel ein Individuum $a \in \mathit{Ind}(\mathcal{A})$ betrachtet wird. Dabei heißt eine ABox $\mathcal{A}$ *azyklisch*, wenn $\mathcal{G}(\mathcal{A})$ als gerichteter Graph azyklisch ist.

Theorem 7.9 *Seien $\mathcal{A}$ eine $\mathcal{EL}$-ABox, $a \in \mathit{Ind}(\mathcal{A})$ und $k \in \mathbb{N}$. Dann ist $C_{\mathcal{T}_k(a,\mathcal{G}(\mathcal{A}))}$ die k-Approximation von a bzgl. $\mathcal{A}$. Ist $\mathcal{A}$ azyklisch, so ist $C_{\mathcal{T}(a,\mathcal{G}(\mathcal{A}))}$ das Most Specific Concept von a bzgl. $\mathcal{A}$.*

Beweis: Seien $\mathcal{G}(\mathcal{A}) = (V_\mathcal{A}, E_\mathcal{A}, \ell_\mathcal{A})$ und $\mathcal{T}_k(a, \mathcal{G}(\mathcal{A})) = (V_k, E_k, a, \ell_k)$.
Zeige zunächst $a \in_\mathcal{A} C_{\mathcal{T}_k(a,\mathcal{G}(\mathcal{A}))}$. Sei dazu φ die Abbildung, die jeden Pfad $p \in V_k$ auf den jeweils letzten Knoten v in p abbildet. Man sieht leicht, dass φ ein Homomorphismus von $\mathcal{T}_k(a, \mathcal{G}(\mathcal{A}))$ nach $\mathcal{G}(\mathcal{A})$ ist mit $\varphi(a) = a$. Da $\mathcal{G}(C_{\mathcal{T}_k(a,\mathcal{G}(\mathcal{A}))})$ bis auf Knotenumbenennung gleich $\mathcal{T}_k(a, \mathcal{G}(\mathcal{A}))$ ist (Eigenschaft (5.2)), folgt mit Theorem 7.4 die Behauptung.
Es bleibt zu zeigen, dass für alle $\mathcal{EL}$-Konzeptbeschreibungen C mit $depth(C) \leq k$ und $a \in_\mathcal{A} C$ gilt: $C_{\mathcal{T}_k(a,\mathcal{G}(\mathcal{A}))} \sqsubseteq C$.
Sei also C eine $\mathcal{EL}$-Konzeptbeschreibung mit $depth(C) \leq k$ und $a \in_\mathcal{A} C$ und sei $\mathcal{G}(C) = (V_C, E_C, v_0, \ell_C)$. Mit Theorem 7.4 existiert ein Homomorphismus φ von $\mathcal{G}(C)$ nach $\mathcal{G}(\mathcal{A})$ mit $\varphi(v_0) = a$. Unter Verwendung von φ definieren wir einen Homomorphismus ψ von $\mathcal{G}(C)$ nach $\mathcal{T}_k(a, \mathcal{G}(\mathcal{A}))$. Mit Theorem 5.4 folgt $C_{\mathcal{T}_k(a,\mathcal{G}(\mathcal{A}))} \sqsubseteq C$.
Zur Definition von ψ: Sei $v \in V_C$ und $v_0 r_1 v_1 r_2 \cdots r_{n-1} v_{n-1} r_n v$ der eindeutige (!) Pfad von v_0 nach v in $\mathcal{G}(C)$. Definiere $p(v) := \varphi(v_0) r_1 \varphi(v_1) r_2 \cdots r_{n-1} \varphi(v_{n-1}) r_n \varphi(v)$. Da φ ein Homomorphismus von $\mathcal{G}(C)$ nach $\mathcal{G}(\mathcal{A})$ ist, ist $p(v)$ wohldefiniert und bildet einen Pfad der Länge $n \leq k$ von $\varphi(v_0)$ nach $\varphi(v)$ in $\mathcal{G}(\mathcal{A})$. Man sieht nun leicht, dass die Abbildung $\psi : V_C \longrightarrow V_k$ mit $\psi(v) := p(v)$ einen Homomorphismus von $\mathcal{G}(C)$ nach $\mathcal{T}_k(a, \mathcal{G}(\mathcal{A}))$ liefert.
Ist $\mathcal{A}$ azyklisch, so ist $\mathcal{T}(a, \mathcal{G}(\mathcal{A}))$ endlich, also ein $\mathcal{EL}$-Beschreibungsbaum. Wie für $\mathcal{T}_k(a, \mathcal{G}(\mathcal{A}))$ folgt $a \in_\mathcal{A} C_{\mathcal{T}(a,\mathcal{G}(\mathcal{A}))}$ und $C_{\mathcal{T}(a,\mathcal{G}(\mathcal{A}))} \sqsubseteq C$ für alle $\mathcal{EL}$-Konzeptbeschreibungen C mit $a \in_\mathcal{A} C$. Also ist $C_{\mathcal{T}(a,\mathcal{G}(\mathcal{A}))}$ das MSC von a bzgl. $\mathcal{A}$. □

Korollar 7.10 *Für eine $\mathcal{EL}$-ABox $\mathcal{A}$, ein Individuum $a \in Ind(\mathcal{A})$ und $k \in \mathbb{N}$ existiert die k-Approximation $\mathsf{msc}_{\mathcal{A},k}(a)$ von a bzgl. $\mathcal{A}$ stets und ist effektiv berechenbar.*
Ist $\mathcal{A}$ azyklisch, so existiert das Most Specific Concept $\mathsf{msc}_\mathcal{A}(a)$ stets und ist effektiv berechenbar.

Komplexität (der k-Approximation) des MSC in $\mathcal{EL}$

Hier interessiert man sich im wesentlichen für die Größe (der k-Approximation) des MSC bzgl. der Größe der ABox und des Parameters k. Dabei ist die Größe $|\mathcal{A}|$ einer ABox $\mathcal{A}$ definiert durch

$$|\mathcal{A}| := |Ind(\mathcal{A})| + |\{(a,b) : r \mid (a,b) : r \in \mathcal{A}\}| + \sum_{a:C \in \mathcal{A}} |C|.$$

Beispiel 7.11 *Sei $\mathcal{A} = \{(a,a) : r, (a,a) : s\}$. Abbildung 7.2 zeigt den $\mathcal{EL}$-Graphen $\mathcal{G}(\mathcal{A})$ sowie die $\mathcal{EL}$-Beschreibungsbäume $\mathcal{T}_k(a, \mathcal{G}(\mathcal{A}))$ für $k = 1$ und $k = 2$. Man sieht leicht, dass $\mathcal{T}_k(a, \mathcal{G}(\mathcal{A}))$ jeweils dem vollen binären Baum der Tiefe k entspricht, wobei jeder Knoten mit der leeren Menge beschriftet ist und jeder Knoten (außer den Blättern) genau einen r-Nachfolger und genau einen s-Nachfolger hat. Mit Theorem 7.9 folgt, dass $C_{\mathcal{T}_k(a,\mathcal{G}(\mathcal{A}))}$ die k-Approximation von a bzgl. $\mathcal{A}$ ist. Die Größe von*

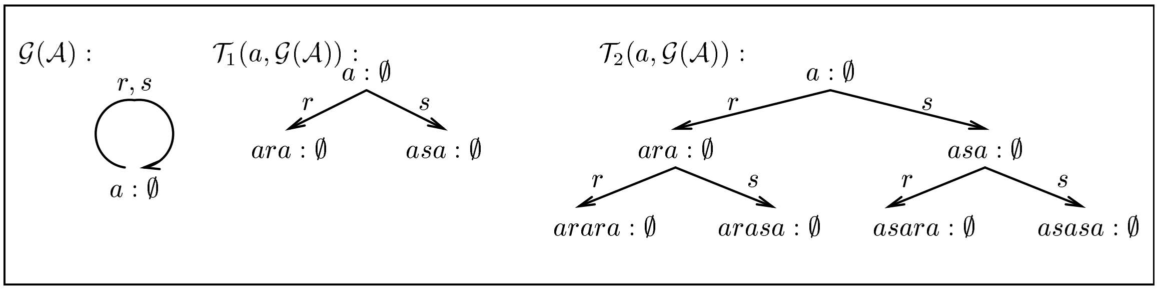


Abbildung 7.2: Beispiele für $\mathcal{EL}$-Beschreibungsbäume der Form $\mathcal{T}_k(a, \mathcal{G}(\mathcal{A}))$.

$C_{\mathcal{T}_k(a,\mathcal{G}(\mathcal{A}))}$ ist exponentiell in der Größe von $\mathcal{A}$ und k. Ähnlich wie im Beweis von Lemma 5.12 kann man zeigen, dass es keine $\mathcal{EL}$-Konzeptbeschreibung gibt, die äquivalent zu $C_{\mathcal{T}_k(a,\mathcal{G}(\mathcal{A}))}$ aber kleiner ist.

Das obige Beispiel zeigt, dass die k-Approximation von a bzgl. $\mathcal{A}$ exponentiell groß bzgl. des Parameters k sein kann (bei konstanter Größe der ABox). Das folgende Beispiel zeigt, dass auch das MSC von a bzgl. $\mathcal{A}$ exponentiell groß bzgl. der Größe der ABox sein kann.

Beispiel 7.12 *Für $n \geq 1$ definiere*

$$\mathcal{A}_n := \{(a_i, a_{i+1}) : r, (a_i, a_{i+1}) : s \mid 1 \leq i < n\}.$$

Offensichtlich ist $\mathcal{A}_n$ azyklisch und die Größe von $\mathcal{A}_n$ jeweils linear in n. Mit Theorem 7.9 folgt, dass $C_{\mathcal{T}(a_1,\mathcal{A}_n)}$ das MSC von a_1 bzgl. $\mathcal{A}_n$ ist. Man sieht leicht, dass $\mathcal{T}(a_1, \mathcal{G}(\mathcal{A}_n))$ mit dem Baum $\mathcal{T}_n(a, \mathcal{G}(\mathcal{A}))$ aus Beispiel 7.11 übereinstimmt. Wie dort folgt, dass $C_{\mathcal{T}(a,\mathcal{G}(\mathcal{A}_n))}$ exponentiell groß bzgl. n (also auch bzgl. der Größe von $\mathcal{A}_n$) ist und dass keine kleinere $\mathcal{EL}$-Konzeptbeschreibung C äquivalent zu $C_{\mathcal{T}(a,\mathcal{G}(\mathcal{A}_n))}$ existiert.

Nach oben kann die Größe der k-Approximation bzw. für azyklische ABoxen des MSC durch die Größe von $\mathcal{T}_k(a, \mathcal{G}(\mathcal{A}))$ bzw. durch die Größe von $\mathcal{T}(a, \mathcal{G}(\mathcal{A}))$ beschränkt werden. Die Größe von $\mathcal{T}_k(a, \mathcal{G}(\mathcal{A}))$ ist exponentiell in k beschränkt: Der Verzweigungsgrad ist kleiner gleich der Anzahl der Rollenassertionen in $\mathcal{A}$ und damit kleiner gleich $|\mathcal{A}| =: m_0$, wobei o.B.d.A. $|\mathcal{A}| > 1$ sei. Bei Beschränkung der Tiefe durch k ist die Anzahl der Knoten also kleiner gleich $\sum_{i=0}^{k} m_0^i = (m_0^{k+1} - 1)/(m_0 - 1)$. Jedes Knotenlabel enthält höchstens so viele Konzeptnamen wie in $\mathcal{A}$ auftreten, ist also kleiner gleich m_0. Damit ist die Größe von $\mathcal{T}_k(a, \mathcal{G}(\mathcal{A}))$ kleiner gleich m_0^{k+2}, also exponentiell in k beschränkt. Für eine azyklische ABox ist die Tiefe von $\mathcal{T}(a, \mathcal{G}(\mathcal{A}))$ beschränkt durch die Anzahl der Rollenassertionen in $\mathcal{A}$, also ist die Größe von $\mathcal{T}(a, \mathcal{G}(\mathcal{A}))$ exponentiell in der Größe von $\mathcal{A}$ beschränkt.
Insgesamt erhalten wir

Satz 7.13 *Für eine $\mathcal{EL}$-ABox $\mathcal{A}$, $a \in Ind(\mathcal{A})$ und $k \in \mathbb{N}$ kann*

1. *die k-Approximation $\mathsf{msc}_{\mathcal{A},k}(a)$ von a bzgl. $\mathcal{A}$ (höchstens) exponentiell groß in k und der Größe von $\mathcal{A}$ sein; und*

2. *das Most Specific Concept $\mathsf{msc}_{\mathcal{A}}(a)$ von a bzgl. $\mathcal{A}$ (höchstens) exponentiell groß in der Größe von $\mathcal{A}$ sein.*

7.3 Most Specific Concepts in $\mathcal{EL}_\neg$

Eine 1–1-Übertragung der Resultate von $\mathcal{EL}$ nach $\mathcal{EL}_\neg$ ist nicht möglich: Zunächst hat man aufgrund der primitiven Negation inkonsistente ABoxen als Spezialfall zu berücksichtigen. Aber auch für konsistente $\mathcal{EL}_\neg$-ABoxen führt die „naive" Übertragung der Charakterisierung der Instanz und der k-Approximation des MSC nicht zu einer vollständigen Charakterisierung der Instanz und nicht zur k-Approximation des MSC in $\mathcal{EL}_\neg$. Unterabschnitt 7.3.1 charakterisiert zunächst Konsistenz von $\mathcal{EL}_\neg$-ABoxen und stellt dann eine korrekte und vollständige Charakterisierung der Instanz vor, die auf dem Begriff der *primitiven Vervollständigung* basiert. Diese Charakterisierung bildet den Ausgangspunkt für die in Unterabschnitt 7.3.2 formulierte Charakterisierung (der k-Approximation) des MSC in $\mathcal{EL}_\neg$.

7.3.1 Die Charakterisierung der Instanz in $\mathcal{EL}_\neg$

Wir zeigen zunächst, das Konsistenz von $\mathcal{EL}_\neg$-ABoxen in polynomieller Zeit entschieden werden kann. Dazu sind einige Begriffe auf $\mathcal{EL}_\neg$ zu übertragen: *$\mathcal{EL}_\neg$-Beschreibungsbäume* und *$\mathcal{EL}_\neg$-Graphen* sind jeweils wie für $\mathcal{EL}$ definiert mit dem Unterschied, dass Knoten nun mit Teilmengen von $N_C \cup \{\neg P \mid P \in N_C\}$ beschriftet werden. Auch der *$\mathcal{EL}_\neg$-Beschreibungsbaum* $\mathcal{G}(C)$ zur $\mathcal{EL}_\neg$-Konzeptbeschreibung C und der *$\mathcal{EL}_\neg$-Graph* zur $\mathcal{EL}_\neg$-ABox $\mathcal{A}$ sind wie in $\mathcal{EL}$ definiert, wobei negierte Konzeptnamen wie Konzeptnamen behandelt werden. Schließlich wird die Definition der kanonischen Interpretation $\mathcal{I}(\mathcal{G})$ (s. Definition 7.5) 1–1 für $\mathcal{EL}_\neg$-Graphen übernommen, d.h. negierte Konzeptnamen werden bei der Definition der Interpretation eines Konzeptnamens P durch $P^{\mathcal{I}(\mathcal{G})} := \{v \mid P \in \ell(v)\}$ nicht berücksichtigt.

Nun kann Konsistenz von $\mathcal{EL}_\neg$-ABoxen wie folgt charakterisiert werden:

Lemma 7.14 *Eine $\mathcal{EL}_\neg$-ABox $\mathcal{A}$ ist genau dann inkonsistent, wenn ein $a \in \mathit{Ind}(\mathcal{A})$ existiert mit $C_a = \bigsqcap\limits_{a:D\in\mathcal{A}} D \equiv \bot$.*

Beweis: Sei $\mathcal{A}$ eine $\mathcal{EL}_\neg$-ABox. Existiert ein $a \in \mathit{Ind}(\mathcal{A})$ mit $C_a \equiv \bot$, so ist $\mathcal{A}$ offensichtlich inkonsistent. Sei umgekehrt $C_a \not\equiv \bot$ für alle $a \in \mathit{Ind}(\mathcal{A})$. Wir zeigen, dass dann die kanonische Interpretation $\mathcal{I}(\mathcal{G}(\mathcal{A}))$ ein Modell von $\mathcal{A}$ ist.

Man sieht leicht, dass eine $\mathcal{EL}_\neg$-Konzeptbeschreibung C genau dann inkonsistent ist, wenn im $\mathcal{EL}_\neg$-Beschreibungsbaum $\mathcal{G}(C) = (V_C, E_C, v_0, \ell_C)$ ein Knoten $v \in V_C$ existiert mit $\{P, \neg P\} \subseteq \ell_C(v)$ für ein $P \in N_C$. Sonst liefert die kanonische Interpretation $\mathcal{I}(\mathcal{G}(C))$ ein Modell von C: Denn existiert kein $v \in V_C$ mit $\{P, \neg P\} \subseteq \ell_C(v)$ für ein $P \in N_C$, so ist C in $\mathcal{ALE}$-Normalform und mit Lemma 5.25 folgt, dass $\mathcal{I}(\mathcal{G}(C))$ Modell von C ist.

Aus der Voraussetzung $C_a \not\equiv \bot$ für alle $a \in \mathit{Ind}(\mathcal{A})$ folgt also, dass kein Knoten v in $\mathcal{G}(\mathcal{A}) = (V, E, \ell)$ existiert mit $\{P, \neg P\} \subseteq \ell(v)$ für ein $P \in N_C$. Zu zeigen: $\mathcal{I}(\mathcal{G}(\mathcal{A}))$ erfüllt jede Assertion in $\mathcal{A}$. Nach Definition von $\mathcal{I}(\mathcal{G}(\mathcal{A}))$ ist jede Rollenassertion aus $\mathcal{A}$ erfüllt. Sei $a : D \in \mathcal{A}$ mit $D = Q_1 \sqcap \ldots \sqcap Q_n \sqcap \exists r_1.D_1 \sqcap \ldots \sqcap \exists r_m.D_m$ und $Q_i \in N_C \cup \{\neg P \mid P \in N_C\}$ für alle $1 \leq i \leq n$. Zeige

1. $a^{\mathcal{I}(\mathcal{G}(\mathcal{A}))} \in Q_i^{\mathcal{I}(\mathcal{G}(\mathcal{A}))}$ für alle $1 \leq i \leq n$ und

2. $a^{\mathcal{I}(\mathcal{G}(\mathcal{A}))} \in (\exists r_j.D_j)^{\mathcal{I}(\mathcal{G}(\mathcal{A}))}$ für alle $1 \leq j \leq m$.

Ad (1): Angenommen, $Q_i \in N_C$. Dann folgt nach Definition $a^{\mathcal{I}(\mathcal{G}(\mathcal{A}))} \in Q_i^{\mathcal{I}(\mathcal{G}(\mathcal{A}))}$. Andernfalls sei $Q_i = \neg P$ für ein $P \in N_C$. Mit der Voraussetzung folgt dann $P \notin \ell(a)$, d.h. es existiert keine Konzeptassertion $a : D' \in \mathcal{A}$ sodass P in der Konjunktion auf Toplevel von D' auftritt. Nach Definition $a^{\mathcal{I}(\mathcal{G}(\mathcal{A}))} \notin P^{\mathcal{I}(\mathcal{G}(\mathcal{A}))}$. Also ist $a^{\mathcal{I}(\mathcal{G}(\mathcal{A}))} \in Q_i^{\mathcal{I}(\mathcal{G}(\mathcal{A}))}$ für alle $1 \leq i \leq n$.

Ad (2): Mit der Voraussetzung an C_a folgt, dass jedes D_j konsistent ist. Insbesondere ist keine der $\mathcal{ALE}$-Normalisierungsregeln auf D_j anwendbar, d.h. D_j ist in $\mathcal{ALE}$-Normalform. Wie im Beweis von Lemma 7.6 folgt nun die Behauptung. □

Als einfache Konsequenz aus Lemma 7.14 und seinem Beweis erhalten wir die folgenden Aussagen.

Lemma 7.15 *Konsistenz von $\mathcal{EL}_\neg$-ABoxen ist in polynomieller Zeit entscheidbar.*

Lemma 7.16 *Ist $\mathcal{A}$ eine konsistente $\mathcal{EL}_\neg$-ABox, so ist $\mathcal{I}(\mathcal{G}(\mathcal{A}))$ ein Modell von $\mathcal{A}$.*

Inkonsistente $\mathcal{EL}_\neg$-ABoxen sind als Spezialfall bei der Charakterisierung der Instanz zu behandeln: Ist $\mathcal{A}$ inkonsistent, so gilt $a \in_{\mathcal{A}} C$ trivialerweise für alle $\mathcal{EL}_\neg$-Konzeptbeschreibungen C. Überträgt man für konsistente ABoxen die Charakterisierung der Instanz von $\mathcal{EL}$ auf $\mathcal{EL}_\neg$, d.h. um $a \in_{\mathcal{A}} C$ zu entscheiden testet man, ob $\mathcal{G}(C)$ homomorph in $\mathcal{G}(\mathcal{A})$ eingebettet werden kann, so würde man insgesamt ein polynomielles Entscheidungsverfahren erhalten. Dieses Verfahren kann aber (unter der Annahme P≠NP) nicht vollständig und korrekt sein: Eine genaue Analyse der Resultate in [Sch93] zeigt nämlich, dass das Instanzproblem für $\mathcal{EL}_\neg$ coNP-hart ist. Das folgende Beispiel ist eine abstrakte Variante von Beispiel 3.1 in [Sch93] und illustriert die Unvollständigkeit des Verfahrens.

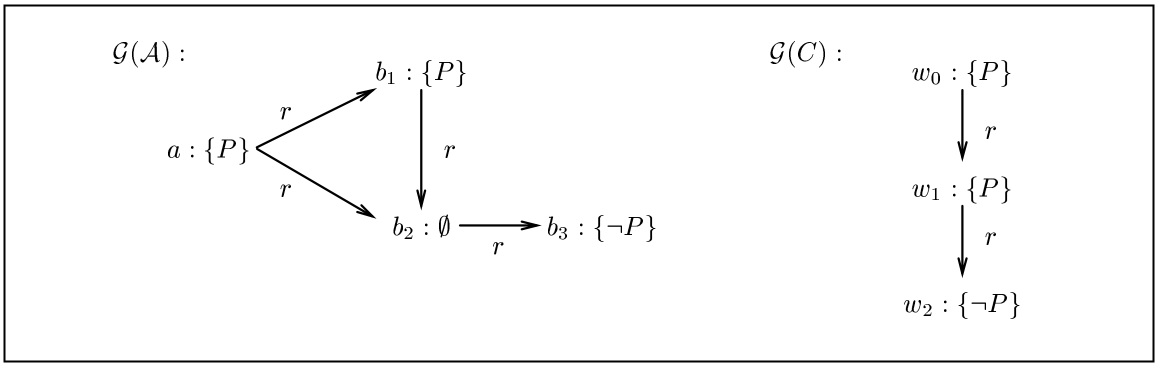


Abbildung 7.3: Beispiel für die Unvollständigkeit der Charakterisierung der Instanz in $\mathcal{EL}$ für $\mathcal{EL}_\neg$.

Beispiel 7.17 *Betrachte die $\mathcal{EL}_\neg$-Konzeptbeschreibung $C = P \sqcap \exists r.(P \sqcap \exists r.\neg P)$ und die $\mathcal{EL}_\neg$-ABox*

$$\mathcal{A} = \{a : P, b_1 : P, b_3 : \neg P, (a, b_1) : r, (a, b_2) : r, (b_1, b_2) : r, (b_2, b_3) : r\}.$$

Abbildung 7.3 zeigt $\mathcal{G}(\mathcal{A})$ und $\mathcal{G}(C)$. Offensichtlich kann $\mathcal{G}(C)$ nicht homomorph in $\mathcal{G}(\mathcal{A})$ eingebettet werden, da $\{\neg P\} \not\subseteq \ell(b_2)$. Andererseits gilt aber für jedes Modell $\mathcal{I}$ von $\mathcal{A}$ entweder $b_2^\mathcal{I} \in P^\mathcal{I}$ oder $b_2^\mathcal{I} \in (\neg P)^\mathcal{I}$, woraus stets $a^\mathcal{I} \in C^\mathcal{I}$ folgt. Also ist a Instanz von C bzgl. $\mathcal{A}$, obwohl kein Homomorphismus φ von $\mathcal{G}(C)$ nach $\mathcal{G}(\mathcal{A})$ mit $\varphi(w_0) = a$ existiert.

Das im obigen Beispiel aufgezeigte Problem rührt daher, dass für die Individuen in der ABox nicht vollständig festgelegt ist, von welchen Konzeptnamen sie Instanz sind und von welchen nicht. Dieses Problem lässt sich dadurch lösen, dass man bei der Charakterisierung der Instanz vom $\mathcal{EL}_\neg$-Graphen $\mathcal{G}(\mathcal{A})$ übergeht zur Menge der sogenannten primitiven Vervollständigungen von $\mathcal{G}(\mathcal{A})$. Informell lässt sich eine primitive Vervollständigung wie folgt beschreiben: Knoten- und Kantenmenge stimmen mit denen des Ausgangsgraphen überein. Vervollständigt werden lediglich die Knotenlabel und zwar in dem Sinne, dass in der primitiven Vervollständigung jedes Knotenlabel jeden Konzeptnamen, der im Ausgangsgraphen positiv oder negiert auftritt, entweder positiv oder negiert enthält.

Definition 7.18 (Primitive Vervollständigung)
Sei $\mathcal{G} = (V, E, \ell)$ ein $\mathcal{EL}_\neg$-Graph und $N_C^* := \{P \in N_C \mid$ existiert $v \in V$ mit $P \in \ell(v)$ oder $\neg P \in \ell(v)\}$. Ein $\mathcal{EL}_\neg$-Graph $\mathcal{G}^* = (V, E, \ell^*)$ heißt primitive Vervollständigung *von $\mathcal{G}$ genau dann, wenn für alle $v \in V$ gilt:*

1. *$\ell(v) \subseteq \ell^*(v)$ und*
2. *für alle Konzeptnamen $P \in N_C^*$ ist entweder $P \in \ell^*(v)$ oder $\neg P \in \ell^*(v)$.*

Nun können wir Instanz in $\mathcal{EL}_\neg$ vollständig und korrekt charakterisieren.

Theorem 7.19 *Seien $\mathcal{A}$ eine $\mathcal{EL}_\neg$-ABox mit $\mathcal{G}(\mathcal{A}) = (V, E, \ell)$ und C eine $\mathcal{EL}_\neg$-Konzeptbeschreibung mit $\mathcal{G}(C) = (V_C, E_C, w_0, \ell_C)$. Für $a \in \mathit{Ind}(\mathcal{A})$ gilt $a \in_\mathcal{A} C$ genau dann, wenn $\mathcal{A}$ inkonsistent ist oder für jede primitive Vervollständigung $\mathcal{G}(\mathcal{A})^*$ von $\mathcal{G}(\mathcal{A})$ ein Homomorphismus φ von $\mathcal{G}(C)$ nach $\mathcal{G}(\mathcal{A})^*$ existiert mit $\varphi(w_0) = a$.*

Beweis der Rückrichtung: Da die Voraussetzung nur Homomorphismen von $\mathcal{G}(C)$ nach primitiven Vervollständigungen liefert und diese im allgemeinen nicht mit dem $\mathcal{EL}_\neg$-Graphen $\mathcal{G}(\mathcal{A})$ übereinstimmen, können wir für den Beweis der Rückrichtung nicht wie für $\mathcal{EL}$ auf den Korrektheitsbeweis für $\mathcal{ALE}$ (cf Lemma 7.28) zurückgreifen. Wir führen für $\mathcal{EL}_\neg$ einen alternativen Beweis.

Sei dazu $\mathcal{A}$ eine konsistente $\mathcal{EL}_\neg$-ABox (für inkonsistente ABoxen ist nichts zu zeigen) und $\mathcal{I} = (\Delta_\mathcal{I}, \cdot^\mathcal{I})$ ein beliebiges Modell von $\mathcal{A}$. Wir zeigen $a^\mathcal{I} \in C^\mathcal{I}$ wie folgt: zunächst definieren wir eine Abbildung $\psi : V \longrightarrow \Delta_\mathcal{I}$ mit

1. $\psi(b) = b^\mathcal{I}$ für alle $b \in \mathit{Ind}(\mathcal{A})$,
2. $(\psi(v), \psi(w)) \in r^\mathcal{I}$ für alle $vrw \in E$ und
3. $\psi(v) \in Q^\mathcal{I}$ für alle $Q \in \ell(v)$ und alle $v \in V$.

Mit Hilfe dieser Abbildung wird dann eine primitive Vervollständigung $\mathcal{G}(\mathcal{A})^*_\psi = (V, E, \ell^*_\psi)$ definiert, sodass ψ weiterhin die Bedingung

4. $\psi(v) \in Q^\mathcal{I}$ für alle $Q \in \ell^*_\psi(v)$ und alle $v \in V$

erfüllt. Nach Voraussetzung existiert ein Homomorphismus φ von $\mathcal{G}(C)$ nach $\mathcal{G}(\mathcal{A})^*_\psi$. Durch Induktion nach der Tiefe von $\mathcal{G}(C)(w)$ werden wir für alle $w \in V_C$ zeigen

$$\psi(\varphi(w)) \in C_{\mathcal{G}(C)(w)}{}^\mathcal{I}. \tag{7.1}$$

Mit $\varphi(w_0) = a$ und $\psi(a) = a^\mathcal{I}$ sowie $C \equiv C_{\mathcal{G}(C)(w_0)}$ folgt daraus schließlich $a^\mathcal{I} \in C^\mathcal{I}$.

Zur Definition der Abbildung $\psi : V \longrightarrow \Delta_\mathcal{I}$: Sei $a \in \mathit{Ind}(\mathcal{A})$, $C_a = \bigsqcap_{a:D\in\mathcal{A}} D$ und $\mathcal{G}(C_a) = (V_a, E_a, a, \ell_a)$. Definiere $\psi(v)$ für $v \in V_a$ durch Induktion nach der Länge λ des eindeutigen Pfades von a nach v in $\mathcal{G}(C_a)$ so, dass für $\psi(v)$ jeweils auch $\psi(v) \in C_{\mathcal{G}(C_a)(v)}{}^\mathcal{I}$ gilt.

$\lambda = 0$: Dann ist $v = a$. Definiere $\psi(a) := a^\mathcal{I}$. Wegen $\mathcal{I} \models \mathcal{A}$ folgt $a^\mathcal{I} \in C_a^\mathcal{I}$. Insbesondere ist $a^\mathcal{I} \in Q^\mathcal{I}$ für alle $Q \in \ell(a)$. Damit folgt Bedingung (1) sowie Bedingung (3) für $a \in \mathit{Ind}(\mathcal{A})$.

$\lambda > 0$: Dann existiert eine eindeutige Kante $v'rv \in E_a$. Nach Induktionsvoraussetzung ist $\psi(v')$ bereits definiert mit $\psi(v') \in C_{\mathcal{G}(C_a)(v')}{}^{\mathcal{I}}$. Wegen $v'rv \in E_a$ existiert eine Existenzrestriktion der Form $\exists r.C_{\mathcal{G}(C_a)(v)}$ auf Toplevel von $C_{\mathcal{G}(C_a)(v')}$. Aus $\psi(v') \in C_{\mathcal{G}(C_a)(v')}{}^{\mathcal{I}}$ folgt, dass ein $\alpha \in \Delta_{\mathcal{I}}$ existiert mit $(\psi(v'), \alpha) \in r^{\mathcal{I}}$ und $\alpha \in C_{\mathcal{G}(C_a)(v)}{}^{\mathcal{I}}$. Definiere $\psi(v) := \alpha$. Mit $\alpha \in C_{\mathcal{G}(C_a)(v)}{}^{\mathcal{I}}$ und $\ell(v) = \ell_a(v)$ erhalten wir insbesondere $\alpha \in Q^{\mathcal{I}}$ für alle $Q \in \ell(v)$, also gilt Bedingung (3) für diesen Knoten.

Nach Konstruktion erfüllt ψ obige Bedingungen (1)–(3).

Zur Definition der durch ψ induzierten primitiven Vervollständigung sei N_C^* die Menge aller Konzeptnamen, die positiv oder negiert in $\mathcal{G}(\mathcal{A})$ vorkommen. Definiere $\mathcal{G}(\mathcal{A})^*_\psi := (V, E, \ell^*_\psi)$ mit

$$\ell^*_\psi(v) := \{P \in N_C^* \mid \psi(v) \in P^{\mathcal{I}}\} \cup \{\neg P \mid P \in N_C^* \text{ und } \psi(v) \notin P^{\mathcal{I}}\}.$$

Mit Bedingung (3) folgt $\ell(v) \subseteq \ell^*_\psi(v)$ für alle $v \in V$, also ist $\mathcal{G}(\mathcal{A})^*_\psi$ tatsächlich eine primitive Vervollständigung von $\mathcal{G}(\mathcal{A})$. Nach Definition gilt offensichtlich Bedingung (4).

Nach Voraussetzung existiert also ein Homomorphismus φ von $\mathcal{G}(C)$ nach $\mathcal{G}(\mathcal{A})^*_\psi$ mit $\varphi(w_0) = a$. Wir zeigen Eigenschaft (7.1) durch Induktion nach der Tiefe von $\mathcal{G}(C)(w)$:

$\mathit{depth}(\mathcal{G}(C)(w)) = 0$: Dann ist $C_{\mathcal{G}(C)(w)} = \bigsqcap_{Q \in \ell_C(w)} Q$.

Wegen $\ell_C(w) \subseteq \ell^*_\psi(\varphi(w))$ folgt mit Bedingung (4) $\psi(\varphi(w)) \in Q^{\mathcal{I}}$ für alle $Q \in \ell_C(w)$. Also ist $\psi(\varphi(w)) \in C_{\mathcal{G}(C)(w)}{}^{\mathcal{I}}$.

$\mathit{depth}(\mathcal{G}(C)(w)) > 0$: Dann ist $C_{\mathcal{G}(C)(w)} = \bigsqcap_{Q \in \ell_C(w)} Q \sqcap \bigsqcap_{wrw' \in E_C} \exists r.C_{\mathcal{G}(C)(w')}$.

Wie oben folgt $\psi(\varphi(w)) \in Q^{\mathcal{I}}$ für alle $Q \in \ell_C(w)$. Sei also $wrw' \in E_C$ beliebig. Nach Definition von ψ und da φ Homomorphismus ist $(\psi(\varphi(w)), \psi(\varphi(w'))) \in r^{\mathcal{I}}$. Per Induktion folgt $\psi(\varphi(w')) \in C_{\mathcal{G}(C)(w')}{}^{\mathcal{I}}$ und damit $\psi(\varphi(w)) \in (\exists r.C_{\mathcal{G}(C)(w')})^{\mathcal{I}}$.

Insgesamt folgt $\psi(\varphi(w)) \in C_{\mathcal{G}(C)(w)}{}^{\mathcal{I}}$.

Dies schließt den Beweis der Rückrichtung ab.

Beweis der Hinrichtung: Sei wiederum $\mathcal{A}$ eine konsistente $\mathcal{EL}_\neg$-ABox, $N_C^* := \{P \in N_C \mid \exists v \in V : P \in \ell(v) \vee \neg P \in \ell(v)\}$ und $a \in_{\mathcal{A}} C$. Sei $\mathcal{G}(C) = (V_C, E_C, w_0, \ell_C)$, $\mathcal{G}(\mathcal{A})^* = (V, E, \ell^*)$ eine beliebige primitive Vervollständigung von $\mathcal{G}(\mathcal{A})$ und $\mathcal{I}(\mathcal{G}(\mathcal{A})^*)$ die durch $\mathcal{G}(\mathcal{A})^*$ induzierte kanonische Interpretation.

Mit Lemma 7.16 folgt, dass die kanonische Interpretation $\mathcal{I}(\mathcal{G}(\mathcal{A}))$ ein Modell von $\mathcal{A}$ ist. Wegen

- $r^{\mathcal{I}(\mathcal{G}(\mathcal{A}))} = r^{\mathcal{I}(\mathcal{G}(\mathcal{A})^*)}$ für alle $r \in N_R$,

- $P^{\mathcal{I}(\mathcal{G}(\mathcal{A}))} \subseteq P^{\mathcal{I}(\mathcal{G}(\mathcal{A})^*)}$ für alle $P \in N_C$ und
- $\neg P \in \ell(v) \implies P \notin \ell^*(v) \implies v \notin P^{\mathcal{I}(\mathcal{G}(\mathcal{A})^*)}$

ist auch $\mathcal{I}(\mathcal{G}(\mathcal{A})^*)$ ein Modell von $\mathcal{A}$. Zeige zunächst: für alle Konzeptnamen P, die positiv oder negiert in C vorkommen, gilt $P \in N_C^*$. Angenommen, es existiert ein $P \notin N_C^*$ und ein $r_1 \dots r_n$-Nachfolger $w \in V_C$ von w_0 in $\mathcal{G}(C)$ mit $P \in \ell_C(w)$. Dann gilt für jedes Modell $\mathcal{I}$ von C und jedes $\alpha \in C^{\mathcal{I}}$, dass ein $(r_1 \dots r_n)^{\mathcal{I}}$-Nachfolger β von α in $\mathcal{I}$ existiert mit $\beta \in P^{\mathcal{I}}$. Wegen $P \notin N_C^*$ ist aber $P^{\mathcal{I}(\mathcal{G}(\mathcal{A})^*)} = \emptyset$ und damit $C^{\mathcal{I}(\mathcal{G}(\mathcal{A})^*)} = \emptyset$ im Widerspruch zu $a \in_{\mathcal{A}} C$. Ist $\neg P \in \ell_C(w)$, so gilt für jedes Modell $\mathcal{I}$ von C und jedes $\alpha \in C^{\mathcal{I}}$, dass ein $(r_1 \dots r_n)^{\mathcal{I}}$-Nachfolger β von α in $\mathcal{I}$ existiert mit $\beta \notin P^{\mathcal{I}}$. Definiere $\mathcal{J} := (V, \cdot^{\mathcal{J}})$ mit $Q^{\mathcal{J}} := Q^{\mathcal{I}(\mathcal{G}(\mathcal{A})^*)}$ für alle $Q \in N_C^*$ und $P^{\mathcal{J}} := V$. Dann ist $\mathcal{J}$ Modell von $\mathcal{A}$, da die Interpretation der Konzept- und Rollennamen, die in $\mathcal{A}$ vorkommen, gegenüber $\mathcal{I}(\mathcal{G}(\mathcal{A})^*)$ nicht geändert wurde. Aber offensichtlich ist $(\neg P)^{\mathcal{J}} = \emptyset$ und damit $C^{\mathcal{J}} \neq \emptyset$ im Widerspruch zu $a \in_{\mathcal{A}} C$.

Damit ist gezeigt, dass für eine konsistente $\mathcal{EL}_\neg$-ABox $\mathcal{A}$ aus $a \in_{\mathcal{A}} C$ folgt, dass C nur Konzeptnamen enthält, die auch in $\mathcal{A}$ vorkommen. Aus der Tatsache, dass $\mathcal{I}(\mathcal{G}(\mathcal{A}^*))$ Modell von $\mathcal{A}$ ist und der Voraussetzung $a \in_{\mathcal{A}} C$ folgt nun die Existenz eines Homomorphismus φ von $\mathcal{G}(C)$ nach $\mathcal{G}(\mathcal{A})^*$ unmittelbar aus der folgenden

Behauptung: Ist $v \in C^{\mathcal{I}(\mathcal{G}(\mathcal{A})^*)}$ und enthält C nur Konzeptnamen, die auch in $\mathcal{A}$ vorkommen, so existiert ein Homomorphismus φ von $\mathcal{G}(C)$ nach $\mathcal{G}(\mathcal{A})^*$ mit $\varphi(w_0) = v$.

Beweis der Behauptung durch Induktion nach $depth(C)$:

$depth(C) = 0$: Dann ist $C = Q_1 \sqcap \dots \sqcap Q_n$ mit $Q_i \in N_C \cup \{\neg P \mid P \in N_C\}$.

Definiere φ durch $\varphi(w_0) := v$. Wir wissen, dass $v \in C^{\mathcal{I}(\mathcal{G}(\mathcal{A})^*)}$. Für $Q_i \in N_C$ folgt aus der Definition von $\mathcal{I}(\mathcal{G}(\mathcal{A})^*)$, dass $Q_i \in \ell(v)$. Für $Q_i = \neg P$ folgt aus $v \in Q_i^{\mathcal{I}(\mathcal{G}(\mathcal{A})^*)}$, dass $P \notin \ell(v)$. Aus der Bedingung an C folgt $P \in N_C^*$. Da $\mathcal{G}(\mathcal{A})^*$ eine primitive Vervollständigung ist, folgt daraus $\neg P \in \ell(v)$. Insgesamt ist also $\ell_C(w_0) \subseteq \ell^*(v)$ und φ ein Homomorphismus von $\mathcal{G}(C)$ nach $\mathcal{G}(\mathcal{A})^*$.

$depth(C) > 0$: Analog zu $\mathcal{EL}$.

Dies schließt den Beweis der Hinrichtung ab. □

Komplexität des Instanzproblems in $\mathcal{EL}_\neg$

Die in Theorem 7.19 gefundene Charakterisierung der Instanz $a \in_{\mathcal{A}} C$ in $\mathcal{EL}_\neg$ führt zu folgendem Entscheidungsalgorithmus für dieses Problem: Teste zuerst, ob $\mathcal{A}$ inkonsistent ist (in polynomieller Zeit; Lemma 7.15). Wenn ja, so gilt $a \in_{\mathcal{A}} C$ trivial. Andernfalls entscheide $a \notin_{\mathcal{A}} C$ in nichtdeterministisch polynomieller Zeit durch

1. raten einer primitiven Vervollständigung $\mathcal{G}(\mathcal{A})^*$ und
2. zeigen, dass kein Homomorphismus von $\mathcal{G}(C)$ nach $\mathcal{G}(\mathcal{A})^*$ mit $\varphi(w_0) = a$ existiert.

Damit ist das Instanzproblem in $\mathcal{EL}_\neg$ in coNP. Zusammen mit dem Härteresultat aus [Sch93] folgt

Satz 7.20 *Das Instanzproblem in $\mathcal{EL}_\neg$ ist coNP-vollständig.*

7.3.2 Die k-Approximation des MSC in $\mathcal{EL}_\neg$

Auf Grund des in Beispiel 7.17 illustrierten Problems liefert der für $\mathcal{EL}$ beschriebene Algorithmus zur Berechnung der k-Approximation des MSC (im zyklischen Fall) bzw. des MSC (im azyklischen Fall) für $\mathcal{EL}_\neg$-ABoxen im allgemeinen nicht das Gewünschte. So würden wir mit der in Abschnitt 7.2.2 beschriebenen Vorgehensweise für die $\mathcal{EL}_\neg$-ABox $\mathcal{A}$ aus Beispiel 7.17 die $\mathcal{EL}_\neg$-Konzeptbeschreibung

$$\begin{array}{rcl} C_{\mathcal{T}(a,\mathcal{A})} & = & P \sqcap \\ & & \exists r.\exists r.(\neg P) \sqcap \\ & & \exists r.(P \sqcap \exists r.\exists r.(\neg P)) \end{array}$$

erhalten. Tatsächlich ist $\mathsf{msc}_\mathcal{A}(a)$ aber—wie wir unten sehen werden—gegeben durch

$$\mathsf{msc}_\mathcal{A}(a) = P \sqcap \exists r.(P \sqcap \exists r.\neg P) \sqcap \exists r.(P \sqcap \exists r.\exists r.\neg P)$$

und offensichtlich gilt $\mathsf{msc}_\mathcal{A}(a) \sqsubset C_{\mathcal{T}(a,\mathcal{A})}$.

Um das MSC (bzw. im zyklischen Fall eine k-Approximation) zu erhalten, sind alle primitiven Vervollständigungen zu berücksichtigen, genauer gesagt, das „Gemeinsame“ aus allen primitiven Vervollständigungen.

Betrachtet man eine Menge von Konzeptbeschreibungen, so repräsentiert der LCS das „Gemeinsame“ dieser Konzeptbeschreibungen. Wie gezeigt wurde, entspricht der LCS genau dem Produkt der zugehörigen Beschreibungsbäume. Nun hat sich herausgestellt, dass auch in der vorliegenden Situation das Produkt der primitiven Vervollständigungen das „Gemeinsame“ geeignet beschreibt.

Definition 7.21 *Sei $\{\mathcal{G}_i = (V_i, E_i, \ell_i) \mid 1 \leq i \leq n\}$ eine endliche Menge von $\mathcal{EL}_\neg$-Graphen. Das* Produkt $\prod_{1 \leq i \leq n} \mathcal{G}_i$ von $\mathcal{G}_1, \ldots, \mathcal{G}_n$ *ist definiert als der $\mathcal{EL}_\neg$-Graph (V, E, ℓ) mit*

- $V = V_1 \times \ldots \times V_n$,

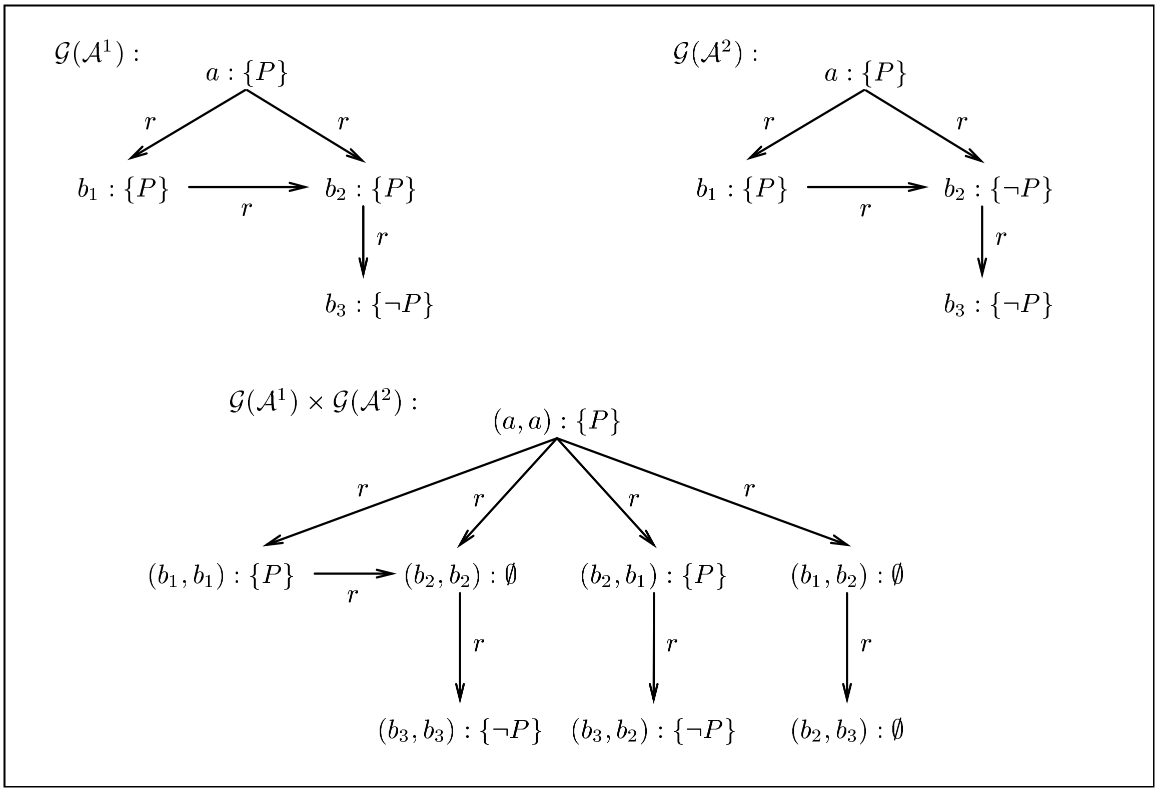


Abbildung 7.4: Ein Beispiel für das Produkt zweier $\mathcal{EL}_\neg$-Graphen, wobei im Produkt nur die vom Knoten (a, a) aus erreichbaren Knoten eingezeichnet sind.

- $E = \{(v_1, \dots, v_n)r(w_1, \dots, w_n) \mid v_i r w_i \in E_i$ *für alle* $1 \leq i \leq n\}$ *und*
- $\ell(v_1, \dots, v_n) = \bigcap_{1 \leq i \leq n} \ell_i(v_i)$.

Für die $\mathcal{EL}_\neg$-ABox $\mathcal{A}$ aus Beispiel 7.17 erhält man die beiden oben in Abbildung 7.4 dargestellten primitiven Vervollständigungen von $\mathcal{G}(\mathcal{A})$. Darunter ist das Produkt von $\mathcal{G}(\mathcal{A}^1)$ und $\mathcal{G}(\mathcal{A}^2)$ abgebildet, wobei nur die vom Knoten (a, a) aus erreichbaren Knoten berücksichtigt wurden. Diese Einschränkung der Knoten- und Kantenmenge ist mit Blick auf die Definition des Baumes mit Wurzel (a, a) offensichtlich o.B.d.A. möglich. Im konkreten Beispiel erhält man als Konzeptbeschreibung zum $\mathcal{EL}_\neg$-Beschreibungsbaum $\mathcal{T}((a, a), \mathcal{G}(\mathcal{A}^1) \times \mathcal{G}(\mathcal{A}^2))$

$$C = P \sqcap \exists r.\exists r.\top \sqcap \exists r.(P \sqcap \exists r.\neg P) \sqcap \exists r.(P \sqcap \exists r.\exists r.\neg P) \sqcap \exists r.\exists r.\neg P.$$

Man sieht leicht, dass $C \equiv \mathsf{msc}_\mathcal{A}(a)$. Im allgemeinen gilt

Theorem 7.22 *Seien* $\mathcal{A}$ *eine* $\mathcal{EL}_\neg$*-ABox,* $a \in \mathit{Ind}(\mathcal{A})$ *und* $k \in \mathbb{N}$*. Ist* $\mathcal{A}$ *inkonsistent, so ist die* k*-Approximation von* a *bzgl.* $\mathcal{A}$ *gleich dem Most Specific Concept von* a *bzgl.* $\mathcal{A}$ *und dieses ist gleich* $\bot$*. Andernfalls sei* $\{\mathcal{G}(\mathcal{A})^1, \dots, \mathcal{G}(\mathcal{A})^n\}$ *die*

Menge aller primitiven Vervollständigungen von $\mathcal{G}(\mathcal{A})$, $\mathcal{G} = (V, E, \ell)$ das Produkt $\prod_{1 \leq i \leq n} \mathcal{G}(\mathcal{A})^i$ der Vervollständigungen und schließlich $\boldsymbol{a} \in V$ das n-Tupel $(a, \ldots, a)$. Dann ist $C_{\mathcal{T}_k(\boldsymbol{a},\mathcal{G})}$ die k-Approximation von a bzgl. $\mathcal{A}$. Ist $\mathcal{A}$ azyklisch, so ist $C_{\mathcal{T}(\boldsymbol{a},\mathcal{G})}$ das Most Specific Concept von a bzgl. $\mathcal{A}$.

Beweis: Ist $\mathcal{A}$ inkonsistent, so ist $\bot$ offensichtlich das bzgl. $\sqsubseteq$ speziellste Konzept (unabhängig von der Schranke k), das die Bedingungen an die k-Approximation von a bzgl. $\mathcal{A}$ bzw. das MSC von a bzgl. $\mathcal{A}$ erfüllt.
Sei also im folgenden $\mathcal{A}$ eine konsistente $\mathcal{EL}_\neg$-ABox. Für eine zyklische ABox $\mathcal{A}$ sind zu zeigen

1. $a \in_{\mathcal{A}} C_{\mathcal{T}_k(\mathbf{a},\mathcal{G})}$ und
2. für alle C mit $depth(C) \leq k$ und $a \in_{\mathcal{A}} C$ gilt $C_{\mathcal{T}_k(\mathbf{a},\mathcal{G})} \sqsubseteq C$.

Ad (1): Betrachte (wie im Beweis von Theorem 7.9) die Abbildung φ, die jeden Knoten p in $\mathcal{T}_k(\mathbf{a}, \mathcal{G})$ auf den jeweils letzten Knoten des durch p repräsentierten Pfades in $\mathcal{G}$ abbildet. Definiere φ_i als die Hintereinanderausführung $\pi_i \circ \varphi$ von φ und der Projektion π_i der i. Komponente des Knotens $\varphi(p)$. Man sieht leicht, dass φ_i einen Homomorphismus von $\mathcal{T}_k(\mathbf{a}, \mathcal{G})$ nach $\mathcal{G}(\mathcal{A})^i$ mit $\varphi_i(\mathbf{a}) = a$ liefert. Da $\mathcal{T}_k(\mathbf{a}, \mathcal{G})$ bis auf Knotenumbenennung gleich $\mathcal{G}(C_{\mathcal{T}_k(\mathbf{a},\mathcal{G})})$ ist (Eigenschaft (5.13)), folgt daraus mit Theorem 7.19 die Behauptung.
Ad (2): Sei C eine $\mathcal{EL}_\neg$-Konzeptbeschreibung mit $a \in_{\mathcal{A}} C$, $depth(C) \leq k$ und $\mathcal{G}(C) = (V_C, E_C, w_0, \ell_C)$. Nach Annahme ist $\mathcal{A}$ konsistent, also folgt mit Theorem 7.19, dass Homomorphismen φ_i von $\mathcal{G}(C)$ nach $\mathcal{G}(\mathcal{A})^i$ mit $\varphi_i(w_0) = a$ existieren für alle $1 \leq i \leq n$. Analog zum Beweis von Theorem 5.9 bzw. Theorem 5.28 kann man zeigen, dass das n-stellige Produkt der Homomorphismen φ_i einen Homomorphismus φ von $\mathcal{G}(C)$ nach $\mathcal{G}$ liefert. Unter Verwendung von φ kann dann wie im Beweis von Theorem 7.9 ein Homomorphismus ψ von $\mathcal{G}(C)$ nach $\mathcal{T}_k(\mathbf{a}, \mathcal{G})$ definiert werden. Daraus folgt mit Eigenschaft (5.13) und Theorem 5.21 $C_{\mathcal{T}_k(\mathcal{G}(\mathcal{A}))} \sqsubseteq C$.
Ist $\mathcal{A}$ azyklisch, so auch das Produkt $\mathcal{G}$ der primitiven Vervollständigungen von $\mathcal{G}(\mathcal{A})$. Damit ist $\mathcal{T}(\mathbf{a}, \mathcal{G})$ endlich und $C_{\mathcal{T}(\mathbf{a},\mathcal{G})}$ eine $\mathcal{EL}_\neg$-Konzeptbeschreibung. Wie für $\mathcal{T}_k(a, \mathcal{G}(\mathcal{A}))$ folgt $a \in_{\mathcal{A}} C_{\mathcal{T}(a,\mathcal{G}(\mathcal{A}))}$ und $C_{\mathcal{T}(a,\mathcal{G}(\mathcal{A}))} \sqsubseteq C$ für alle $\mathcal{EL}_\neg$-Konzeptbeschreibungen C mit $a \in_{\mathcal{A}} C$. Also ist $C_{\mathcal{T}(a,\mathcal{G}(\mathcal{A}))}$ das MSC von a bzgl. $\mathcal{A}$. □

Korollar 7.23 *Für eine $\mathcal{EL}_\neg$-ABox $\mathcal{A}$, ein Individuum $a \in Ind(\mathcal{A})$ und $k \in \mathbb{N}$ existiert die k-Approximation $\mathsf{msc}_{\mathcal{A},k}(a)$ von a bzgl. $\mathcal{A}$ stets und ist effektiv berechenbar. Ist $\mathcal{A}$ azyklisch, so existiert das Most Specific Concept $\mathsf{msc}_{\mathcal{A}}(a)$ stets und ist effektiv berechenbar.*

Man beachte, dass in der Definition einer primitiven Vervollständigung die Menge der zu berücksichtigenden Konzeptnamen auf die endliche Menge von Konzeptnamen, die in $\mathcal{A}$ tatsächlich vorkommen, eingeschränkt wird. Damit ist die Menge der primitiven Vervollständigungen zu einer $\mathcal{EL}_\neg$-ABox stets endlich und daher $\mathsf{msc}_{\forall,k}(a)$ bzw. $\mathsf{msc}_{\mathcal{A}}(a)$ effektiv berechenbar.

Komplexität (der k-Approximation) des MSC in $\mathcal{EL}_\neg$

Die Beispiele zur Komplexität (der k-Approximation) des MSC in $\mathcal{EL}$ basieren auf ABoxen, die keinerlei Konzeptnamen enthalten. Betrachtet man also die ABox $\mathcal{A}$ aus Beispiel 7.11 bzw. $\mathcal{A}_i$, $i \geq 1$, aus Beispiel 7.12 als $\mathcal{EL}_\neg$-ABoxen über der leeren Menge als Menge von Konzeptnamen, so bilden $\mathcal{G}(\mathcal{A})$ bzw. $\mathcal{G}(\mathcal{A}_i)$ jeweils die einzige primitive Vervollständigung von sich selbst und man erhält $C_{\mathcal{T}_k(a,\mathcal{G}(\mathcal{A}))}$ als k-Approximation von a bzgl. $\mathcal{A}$ bzw. $C_{\mathcal{T}(a_1,\mathcal{G}(\mathcal{A}_i))}$ als MSC von a_1 bzgl. $\mathcal{A}_i$, $i \geq 1$. Auch hier existieren keine äquivalenten, kleineren $\mathcal{EL}_\neg$-Konzeptbeschreibungen und wir erhalten

Satz 7.24 *Für eine $\mathcal{EL}_\neg$-ABox $\mathcal{A}$, $a \in Ind(\mathcal{A})$ und $k \in \mathbb{N}$ kann*

1. *die k-Approximation $\mathsf{msc}_{\mathcal{A},k}(a)$ von a bzgl. $\mathcal{A}$ exponentiell groß in k und der Größe von $\mathcal{A}$ sein; und*

2. *das Most Specific Concept $\mathsf{msc}_{\mathcal{A}}(a)$ von a bzgl. $\mathcal{A}$ exponentiell groß in der Größe von $\mathcal{A}$ sein.*

Versucht man ähnlich wie für $\mathcal{EL}$ die Größe des MSC bzw. der k-Approximation über die Größe von $\mathcal{T}(\mathbf{a}, \mathcal{G}(\mathcal{A}))$ bzw. $\mathcal{T}_k(\mathbf{a}, \mathcal{G}(\mathcal{A}))$ nach oben abzuschätzen, so erhält man als grobe Abschätzung eine doppelt exponentielle Schranke: Jede primitive Vervollständigung ist polynomiell groß bzgl. der Größe von $\mathcal{A}$. Die Anzahl primitiver Vervollständigungen kann aber nur exponentiell in $|\mathcal{A}|$ beschränkt werden. Die Größe der Bäume zu den primitiven Vervollständigungen ist jeweils exponentiell in $|\mathcal{A}|$ bzw. k beschränkt. Folglich kann die Größe des Produktes (und damit des MSC bzw. der k-Approximation) nur doppelt exponentiell beschränkt werden. Da aber MSC bzw. k-Approximation in jede primitive Vervollständigung einzubetten sind und diese jeweils polynomiell groß in $|\mathcal{A}|$ sind, ist stark zu vermuten, dass wie für $\mathcal{EL}$ die Größe von MSC bzw. k-Approximation bei genauerer Analyse jeweils einfach exponentiell beschränkt werden kann.

7.4 Most Specific Concepts in $\mathcal{ALE}$

Wie bereits in Abschnitt 7.1 erwähnt konnte im Rahmen der vorliegenden Arbeit keine Charakterisierung der Instanz in $\mathcal{ALE}$ entwickelt werden, die analog zu $\mathcal{EL}$ bzw. $\mathcal{EL}_\neg$ eine effiziente Berechnung der k-Approximation bzw. im azyklischen Fall sogar des MSC erlaubt. Dieser Abschnitt beschränkt sich daher darauf, zum einen Existenz und Berechenbarkeit der k-Approximation von a bzgl. $\mathcal{A}$ und k für den Fall nachzuweisen, dass die Mengen N_C und N_R endlich sind (cf Theorem 7.25). Zum anderen werden aber

- die Korrektheit der für $\mathcal{EL}$ gefundenen Charakterisierung der Instanz für $\mathcal{ALE}$ bewiesen (cf Lemma 7.28) und

- durch Werterestriktionen verursachte Probleme bei der Charakterisierung der Instanz in $\mathcal{ALE}$ anhand eines Beispiels aufgezeigt (cf Beispiel 7.29).

Die Korrektheitsaussage bildet die Basis für eine „Approximation der Approximation“ (cf Lemma 7.30), von der zu hoffen ist, dass sie in der Anwendung bereits nützliche Resultate liefern kann. Für die Beispiele zur Unvollständigkeit der Charakterisierung aufgrund von primitiver Negation (s. Beispiel 7.17) bzw. Werterestriktionen (cf Beispiel 7.29) ist zu erwarten, dass ihre Analyse zu Erkenntnissen und ersten Ansatzpunkten für zukünftige Arbeiten führt, deren Ziel die Entwicklung einer vollständigen Charakterisierung der Instanz und eines effizienten Algorithmus für die Berechnung der Approximation des MSC in $\mathcal{ALE}$ sein soll.

Existenz und Berechenbarkeit der k-Approximation in $\mathcal{ALE}$

Wir beginnen mit dem Nachweis der Existenz und Berechenbarkeit der k-Approximation des MSC in $\mathcal{ALE}$ bei endlicher Signatur.

Theorem 7.25 *Seien N_C und N_R endliche Mengen von Konzept- bzw. Rollennamen und $\mathcal{A}$ eine $\mathcal{ALE}$-ABox, die nur Namen aus $N_C \cup N_R$ enthält. Dann existiert für $k \in \mathbb{N}$ und $a \in Ind(\mathcal{A})$ die k-Approximation von a bzgl. $\mathcal{A}$ stets und kann effektiv berechnet werden.*

Beweis: Durch Induktion nach k sieht man leicht, dass

- es für endliche Mengen N_C und N_R bis auf Äquivalenz nur endlich viele verschiedene $\mathcal{ALE}$-Konzeptbeschreibungen mit Tiefe kleiner gleich k gibt und
- die Menge $\mathcal{C}_k := \{[C]_{\equiv,k} \mid depth(C) \leq k\}$ endlich ist und effektiv aufgezählt werden kann, wobei $[C]_{\equiv,k} := \{D \mid depth(D) \leq k \text{ und } C \equiv D\}$.

Ist nun $\mathcal{A}$ inkonsistent, so ist offensichtlich $\mathsf{msc}_{\mathcal{A}}(a) = \bot$ und es ist nichts zu zeigen. Sei also $\mathcal{A}$ konsistent. Definiere $\mathcal{C}_k(a, \mathcal{A}) := \{[C]_{\equiv,k} \in \mathcal{C}_k \mid a \in_{\mathcal{A}} C\}$. Da $\mathcal{C}_k$ endlich ist und effektiv aufgezählt werden kann und da das Instanz- und das Konsistenzproblem in $\mathcal{ALE}$ effektiv entscheidbar sind [Sch93], ist $\mathcal{C}_k(a, \mathcal{A})$ endlich und kann effektiv berechnet werden. Dann ist

$$\mathsf{msc}_{\mathcal{A},k}(a) \equiv \bigsqcap_{[C]_{\equiv,k} \in \mathcal{C}_k(a,\mathcal{A})} C,$$

wobei für jede Klasse aus $\mathcal{C}_k(a, \mathcal{A})$ genau ein Repräsentant berücksichtigt wird. Also ist die Konjunktion endlich und die Approximation von a bzgl. $\mathcal{A}$ und k kann effektiv berechnet werden. □

Bemerkung 7.26 *Die Voraussetzung in Theorem 7.25, dass N_C und N_R endlich sein müssen, stellt für praktische Anwendungen keine wesentliche Einschränkung dar, da dort meistens endliche Signaturen betrachtet werden.*

Darüberhinaus ist stark zu vermuten, dass auf diese einschränkende Bedingung verzichtet werden kann. Voraussetzung hierfür ist die Gültigkeit der folgenden Behauptung: Bei der Berechnung von $\mathsf{msc}_{\mathcal{A},k}(a)$ kann man sich stets auf die Mengen der in $\mathcal{A}$ tatsächlich auftretenden Konzept- und Rollennamen einschränken, d.h. wenn $\mathsf{msc}_{\mathcal{A},k}(a)$ existiert, dann existiert auch eine $\mathcal{ALE}$-Konzeptbeschreibung C mit $C \equiv \mathsf{msc}_{\mathcal{A},k}(a)$, die nur Konzept- und Rollennamen verwendet, die auch in $\mathcal{A}$ auftreten.

Für $\mathcal{EL}$ und $\mathcal{EL}_\neg$ folgt diese Behauptung unmittelbar aus der jeweiligen Charakterisierung der Instanz. Für $\mathcal{ALE}$ bleibt der Beweis jedoch offen, da keine geeignete Charakterisierung der Instanz entwickelt werden konnte.

Es sei an dieser Stelle auch noch angemerkt, dass anders als bei $\mathcal{EL}$ und $\mathcal{EL}_\neg$ hier keine stärkere Aussage für den azyklischen Fall gemacht werden kann. Da nicht klar ist, ob und wie die Tiefe des MSC eines Individuums $a \in \mathit{Ind}(\mathcal{A})$ für eine azyklische $\mathcal{ALE}$-ABox zu beschränken ist, kann der Algorithmus aus dem Beweis von Theorem 7.25 nicht für die Berechnung des MSC herangezogen werden.

Eine hinreichende Bedingung für Instanzbeziehungen in $\mathcal{ALE}$

Für die Formulierung der hinreichenden Bedingung müssen zunächst noch einige Notationen auf $\mathcal{ALE}$ übertragen werden.
Ein *$\mathcal{ALE}$-Graph* $\mathcal{G} = (V, E, \ell)$ ist ein beschrifteter Graph, wobei

- $E \subseteq V \times N_R \times V \cup V \times \{\forall r \mid r \in N_R\} \times V$ und
- $\ell(v) \subseteq N_C \cup \{\neg P \mid P \in N_C\} \cup \{\bot\}$ für alle $v \in V$.

Der $\mathcal{ALE}$-Graph zu einer $\mathcal{ALE}$-ABox $\mathcal{A}$ wird formal wie folgt definiert:

Definition 7.27 ($\mathcal{ALE}$-Graph zur $\mathcal{ALE}$-ABox) *Sei $\mathcal{A}$ eine $\mathcal{ALE}$-ABox. Für alle $a \in \mathit{Ind}(\mathcal{A})$ sei $C_a := \bigsqcap_{a:D\in\mathcal{A}} D$, falls eine Konzeptassertion der Form $a : D$ in $\mathcal{A}$ existiert; sonst sei $C_a := \top$. Die $\mathcal{ALE}$-Beschreibungsbäume zu C_a, $a \in \mathit{Ind}(\mathcal{A})$, werden mit $\mathcal{G}(C_a) = (V_a, E_a, a, \ell_a)$ bezeichnet, wobei die Knotenmengen V_a o.B.d.A. paarweise disjunkt seien.*

*Der $\mathcal{ALE}$-*Graph *$\mathcal{G}(\mathcal{A})$* zu *$\mathcal{A}$ ist definiert durch $\mathcal{G}(\mathcal{A}) := (V, E, \ell)$ mit*

- $V := \bigcup_{a\in \mathit{Ind}(\mathcal{A})} V_a$,
- $E := \{arb \mid (a, b) : r \in \mathcal{A}\} \cup \bigcup_{a\in \mathit{Ind}(\mathcal{A})} E_a$ *und*
- $\ell(v) := \ell_a(v)$ *für* $v \in V_a$.

Für den Begriff des Homomorphismus zwischen einem $\mathcal{ALE}$-Beschreibungsbaum und einem $\mathcal{ALE}$-Graphen ist die Definition eines Homomorphismus zwischen $\mathcal{ALE}$-Beschreibungsbäumen nur leicht zu modifizieren: Eine Abbildung $\varphi : V_C \longrightarrow V$ ist ein Homomorphismus vom $\mathcal{ALE}$-Beschreibungsbaum $\mathcal{H} = (V_H, E_H, v_0, \ell_H)$ in den $\mathcal{ALE}$-Graphen $\mathcal{G} = (V, E, \ell)$, wenn φ die Bedingungen (2) und (3) aus Definition 5.19 erfüllt. Die erste Bedingung der Definition fließt wieder explizit in die hinreichende Bedingung für die Instanzbeziehung $a \in_{\mathcal{A}} C$ ein.

Lemma 7.28 *Sei $\mathcal{A}$ eine $\mathcal{ALE}$-ABox, $a \in Ind(\mathcal{A})$ und C eine $\mathcal{ALE}$-Konzeptbeschreibung. Existiert ein Homomorphismus φ von $\mathcal{G}(C) = (V_C, E_C, w_0, \ell_C)$ nach $\mathcal{G}(\mathcal{A})$ mit $\varphi(w_0) = a$, so gilt $a \in_{\mathcal{A}} C$.*

Beweis: Ist $\mathcal{A}$ inkonsistent, so ist nichts zu zeigen. Andernfalls sei $\mathcal{I}$ Modell von $\mathcal{A}$ und $C_a = \bigsqcap\limits_{a:D \in \mathcal{A}} D$ mit $\mathcal{G}(C_a) = (V_a, E_a, a, \ell_a)$. Aus der Annahme $\mathcal{I} \models \mathcal{A}$ folgt $a^{\mathcal{I}} \in C_a^{\mathcal{I}}$. Zeige $a^{\mathcal{I}} \in C^{\mathcal{I}}$ durch Induktion nach $depth(C)$:

$depth(C) = 0$: Dann ist $C = Q_1 \sqcap \ldots \sqcap Q_n$ mit $Q_i \in N_C \cup \{\neg P \mid P \in N_C\} \cup \{\top, \bot\}$. Zeige $a^{\mathcal{I}} \in Q_i^{\mathcal{I}}$ für alle $1 \leq i \leq n$.

Für $Q_i = \top$ ist nichts zu zeigen.

Angenommen, $Q_i = \bot$. Dann wäre wegen $\bot \in \ell(v)$ auch $C_a \equiv \bot$ im Widerspruch zur Annahme, dass $\mathcal{A}$ konsistent ist.

Angenommen, $Q_i \in N_C$ oder $Q_i = \neg P$ für ein $P \in N_C$. Nach Voraussetzung ist dann $P \in \ell(a)$ bzw. $\neg P \in \ell(a)$. Aus der Definition von $\mathcal{G}(C_a)$ und C_a folgt dann $C_a \sqsubseteq P$ bzw. $C_a \sqsubseteq \neg P$ und somit $a^{\mathcal{I}} \in P^{\mathcal{I}}$ bzw. $a^{\mathcal{I}} \in (\neg P)^{\mathcal{I}}$.

$depth(C) > 0$: Dann ist $C = Q_1 \sqcap \ldots \sqcap Q_n \sqcap \exists r_1.C_1 \sqcap \ldots \sqcap \exists r_m.C_m \sqcap \forall s_1.D_1 \sqcap \ldots \sqcap \forall s_k.D_k$ mit $Q_i \in N_C \cup \{\neg P \mid P \in N_C\} \cup \{\top \bot\}$. Zeige $a^{\mathcal{I}} \in C'^{\mathcal{I}}$ für alle Konjunkte C' auf Toplevel von C.

Für $C' = Q_i$ folgt die Behauptung wie im Induktionsanfang.

Sei $C' = \forall s_j.D_j$ und $w \in V_C$ der $\forall s_j$-Nachfolger von w_0 mit $\mathcal{G}(C)(w) = \mathcal{G}(D_j)$. Nach Voraussetzung existiert ein $\forall s_j$-Nachfolger v von a in $\mathcal{G}(\mathcal{A})$ mit $\varphi(w) = v$. Desweiteren folgt aus der Konstruktion von $\mathcal{G}(\mathcal{A})$, dass v die Wurzel eines Unterbaumes in $\mathcal{G}(C_a)$ bildet. Insbesondere folgt $C_a \sqsubseteq \forall s_j.C_{\mathcal{G}(C_a)(v)}$. Also folgt aus $a^{\mathcal{I}} \in C_a^{\mathcal{I}}$ auch $a^{\mathcal{I}} \in (\forall s_j.C_{\mathcal{G}(C_a)(v)})^{\mathcal{I}}$. Da nach Voraussetzung $\mathcal{G}(C)(w)$ homomorph in $\mathcal{G}(C_a)(v)$ eingebettet werden kann, folgt mit Lemma 5.22 $C_{\mathcal{G}(C_a)(v)} \sqsubseteq C_{\mathcal{G}(C)(w)}$ und damit $a^{\mathcal{I}} \in (\forall s_j.D_j)^{\mathcal{I}}$.

Sei schließlich $C' = \exists r_j.C_j$ und $w \in V_C$ der r_j-Nachfolger von w_0 mit $\mathcal{G}(C)(w) = \mathcal{G}(C_j)$. Nach Voraussetzung existiert ein r_j-Nachfolger v von a in $\mathcal{G}(\mathcal{A})$ mit $\varphi(w) = v$. Ist $v \notin Ind(\mathcal{A})$, so folgt aus der Konstruktion von $\mathcal{G}(\mathcal{A})$, dass v die Wurzel eines Unterbaumes in $\mathcal{G}(C_a)$ bildet. Wie im vorherigen Fall folgt dann

$a^\mathcal{I} \in (\exists r_j.C_j)^\mathcal{I}$. Ist $v \in \mathit{Ind}(\mathcal{A})$, dann liefert φ eingeschränkt auf die Knoten in $\mathcal{G}(C)(w)$ einen Homomorphismus ψ von $\mathcal{G}(C)(w)$ nach $\mathcal{G}(\mathcal{A})$ mit $\psi(w) = v$. Per Induktion folgt $v \in_\mathcal{A} C_{\mathcal{G}(C)(w)}$. Mit $\mathcal{I} \models \mathcal{A}$ folgt daraus $v^\mathcal{I} \in C^\mathcal{I}_{\mathcal{G}(C)(w)}$ und wegen $C_j \equiv C_{\mathcal{G}(C)(w)}$ auch $a^\mathcal{I} \in (\exists r_j.C_j)^\mathcal{I}$. □

Das folgende Beispiel zeigt nun, dass die im vorherigen Lemma formulierte hinreichende Bedingung nicht notwendig für die Instanzbeziehung $a \in_\mathcal{A} C$ ist.

Beispiel 7.29 *Betrachte die $\mathcal{ALE}$-ABox*

$$\begin{aligned}\mathcal{A} \ := \ & \{a : P, b_1 : P \sqcap \forall s.P \sqcap \exists r.P, b_2 : P \sqcap \exists r.(P \sqcap \exists s : P),\\ & (a, b_1) : r, (a, b_2) : r, (b_1, b_2) : r\}.\end{aligned}$$

Der zugehörige $\mathcal{ALE}$-Graph ist links in Abbildung 7.5 abgebildet, rechts der $\mathcal{ALE}$-Beschreibungsbaum zu

$$C = \exists r.(\forall s.P \sqcap \exists r.\exists s.\top).$$

Man beachte, dass $\mathcal{G}(\mathcal{A})$ primitiv vollständig, $\mathcal{G}(C)$ in $\top$-Normalform und alle Konzeptbeschreibungen aus Konzeptassertionen in $\mathcal{A}$ in $\mathcal{ALE}$-Normalform sind.
Es existiert aber kein Homomorphismus φ von $\mathcal{G}(C)$ nach $\mathcal{G}(\mathcal{A})$ mit $\varphi(w_0) = a$: bildet man w_1 auf b_1 ab, so findet man keinen rs-Nachfolger von b_1, auf den w_4 abgebildet werden kann; bildet man andererseits w_1 auf b_2 ab, so existiert kein $\forall s$-Nachfolger von b_2, auf den w_3 abgebildet werden kann.
Es gilt aber $a \in_\mathcal{A} C$: In jedem Modell $\mathcal{I}$ von $\mathcal{A}$ hat $b_2^\mathcal{I}$ entweder keinen s-Nachfolger oder mindestens einen s-Nachfolger. Im ersten Fall gilt $b_2^\mathcal{I} \in \forall s.P$ und damit liefert $b_2^\mathcal{I}$ den geforderten r-Nachfolger von $a^\mathcal{I}$ in $(\forall s.P \sqcap \exists r.\exists s.\top)^\mathcal{I}$. Andernfalls ist $b_2^\mathcal{I} \in (\exists s.\top)^\mathcal{I}$ und $b_1^\mathcal{I}$ liefert den gewünschten r-Nachfolger von $a^\mathcal{I}$. In jedem Modell gilt also $a^\mathcal{I} \in C^\mathcal{I}$.

Es ist offen, wie eine Charakterisierung der Instanz in $\mathcal{ALE}$ basierend auf Homomorphismen zwischen Bäumen und Graphen zu formulieren ist. Intuitiv zeigt Beispiel 7.29, dass die Fallunterscheidung aus $\mathcal{EL}_\neg$, ob P oder $\neg P$ für einen Knoten v in $\mathcal{G}(\mathcal{A})$ gilt, auf größere Konzeptbeschreibungen ausgedehnt werden müßte. Es ist aber nicht klar, für welche endliche Menge eine solche Fallunterscheidung genügen würde. Man beachte außerdem, dass die im obigen Beispiel betrachtete ABox $\mathcal{A}$ und die Konzeptbeschreibung C aus $\mathcal{FLE}$ sind. Dies zeigt, dass die Ursache für das aufgeworfene Problem in den Wertererestriktionen und nicht (wie bei $\mathcal{EL}_\neg$) in Inkonsistenzen liegt. Darüberhinaus ist es nicht durch Propagieren von Wertererestriktionen zu lösen, da keine der Normalisierungsregeln aus Definition 5.18 anwendbar ist. Wie in einer geeigneten Charakterisierung der Instanz in $\mathcal{ALE}$ – „geeignet" im Sinne von: sie liefert intuitiv ein Verfahren zur Berechnung (der k-Approximation) des MSC – Wertererestriktionen und Inkonsistenzen zu behandeln sind, ist ein offenes Problem und bleibt im Rahmen dieser Arbeit ungelöst.

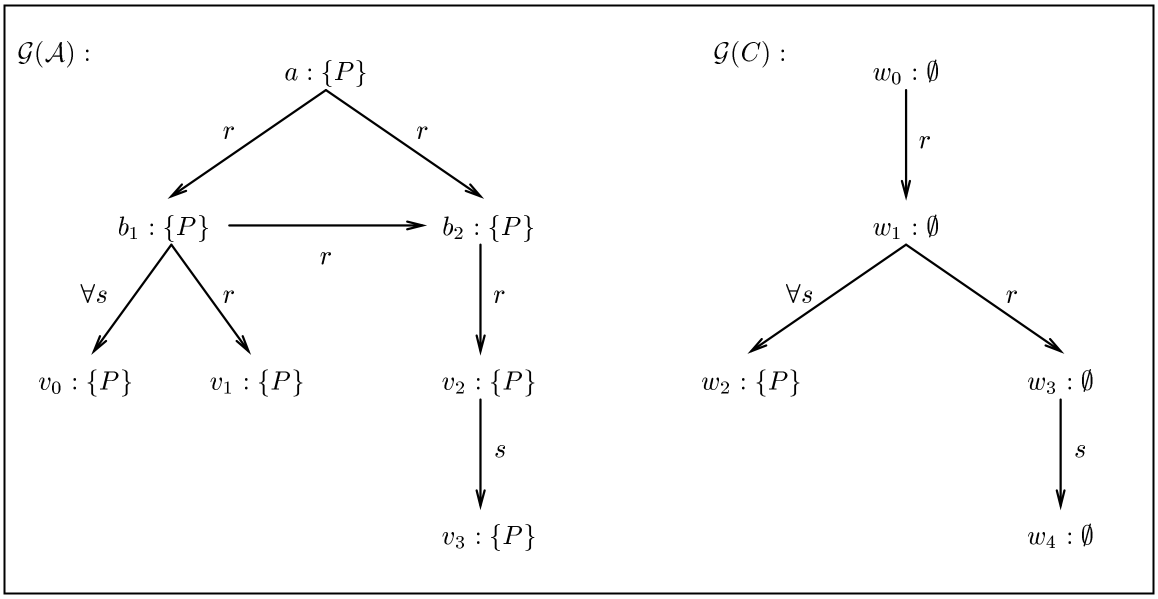


Abbildung 7.5: Beispiel zur Charakterisierung der Instanz in $\mathcal{ALE}$.

Eine Approximation der k-Approximation in $\mathcal{ALE}$

Zum Schluss dieses Abschnitts kommen wir noch zur angekündigten Approximation der k-Approximation. Die Definition der Bäume $\mathcal{T}(a,\mathcal{A})$ und $\mathcal{T}_k(a,\mathcal{G}(\mathcal{A}))$ überträgt sich auf natürliche Weise auf $\mathcal{ALE}$-ABoxen $\mathcal{A}$. Insbesondere ist also $\mathcal{T}_k(a,\mathcal{A}))$ ein $\mathcal{ALE}$-Beschreibungsbaum. Die hinreichende Bedingung aus Lemma 7.28 reicht nun aus zu zeigen, dass $a \in_{\mathcal{A}} C_{\mathcal{T}_k(a,\mathcal{G}(\mathcal{A}))}$.

Lemma 7.30 *Sei $\mathcal{A}$ eine $\mathcal{ALE}$-ABox. Für $a \in Ind(\mathcal{A})$ und $k \in \mathbb{N}$ gilt $a \in_{\mathcal{A}} C_{\mathcal{T}_k(a,\mathcal{G}(\mathcal{A}))}$.*

Beweis: Wie im Beweis von Theorem 7.9 kann eine Abbildung φ von $\mathcal{T}_k(a,\mathcal{G}(\mathcal{A}))$ nach $\mathcal{G}(\mathcal{A})$ definiert werden, die einen Homomorphismus mit $\varphi(a) = a$ bildet. Da wiederum $\mathcal{G}(C_{\mathcal{T}_k(a,\mathcal{G}(\mathcal{A}))})$ bis auf Knotenumbenennung gleich $\mathcal{T}_k(a,\mathcal{G}(\mathcal{A}))$ ist, folgt mit Lemma 7.28 die Behauptung. □

Im allgemeinen liefert die Konzeptbeschreibung $C_{\mathcal{T}_k(a,\mathcal{G}(\mathcal{A}))}$ aber natürlich nicht das speziellste Konzept zu a bzgl. $\mathcal{A}$ und k. Beispielsweise ist für die ABox $\mathcal{A}$ aus Beispiel 7.29

$$
\begin{aligned}
C_{\mathcal{T}_4(a,\mathcal{A})} \quad = \quad & P \sqcap \\
& \exists r.(P \sqcap \forall s.P \sqcap \exists r.P \sqcap \exists r.(P \sqcap \exists r.(P \sqcap \exists s.P))) \sqcap \\
& \exists r.(P \sqcap \exists r.(P \sqcap \exists s.P)).
\end{aligned}
$$

Für die Konzeptbeschreibung C (aus Beispiel 7.29) wurde $a \in_{\mathcal{A}} C$ nachgewiesen, also ist $a \in_{\mathcal{A}} C \sqcap C_{\mathcal{T}_4(a,\mathcal{A})}$. Wegen $C_{\mathcal{T}_4(a,\mathcal{A})} \not\sqsubseteq C$ folgt aber, dass $C_{\mathcal{T}_4(a,\mathcal{A})} \sqcap C \sqsubset C_{\mathcal{T}_4(a,\mathcal{A})}$, also ist $C_{\mathcal{T}_4(a,\mathcal{A})}$ nicht das speziellste Konzept zu a bzgl. $\mathcal{A}$ und Tiefe 4.

Um aus diesem einfachen Ansatz eine möglichst gute Approximation der k-Approximation zu erhalten, sollte vor der Berechnung von $\mathcal{T}_k(a, \mathcal{A})$ noch möglichst viel implizites Wissen in $\mathcal{A}$ explizit gemacht werden. Formal bedeutet dies z.B. die Werterestriktionen $\forall r.D$ aus Konzeptassertionen der Form $a : C$ entlang der Rollenassertionen der Form $(a, b) : r$ zu propagieren, d.h. die Regel

$\mathcal{A} \longrightarrow \mathcal{A} \cup \{b : D\}$, falls
$\{(a, b) : r, a : C\} \subseteq \mathcal{A}$ und
$\forall r.D$ eine Werterestriktion auf Toplevel von C und
existiert keine Konzeptassertion $b : C'$ in $\mathcal{A}$ mit $C' \sqsubseteq D$

erschöpfend auf $\mathcal{A}$ anzuwenden. Bezeichnet $\mathcal{A}^*$ die so erhaltene ABox, so ist intuitiv klar, dass die Konzeptbeschreibung $C_{\mathcal{T}_k(a,\mathcal{G}(\mathcal{A}^*))}$ im allgemeinen spezieller sein wird als $C_{\mathcal{T}_k(a,\mathcal{G}(\mathcal{A}))}$, im allgemeinen also eine bessere Approximation der k-Approximation liefert.

Kapitel 8

Rewriting von Konzeptbeschreibungen

Dieses Kapitel stellt die technischen Resultate zu dem in Abschnitt 3.4.3 eingeführten minimalen Rewriting-Problem dar. Um einen Eindruck von der Komplexität dieses Problems zu bekommen, betrachtet Abschnitt 8.1 zunächst das zugehörige Entscheidungsproblem, d.h. die Frage, ob zu einer gegebenen Schranke k ein Rewriting mit Größe $\leq k$ existiert. Abschnitt 8.2 weist zunächst Berechenbarkeit minimaler Rewritings in allen betrachteten BLen nach, allerdings unter Verwendung eines nicht praktikablen Algorithmus. Im nachfolgenden Abschnitt 8.3 wird daher ein verbesserter Algorithmus zur Berechnung minimaler Rewritings in $\mathcal{ALE}$ vorgestellt. Ausgehend von diesem verbesserten aber immer noch nicht-deterministischen Algorithmus führt schließlich Abschnitt 8.4 einen heuristischen Rewriting-Algorithmus für $\mathcal{ALE}$ ein, der auch für die Anwendung implementiert wurde, und untersucht formal die Qualität der mit ihm berechneten Rewritings.

8.1 Das Entscheidungsproblem zum minimalen Rewriting-Problem

Um eine Einschätzung der Komplexität des minimalen Rewriting-Problems für die BLen $\mathcal{FL}_0$, $\mathcal{ALN}$, $\mathcal{ALE}$ und $\mathcal{ALC}$ zu bekommen, betrachtet dieser Abschnitt zunächst das durch dieses Optimierungsproblem induzierte Entscheidungsproblem:

Gegeben: Eine $\mathcal{L}$-Konzeptbeschreibung C, eine $\mathcal{L}$-TBox $\mathcal{T}$ und eine natürliche Zahl $k \in \mathbb{N}$.

Frage: Gibt es ein $\mathcal{L}$-Rewriting D von C bzgl. $\mathcal{T}$ mit $|D| \leq k$?

Als Größenmaß wird dabei das in Definition 3.17 festgelegte Maß zugrunde gelegt.

Da sich dieses Entscheidungsproblem offensichtlich auf das Problem, ein minimales Rewriting zu berechnen, reduzieren lässt, übertragen sich Härteresultate für dieses Problem auf das Optimierungsproblem. Die folgenden Unterabschnitte liefern nun obere und untere Schranken für das Entscheidungsproblem in den betrachteten BLen.

8.1.1 NP-Härte für $\mathcal{FL}_0$, $\mathcal{ALN}$ und $\mathcal{ALE}$

Zum Nachweis der NP-Härte des minimal Rewriting-Problems in $\mathcal{FL}_0$ geben wir eine Reduktion des NP-vollständigen Problems SETCOVER [GJ79] an. Da diese Reduktion auch noch bei Hinzunahme von primitiver Negation und Zahlenrestriktionen bzw. Existenzrestriktionen gültig ist, liefert sie auch NP-Härte für $\mathcal{ALN}$ und $\mathcal{ALE}$.

Eine Instanz des SETCOVER-Problems ist von der folgenden Form:

Gegeben: Eine endliche Menge $\mathcal{U} = \{u_1, \ldots, u_n\}$, eine natürliche Zahl $\kappa \in \mathbb{N}$ und eine Familie $\mathcal{F} = \{F_i \subseteq \mathcal{U} \mid 1 \leq i \leq m\}$ von Teilmengen von $\mathcal{U}$.

Frage: Gibt es eine Teilmenge $\{F_{i_1}, \ldots, F_{i_k}\}$ von $\mathcal{F}$ der Größe $k \leq \kappa$ mit $F_{i_1} \cup \ldots \cup F_{i_k} = \mathcal{U}$?

Offensichtlich kann man sich auf Instanzen des Problems beschränken, bei denen mindestens die ganze Familie $\mathcal{F}$ eine Überdeckung von $\mathcal{U}$ liefert, d.h. im folgenden gelte o.B.d.A. $F_1 \cup \ldots \cup F_m = \mathcal{U}$.

Für eine gegebene Instanz $(\mathcal{U}, \mathcal{F}, \kappa)$ des SETCOVER-Problems betrachten wir $\mathcal{U}$ als Menge primitiver Konzeptnamen und definieren die zugehörige Instanz $(C_\mathcal{U}, \mathcal{T}_\mathcal{F}, \kappa)$ des minimalen Rewriting-Entscheidungsproblems in $\mathcal{FL}_0$ durch

$$\begin{aligned} C_\mathcal{U} &= u_1 \sqcap \ldots \sqcap u_n, \\ \mathcal{T}_\mathcal{F} &= \{A_j \doteq \bigsqcap_{u \in F_j} u \mid 1 \leq j \leq m\}. \end{aligned}$$

Da $C_\mathcal{U}$ und $\mathcal{T}_\mathcal{F}$ offensichtlich polynomiell in der Größe von $(\mathcal{U}, \mathcal{F}, \kappa)$ sind, folgt NP-Härte für das minimale Rewriting-Entscheidungsproblem in $\mathcal{FL}_0$ unmittelbar aus dem folgenden Lemma.

Lemma 8.1 *Es existiert ein minimales Rewriting D von $C_\mathcal{U}$ bzgl. $\mathcal{T}_\mathcal{F}$ mit $|D| \leq \kappa$ genau dann, wenn eine Überdeckung von $\mathcal{U}$ mit $k \leq \kappa$ Mengen $F_{i_1}, \ldots, F_{i_k}$ aus $\mathcal{F}$ existiert.*

Beweis: Ein Rewriting von $C_\mathcal{U}$ bzgl. $\mathcal{T}_\mathcal{F}$ der Größe $k \leq \kappa$ ist von der Form $D = A_{i_1} \sqcap \ldots \sqcap A_{i_l} \sqcap v_{l+1} \sqcap \ldots \sqcap v_k$ für ein $1 \leq l \leq k$ und $v_j \in \mathcal{U}$. Wir zeigen zunächst, dass wir o.B.d.A. annehmen können, dass D keine primitiven Konzeptnamen enthält, d.h. dass $l = k$: Da nach Annahme $\mathcal{F}$ die Menge $\mathcal{U}$ überdeckt, existiert für alle $l+1 \leq j \leq k$ ein $F_{i_j} \in \mathcal{F}$ mit $v_j \in F_{i_j}$. Ersetzt man also jedes v_j durch A_{i_j}, so erhält man ein Rewriting D' von $C_{\mathcal{U}}$, das keine primitiven Konzeptnamen enthält und es gilt $|D| = |D'|$.

Sei nun also $D = A_{i_1} \sqcap \ldots \sqcap A_{i_k}$ ein Rewriting von $C_{\mathcal{U}}$, das keine primitiven Konzeptnamen enthält. Dann impliziert $C_{\mathcal{U}} \equiv_{\mathcal{T}_{\mathcal{F}}} D$, dass für jedes $u \in \mathcal{U}$ ein definierter Name A_{i_j} in D existiert, sodass u auf der rechten Seite der Konzeptdefinition von A_{i_j} auftritt. Also ist $F_{i_1} \cup \ldots \cup F_{i_k}$ eine Überdeckung von $\mathcal{U}$ der Größe $k \leq \kappa$.

Sei umgekehrt $F_{i_1} \cup \ldots \cup F_{i_k}$ eine Überdeckung von $\mathcal{U}$ der Größe $k \leq \kappa$. Dann ist $D := A_{i_1} \sqcap \ldots \sqcap A_{i_k}$ ein Rewriting von $C_{\mathcal{U}}$ bzgl. $\mathcal{T}_{\mathcal{F}}$ mit Größe $\leq \kappa$. □

Lemma 8.1 gilt auch, wenn man $C_{\mathcal{U}}$ als $\mathcal{ALN}$- bzw. $\mathcal{ALE}$-Konzeptbeschreibung und $\mathcal{T}_{\mathcal{F}}$ als $\mathcal{ALN}$- bzw. $\mathcal{ALE}$-TBox auffasst: Ein $\mathcal{ALE}$-/$\mathcal{ALN}$-Rewriting von $C_{\mathcal{U}}$ bzgl. $\mathcal{T}_{\mathcal{F}}$ ist von der Form $D = A_{i_1} \sqcap \ldots \sqcap A_{i_l} \sqcap v_{l+1} \sqcap \ldots \sqcap v_k \sqcap D'$ mit $D' \equiv \top$; sonst wäre $D \not\equiv_{\mathcal{T}_{\mathcal{F}}} C_{\mathcal{U}}$. In diesem Fall ist auch $A_{i_1} \sqcap \ldots \sqcap A_{i_l} \sqcap v_{l+1} \sqcap \ldots \sqcap v_k$ ein Rewriting von $C_{\mathcal{U}}$ und wie im Beweis für $\mathcal{FL}_0$ folgt, dass eine Überdeckung von $\mathcal{U}$ der Größe $k \leq \kappa$ existiert. Insgesamt erhalten wir

Satz 8.2 *Das minimale Rewriting-Entscheidungsproblem ist NP-hart für $\mathcal{FL}_0$, $\mathcal{ALN}$ und $\mathcal{ALE}$.*

8.1.2 PSPACE-Härte für $\mathcal{ALC}$

Das folgende Lemma liefert eine Reduktion von Subsumtion in $\mathcal{ALC}$ auf das Entscheidungsproblem zum minimalen Rewriting-Problem in $\mathcal{ALC}$. Da Subsumtion in $\mathcal{ALC}$ ein PSPACE-vollständiges Problem ist, folgt damit PSPACE-Härte für das Entscheidungsproblem zum minimalen Rewriting-Problem in $\mathcal{ALC}$.

Lemma 8.3 *Seien C, D zwei $\mathcal{ALC}$-Konzeptbeschreibungen und A, P_1, P_2 drei verschiedene Konzeptnamen, die nicht in C und D vorkommen. Dann gilt $C \sqsubseteq D$ genau dann, wenn ein minimales Rewriting E der Größe ≤ 1 der $\mathcal{ALC}$-Konzeptbeschreibung $P_1 \sqcap P_2 \sqcap C$ bzgl. der $\mathcal{ALC}$-TBox $\mathcal{T} = \{A \doteq P_1 \sqcap P_2 \sqcap C \sqcap D\}$ existiert.*

Beweis: Gelte $C \sqsubseteq D$. Dann folgt $C \equiv C \sqcap D$ und damit $P_1 \sqcap P_2 \sqcap C \equiv P_1 \sqcap P_2 \sqcap C \sqcap D$. Also ist A ein Rewriting der Größe 1 von $P_1 \sqcap P_2 \sqcap C$ bzgl. $\mathcal{T}$. Insbesondere existiert also ein minimales Rewriting der Größe kleiner gleich 1.

Sei umgekehrt E ein minimales Rewriting von $P_1 \sqcap P_2 \sqcap C$ bzgl. $\mathcal{T}$ mit Größe kleiner gleich 1. Wir unterscheiden verschiedene Fälle.

1. $E = A$: Es folgt $P_1 \sqcap P_2 \sqcap C \equiv P_1 \sqcap P_2 \sqcap C \sqcap D$. Da P_1 und P_2 primitive Konzeptnamen sind, die nicht in C, D vorkommen, folgt $C \equiv C \sqcap D$ und damit $C \sqsubseteq D$.

2. $E = \bot$: Es ist $P_1 \sqcap P_2 \sqcap C \equiv \bot$. Da P_1, P_2 primitive Konzeptnamen sind, die in C nicht vorkommen, folgt $C \equiv \bot$ und damit $C \sqsubseteq D$.

3. $E = \top$: Es wäre $P_1 \sqcap P_2 \sqcap C \equiv \top$ im Widerspruch zu $P_1 \sqcap P_2 \sqsubset \top$.

4. $E = Q$ für einen Konzeptnamen Q ungleich A:

   (a) $Q \in \{P_1, P_2\}$: Es sei o.B.d.A. $Q = P_1$. Dann folgt $P_1 \equiv P_1 \sqcap P_2 \sqcap C$. Dies impliziert $P_1 \sqsubseteq P_2$ im Widerspruch dazu, dass P_1 und P_2 verschiedene primitive Konzeptnamen sind.

   (b) $Q \notin \{A, P_1, P_2\}$: Dann folgt $Q \equiv P_1 \sqcap P_2 \sqcap C$. Dies impliziert $Q \sqsubseteq P_1$ im Widerspruch dazu, dass P_1 und Q verschiedene primitive Konzeptnamen sind.

5. $E = \forall r.E'$ mit $|E'| = 0$: Dann ist $E' \equiv_{\mathcal{T}} \bot$ oder $E' \equiv_{\mathcal{T}} \top$. Angenommen $E' \equiv_{\mathcal{T}} \top$. Dann ist aber $E \equiv_{\mathcal{T}} \top$ und $|E| = 1 > 0 = |\top|$ im Widerspruch zur Minimalität von E. Angenommen, $E' \equiv_{\mathcal{T}} \bot$. Dann folgt $\forall r.\bot \equiv_{\mathcal{T}} P_1 \sqcap P_2 \sqcap C$ und insbesondere $\forall r.\bot \sqsubseteq P_1$ im Widerspruch dazu, dass P_1 ein primitiver Konzeptname ist.

6. $E = \exists r.E'$ mit $|E'| = 0$: Dann ist $E' \equiv_{\mathcal{T}} \bot$ oder $E' \equiv_{\mathcal{T}} \top$. Angenommen, $E' \equiv_{\mathcal{T}} \bot$. Dann ist aber $E \equiv_{\mathcal{T}} \bot$ und $|E| = 1 > 0 = |\bot|$ im Widerspruch zur Minimalität von E. Angenommen, $E' \equiv_{\mathcal{T}} \top$. Dann folgt $\exists r.\top \equiv_{\mathcal{T}} P_1 \sqcap P_2 \sqcap C$ und insbesondere $\exists r.\top \sqsubseteq P_1$ im Widerspruch dazu, dass P_1 ein primitiver Konzeptname ist.

7. $E = E' \sqcap E''$ oder $E = E' \sqcup E''$ oder $E = \neg E'$. Zeige die Behauptung durch Induktion nach der Anzahl der Vorkommen der Konstruktoren $\sqcap, \sqcup, \neg$.

   (a) $E = E' \sqcap E''$. Da $|E| \leq 1$ ist $|E'| = 0$ oder $|E''| = 0$. Sei o.B.d.A. $|E'| = 0$. Dann ist $E' \equiv_{\mathcal{T}} \bot$ oder $E' \equiv_{\mathcal{T}} \top$. Angenommen, $E' \equiv_{\mathcal{T}} \bot$. Dann ist $E \equiv_{\mathcal{T}} \bot$ und es folgt wie in 2. $C \sqsubseteq D$. Angenommen, $E' \equiv_{\mathcal{T}} \top$. Dann ist E'' ebenfalls ein minimales Rewriting von $P_1 \sqcap P_2 \sqcap C$ bzgl. $\mathcal{T}$ mit Größe kleiner gleich 1. Per Induktion folgt $C \sqsubseteq D$.

   (b) $E = E' \sqcup E''$. Da $|E| \leq 1$ ist $|E'| = 0$ oder $|E''| = 0$. Sei o.B.d.A. $|E'| = 0$, also $E' \equiv_{\mathcal{T}} \bot$ oder $E' \equiv_{\mathcal{T}} \top$. Angenommen, $E' \equiv_{\mathcal{T}} \top$. Dann ist $P_1 \sqcap P_2 \sqcap C \equiv E \equiv \top$ und wie in 3. erhalten wir einen Widerspruch. Angenommen, $E' \equiv_{\mathcal{T}} \bot$. Dann ist E'' ebenfalls ein minimales Rewriting von $P_1 \sqcap P_2 \sqcap C$ bzgl. $\mathcal{T}$ der Größe ≤ 1. Per Induktion folgt $C \sqsubseteq D$.

   (c) $E = \neg E'$ mit $|E'| \leq 1$. Wegen $E \equiv_{\mathcal{T}} \neg(\neg E)$ sei o.B.d.A. E' nicht von der Form $\neg E''$, d.h. E' sei $\top$ oder $\bot$, ein Konzeptname, eine Werte- oder Existenzrestriktion oder eine Konjunktion oder Disjunktion.

      i. Für $E' = \top$ folgt $C \sqsubseteq D$ wie in 2.

      ii. Für $E' = \bot$ folgt ein Widerspruch wie in 3.

      iii. Für $E' = Q$ für einen Konzeptnamen Q mit $Q \notin \{A, P_1, P_2\}$ folgt wie in 4.(b) ein Widerspruch.

iv. Für $E' \in \{P_1, P_2\}$ sei o.B.d.A. $E' = P_1$. Dann folgt $P_1 \sqcap P_2 \sqcap C \equiv \neg P_1$ und insbesondere $\neg P_1 \sqsubseteq P_1$, also ein Widerspruch.

v. Für $E' = A$ folgt $\neg(P_1 \sqcap P_2 \sqcap C \sqcap D) \equiv P_1 \sqcap P_2 \sqcap C$. Mit $\neg(P_1 \sqcap P_2 \sqcap C \sqcap D) \equiv \neg P_1 \sqcup \neg P_2 \sqcup \neg C \sqcup D$ folgt daraus $\neg P_1 \sqsubseteq P_1 \sqcap P_2 \sqcap C$. Dies impliziert $\neg P_1 \sqsubseteq P_1$, also wiederum einen Widerspruch.

vi. Für $E' = \exists r.E''$ ist $E \equiv \forall r.\neg E''$, wobei $|\neg E''| = 0$. Dieser Fall wird in 5. behandelt.

vii. Für $E' = \forall r.E''$ ist $E \equiv \exists r.\neg E''$, wobei $|\neg E''| = 0$. Dieser Fall wird in 6. behandelt.

viii. Für $E' = E_1 \sqcap E_2$ folgt mit $|E| \leq 1$ wieder $|E_1| = 0$ oder $|E_2| = 0$. Es sei o.B.d.A. $|E_1| = 0$, also $E_1 \equiv \top$ oder $E_1 \equiv \bot$. Ist $E_1 \equiv \top$, so ist $E \equiv \neg E_2$, wobei in $\neg E_2$ weniger boolescher Konstruktoren auftreten als in E. Also folgt per Induktion $C \sqsubseteq D$. Ist $E_1 \equiv \bot$, so ist $E \equiv \top$ und wie in 3. folgt ein Widerspruch.

ix. Für $E' = E_1 \sqcup E_2$ folgt mit $|E| \leq 1$ wieder $|E_1| = 0$ oder $|E_2| = 0$. Es sei o.B.d.A. $|E_1| = 0$, also $E_1 \equiv \top$ oder $E_1 \equiv \bot$. Ist $E_1 \equiv \top$, so ist $E \equiv \bot$ und wie in 2. folgt $C \sqsubseteq D$. Ist $E_1 \equiv \bot$, so ist $E \equiv \neg E_2$, wobei in $\neg E_2$ weniger boolesche Konstruktoren auftreten als in E. Also folgt per Induktion $C \sqsubseteq D$. □

Die in Lemma 8.3 formulierte Reduktion von Subsumtion in einer BL $\mathcal{L}$ auf das minimale Rewriting-Entscheidungsproblem in $\mathcal{L}$ gilt auch für Subsprachen von $\mathcal{ALC}$ (sofern sie Konjunktion erlauben) sowie für Erweiterungen von $\mathcal{ALC}$ wie z.B. $\mathcal{ALCN}$ oder $\mathcal{ALCNR}$. Damit folgt, dass für alle diese BLen das minimale Rewriting-Problem aus komplexitätstheoretischer Sicht mindestens so schwer ist wie das Subsumtionsproblem. Dies liefert insbesondere einen alternativen Beweis der NP-Härte für $\mathcal{ALE}$, nicht aber für $\mathcal{FL}_0$ und $\mathcal{ALN}$ (da für diese BLen das Subsumtionsproblem polynomiell entscheidbar ist).

8.1.3 Eine allgemeine obere Schranke

Nachdem in den vorherigen beiden Unterabschnitten untere Schranken nachgewiesen wurden, liefert dieser Abschnitt obere Schranken für alle betrachteten BLen.

Sei dazu $\mathcal{L}$ eine der BLen $\mathcal{FL}_0$, $\mathcal{ALN}$, $\mathcal{ALE}$ oder $\mathcal{ALC}$ und $\mathcal{A}$ ein Algorithmus, der Äquivalenz modulo TBox in $\mathcal{L}$ entscheidet. Dann entscheidet der folgende Algorithmus unter Verwendung von $\mathcal{A}$ als Orakel in nichtdeterministisch polynomieller Zeit, ob ein $\mathcal{L}$-Rewriting D von C bzgl. $\mathcal{T}$ existiert mit $|D| \leq k$:

1. Rate eine $\mathcal{L}$-Konzeptbeschreibung D mit $|D| \leq k$;

2. Teste $D \equiv_{\mathcal{T}} C$.

TBox	aufgefaltet	nicht aufgefaltet
$\mathcal{FL}_0$	NP-vollständig	NP-vollständig
$\mathcal{ALN}$	NP-vollständig	in Σ_2^p, NP-hart
$\mathcal{ALE}$	NP-vollständig	in PSPACE, NP-hart
$\mathcal{ALC}$	PSPACE-vollständig	PSPACE-vollständig

Tabelle 8.1: Komplexitätsresultate für das Entscheidungsproblem zum minimalen Rewriting-Problem.

Man beachte, dass der Äquivalenztest $D \equiv_{\mathcal{T}} C$ im zweiten Schritt ein Spezialfall des Äquivalenzproblems modulo TBox in $\mathcal{L}$ ist, da C *keine* definierten Konzeptnamen enthält. Tatsächlich kann man unter Verwendung der in [Baa96] für $\mathcal{FL}_0$ bzw. in [Küs98] für $\mathcal{ALN}$ eingeführten automatentheoretischen Charakterisierung der Subsumtion zeigen, dass für die BLen $\mathcal{FL}_0$ und $\mathcal{ALN}$ dieses *spezielle Äquivalenzproblem modulo TBox* aus komplexitätstheoretischer Sicht leichter ist als das allgemeine Äquivalenzproblem modulo TBox. Aus der Literatur ergeben sich für die hier betrachteten BLen die folgenden Komplexitätsresultate:

- Für $\mathcal{FL}_0$ ist das Äquivalenzproblem modulo TBox NP-vollständig [Neb90a], aber das spezielle Äquivalenzproblem modulo TBox ist in polynomieller Zeit entscheidbar [BKM99b].

- In $\mathcal{ALN}$ ist bei aufgefalteter TBox das Äquivalenzproblem ebenfalls in polynomieller Zeit entscheidbar [BP94]. Ist die TBox nicht aufgefaltet, so ist das Äquivalenzproblem modulo TBox in Π_2^p ($= \mathrm{coNP}^{\mathrm{NP}}$) und das spezielle Äquivalenzproblem in Δ_2^p ($= \mathrm{P}^{\mathrm{NP}}$) [BKM99b].

- In $\mathcal{ALE}$ ist bei aufgefalteter TBox das Äquivalenzproblem modulo TBox NP-vollständig [DLN+92]. Ist die TBox nicht aufgefaltet, so ist die Komplexität des (speziellen) Äquivalenzproblems ein offenes Problem: einerseits ist es NP-hart, da Subsumtion in $\mathcal{ALE}$ NP-hart ist [DLN+92]; andererseits ist das (spezielle) Äquivalenzproblem modulo TBox in PSPACE, da $\mathcal{ALE}$ eine Teilsprache von $\mathcal{ALC}$ ist.

- Äquivalenz in $\mathcal{ALC}$ ist PSPACE-vollständig [SS91], sogar modulo TBox [Lut99].

Insgesamt ergeben sich nun für das Entscheidungsproblem zum minimalen Rewriting-Problem die in Tabelle 8.1 zusammengefassten Komplexitätsresultate. Die Härteresultate stammen aus den vorhergehenden Unterabschnitten und die oberen Schranken ergeben sich aus dem oben skizzierten Algorithmus und den Komplexitätsresultaten für das (spezielle) Äquivalenzproblem modulo TBox.

Abschließend sei noch bemerkt, dass es offensichtlich zwei unabhängige Quellen für die Komplexität des minimalen Rewriting-Problems gibt: auf der einen Seite ist Äquivalenz modulo TBox zu entscheiden, um zu testen, ob tatsächlich ein *Rewriting* berechnet wurde. Auf der anderen Seite muss ein Optimierungsproblem gelöst werden, um ein *minimales* Rewriting zu berechnen. Da das spezielle Äquivalenzproblem modulo TBox für $\mathcal{FL}_0$ in polynomieller Zeit entscheidbar ist, folgt, dass dieses Optimierungsproblem hart ist, unabhängig von der Komplexität des Äquivalenzproblems.

8.2 Das Berechnungsproblem zum minimalen Rewriting-Problem

Nachdem der vorherige Abschnitt Algorithmen untersuchte, die entscheiden, ob ein (minimales) Rewriting innerhalb einer gegebenen Schranke für die Größe existiert, befasst sich dieser Abschnitt nun mit Algorithmen, die ein (minimales) Rewriting auch tatsächlich berechnen.

Die Härteresultate aus Abschnitt 8.1 implizieren, dass die Berechnung eines minimalen Rewritings ein hartes Problem, d.h. im allgemeinen also nicht in polynomieller Zeit lösbar ist. Darüber hinaus zeigt das folgende Beispiel, dass für alle betrachteten BLen die Anzahl minimaler Rewritings zu einer Konzeptbeschreibung C bzgl. einer TBox $\mathcal{T}$ exponentiell in der Größe von C, $\mathcal{T}$ und der zugrundeliegenden Signatur sein kann. Dabei ist die *Größe* $|\mathcal{T}|$ einer TBox $\mathcal{T}$ definiert als $|\mathcal{T}| := \sum_{A \doteq C \in \mathcal{T}} (1 + |C|)$, d.h. als die Summe der Größen der linken und rechten Seiten der Konzeptdefinitionen in $\mathcal{T}$.

Beispiel 8.4 *Für eine natürliche Zahl $n \in \mathbb{N}$ seien $N_C^n = \{P_i \mid 1 \leq i \leq n\} \cup \{A_i \mid 1 \leq i \leq n\}$ und $N_R^n = \emptyset$ sowie*

$$\begin{array}{rcl} C_n & := & P_1 \sqcap \ldots \sqcap P_n, \\ \mathcal{T}_n & := & \{A_i \doteq P_i \mid 1 \leq i \leq n\}. \end{array}$$

Definiere für jeden Vektor $\mathbf{i} = (i_1, \ldots, i_n) \in \{0,1\}^n$

$$E_{\mathbf{i}} := \bigsqcap_{1 \leq j \leq n, i_j = 0} P_j \sqcap \bigsqcap_{1 \leq j \leq n, i_j = 1} A_j.$$

Man sieht leicht, dass

1. *$E_{\mathbf{i}}$ ein Rewriting von C_n bzgl. $\mathcal{T}_n$ in allen betrachteten BLen $\mathcal{L}$ ist,*

2. *$|E_{\mathbf{i}}| = n = |C_n|$ für alle $\mathbf{i} \in \{0,1\}^n$ und*

3. *in $\mathcal{L}$ jeweils kein kleineres Rewriting von C_n bzgl. $\mathcal{T}_n$ existiert.*

Es folgt, dass die Anzahl verschiedener, d.h. bzgl. der leeren TBox nicht äquivalenter, minimaler Rewritings von C_n bzgl. $\mathcal{T}_n$ exponentiell groß sein kann bzgl. der Größe von C_n, $\mathcal{T}_n$ und den Größen der Mengen N_C und N_R.

Betrachtet man also eine Instanz $(C, \mathcal{T})$ des *minimalen Rewriting-Berechnungsproblems*, so sind zwei Ziele zu unterscheiden: entweder man ist an der Berechnung *eines* minimalen Rewritings von C bzgl. $\mathcal{T}$ interessiert oder an der Berechnung *aller* minimaler Rewritings von C bzgl. $\mathcal{T}$.

Ein *naiver Algorithmus* zur Berechnung eines minimalen Rewritings C bzgl. $\mathcal{T}$ würde alle Konzeptbeschreibungen der Größe $k = 0$, $k = 1$, $k = 2$, usw. aufzählen, bis er ein Rewriting D_0 von C bzgl. $\mathcal{T}$ findet. Nach Konstruktion ist D_0 minimal, und da C ein Rewriting von sich selbst ist, brauchen nur Konzeptbeschreibungen bis zur Größe $|C|$ berücksichtigt zu werden. Ist man an der Berechnung aller minimalen Rewritings interessiert, so sind noch alle Konzeptbeschreibungen der Größe $|D_0|$ aufzuzählen und für jede ist zu testen, ob sie ebenfalls äquivalent zu C modulo $\mathcal{T}$ ist.

Damit dieser naive Algorithmus tatsächlich minimale Rewritings berechnet, ist sicherzustellen, dass er stets terminiert, d.h., dass für alle $k \geq 0$ nur endlich viele Konzeptbeschreibungen der Größe k aufzuzählen sind. Hinreichend hierfür sind folgende Bedingungen:

1. Die Mengen N_C und N_R sind endlich.
2. Die Konzeptbeschreibungen werden modulo Kommutativität, Assoziativität und Idempotenz der Konjunktion aufgezählt.

Die zweite Bedingung muss aufgrund der Definition der Größe—genauer, wegen $|\top| = |\bot| = 0$—hinzugenommen werden. Andernfalls gäbe es (in allen betrachteten BLen) beispielsweise unendlich viele Konzeptbeschreibungen der Form $P \sqcap \top \sqcap \ldots \sqcap \top$, die alle Größe 1 haben. Beide Bedingungen liefern für praktische Anwendungen (insbesondere für unsere Anwendung in der Prozesstechnik) keine ernsthaften Einschränkungen. Zum einen betrachtet man in Anwendungen meistens endliche Signaturen und zum anderen sind Konzeptbeschreibungen, die sich nur modulo Kommutativität, Assoziativität und Idempotenz unterscheiden, äquivalent.

Unter den beiden Bedingungen berechnet der naive Algorithmus also ein bzw. alle minimalen Rewritings in allen betrachteten Beschreibungslogiken. Darüberhinaus liefert er die Basis für folgende Komplexitätsaussagen zum minimalen Rewriting-Berechnungsproblem:

Satz 8.5 *Seien N_C und N_R endlich, $\mathcal{L}$ eine der BLen $\mathcal{FL}_0$, $\mathcal{ALN}$, $\mathcal{ALE}$, $\mathcal{ALC}$, C eine $\mathcal{L}$-Konzeptbeschreibung und $\mathcal{T}$ eine $\mathcal{L}$-TBox. Dann gilt:*

1. *Ein minimales Rewriting von C bzgl. $\mathcal{T}$ kann mit (bzgl. der Größe von C, $\mathcal{T}$, N_C und N_R) polynomiellem Platz berechnet werden.*

2. *Die Menge* aller *minimalen Rewritings von C bzgl. $\mathcal{T}$ kann in (bzgl. der Größe von C, $\mathcal{T}$, N_C und N_R) exponentieller Zeit berechnet werden.*

Beweis: Ad (1): Offensichtlich können (bei endlicher Signatur und modulo Kommutativität, Assoziativität und Idempotenz der Konjunktion) alle $\mathcal{L}$-Konzeptbeschreibungen der Größe $\leq |C|$ mit (bzgl. der Größe von C, $\mathcal{T}$, N_C und N_R) polynomiellen Platz aufgezählt werden. Da desweiteren für jede der betrachteten BLen Äquivalenz modulo TBox in PSPACE entschieden werden kann (für $\mathcal{ALN}$ siehe [Küs98], für $\mathcal{FL}_0$, $\mathcal{ALE}$, $\mathcal{ALC}$ siehe [Lut99]), lässt sich also *ein* minimales Rewriting mit polynomiellem Platz berechnen.

Ad (2): Mit Beispiel 8.4 folgt für alle betrachteten BLen, dass die Zahl der minimalen Rewritings exponentiell groß sein kann. Da die Größe eines minimalen Rewritings durch die Größe von C beschränkt ist, kann es bzgl. der Größe von C, $\mathcal{T}$, N_C und N_R höchstens exponentiell viele minimale Rewritings geben, jedes mit einer Größe kleiner gleich der Größe von C. Folglich kann die Zeit zur Berechnung *aller* minimalen Rewritings exponentiell beschränkt werden. □

Der oben skizzierte naive Algorithmus ist offensichtlich sehr ineffizient. Sein entscheidender Nachteil ist, dass er nicht unmittelbar unter Verwendung der Eingabe C arbeitet: die potentiellen Rewritings werden jeweils ohne Berücksichtigung von C generiert. Im folgenden Abschnitt wird ein *verbesserter Rewriting-Algorithmus* vorgestellt, der versucht, diesen Nachteil zu umgehen, indem er ein Rewriting zwar ebenfalls nichtdeterministisch, aber durch eine direkte Manipulation des Eingabekonzeptes C berechnet. Motiviert durch die Prozesstechnikanwendung wurde dieser verbesserte Algorithmus zunächst für die BL $\mathcal{ALE}$ entwickelt. Die grundlegenden Begriffe und der Algorithmus können aber auch auf $\mathcal{ALN}$ übertragen werden [BKM99b]. Da $\mathcal{ALN}$ aber nicht im Focus dieser Arbeit liegt, wird hier auf die entsprechenden technischen Ausführungen verzichtet.

8.3 Ein Rewriting-Algorithmus für $\mathcal{ALE}$

Die grundlegende Idee zum verbesserten Rewriting-Algorithmus für $\mathcal{ALE}$ aus Abbildung 8.1 besteht darin, die Berechnung eines Rewritings in zwei Schritte aufzuspalten: Im ersten Schritt wird eine sogenannte Extension C^* von C bzgl. $\mathcal{T}$ berechnet.

Definition 8.6 (Extension bzgl. $\mathcal{T}$) *Sei C eine $\mathcal{ALE}$-Konzeptbeschreibung und $\mathcal{T}$ eine $\mathcal{ALE}$-TBox. Die $\mathcal{ALE}$-Konzeptbeschreibung C^* ist eine* Extension von C bzgl. $\mathcal{T}$ *genau dann, wenn $C \equiv_{\mathcal{T}} C^*$ und C^* aus C durch konjunktive Anbindung definierter Namen aus $\mathcal{T}$ an beliebigen Stellen in C erhalten werden kann.*

Im zweiten Schritt wird dann eine sogenannte *Reduktion von C^* bzgl. $\mathcal{T}$* berechnet, d.h. eine $\mathcal{ALE}$-Konzeptbeschreibung $\widehat{C}$ die (i) äquivalent zu C modulo $\mathcal{T}$ ist und (ii)

Eingabe: Eine $\mathcal{ALE}$-Konzeptbeschreibung C und eine $\mathcal{ALE}$-TBox $\mathcal{T}$.

Algorithmus:
Berechne eine Extension C^* von C;
Berechne eine Reduktion $\widehat{C}$ von C^* bzgl. $\mathcal{T}$;
Gib $\widehat{C}$ zurück.

Abbildung 8.1: Der verbesserte Rewriting-Algorithmus.

aus C^* durch Eliminieren aller Redundanzen hervorgeht. Das wesentliche Problem ist nun, sowohl eine geeignete formale Definition des Begriffes 'Reduktion' als auch einen Algorithmus zur Berechnung von Reduktionen zu finden. Dieses Problem wird in Unterabschnitt 8.3.1 gelöst. Zuvor wird noch

- die Vorgehensweise des verbesserten Algorithmus an einem Beispiel verdeutlicht und
- erklärt, in welchem Sinne der verbesserte Algorithmus korrekt ist und was er de facto berechnet.

Beispiel 8.7 (Zum verbesserten Rewriting-Algorithmus)
Betrachte die folgende Instanz $(C, \mathcal{T})$ des minimalen Rewriting-Berechnungsproblems:

$$\begin{aligned} C &= P \sqcap Q \sqcap \forall r.P \sqcap \exists r.(P \sqcap \exists r.Q) \sqcap \exists r.(P \sqcap \forall r.(Q \sqcap \neg Q)), \\ \mathcal{T} &= \{\ A_1 \doteq \exists r.Q,\ A_2 \doteq P \sqcap \forall r.P,\ A_3 \doteq \forall r.P\ \}. \end{aligned}$$

Die $\mathcal{ALE}$-Konzeptbeschreibung

$$C^* = P \sqcap Q \sqcap A_2 \sqcap \forall r.P \sqcap \exists r.(A_1 \sqcap P \sqcap \exists r.Q) \sqcap \exists r.(P \sqcap \forall r.(Q \sqcap \neg Q))$$

ist eine Extension von C bzgl. $\mathcal{T}$. Eine Reduktion von C^* erhält man durch Eliminieren von

- P und $\forall r.P$ auf Toplevel von C^*, da sie wegen A_2 redundant sind;
- P in den beiden existentiellen Restriktionen auf Toplevel von C^*, weil P dort jeweils wegen der Werterestriktion $\forall r.P$ redundant ist;
- $\exists r.Q$ in der Existenzrestriktion $\exists r.(A_1 \sqcap P \sqcap \exists r.Q)$, weil sie wegen A_1 redundant ist;

und Ersetzen von $Q \sqcap \neg Q$ durch $\bot$, da $\bot$ ein bzgl. $|\cdot|$ minimales inkonsistentes Konzept ist. Die so erhaltene Konzeptbeschreibung

$$\widehat{C} = Q \sqcap A_2 \sqcap \exists r.A_1 \sqcap \exists r.\forall r.\bot$$

ist äquivalent zu C modulo $\mathcal{T}$, d.h. $\widehat{C}$ ist ein Rewriting von C bzgl. $\mathcal{T}$. Man sieht leicht, dass $\widehat{C}$ sogar ein minimales Rewriting von C bzgl. $\mathcal{T}$ ist.

Zur Korrektheit des verbesserten Rewriting-Algorithmus

Die folgenden Beispiele zeigen, dass sowohl zu einer Konzeptbeschreibung C exponentiell viele verschiedene Extensionen existieren können, als auch zu einer Extension exponentiell viele verschiedene Reduktionen.

Beispiel 8.8 *Betrachte nochmals Beispiel 8.4: Für jeden Vektor* $\mathbf{i} = (i_1, \ldots, i_n) \in \{0,1\}^n$ *ist die Konzeptbeschreibung*

$$C^*_{\mathbf{i}} := C_n \sqcap \bigsqcap_{1 \le j \le n, i_j = 1} A_j$$

eine Extension von C_n *bzgl.* $\mathcal{T}_n$. *Die so erhaltenen Extensionen sind dabei alle paarweise verschieden, d.h. bzgl. der leeren TBox nicht äquivalent.*

Beispiel 8.9 *Für eine natürliche Zahl* $n \in \mathbb{N}$ *seien*

$$\begin{array}{rcl} \mathcal{T}_n & := & \{A_i \doteq P_i \mid 1 \le i \le n\}, \\ C_n & := & \bigsqcap_{1 \le i \le n} (\exists r.\top \sqcap \exists r.P_i), \ \ \textit{und} \\ C^*_n & := & \bigsqcap_{1 \le i \le n} (\exists r.A_i \sqcap \exists r.P_i). \end{array}$$

C^*_n *ist eine Extension von* C_n *bzgl.* $\mathcal{T}_n$. *Intuitiv erhält man eine Reduktion von* C^*_n, *indem man für jedes* $i \in \{1, \ldots, n\}$ *entweder* $\exists r.A_i$ *oder* $\exists r.P_i$ *aus* C^*_n *eliminiert. Entfernt man sowohl* $\exists r.A_i$ *als auch* $\exists r.P_i$, *so wäre die erhaltene Koneptbeschreibung offensichtlich nicht äquivalent zu* C^*_n. *Entfernt man weder* $\exists r.A_i$ *noch* $\exists r.P_i$, *so würde die erhaltene Konzeptbeschreibung noch Redundanzen enthalten, da das Entfernen eines dieser Konzepte eine äquivalente aber kleinere Konzeptbeschreibung liefern würde. Intuitiv ist also für jeden Vektor* $\mathbf{i} = (i_1, \ldots, i_n) \in \{0,1\}^n$

$$C^*_{n,\mathbf{i}} := \bigsqcap_{1 \le j \le n, i_j = 0} \exists r.A_j \sqcap \bigsqcap_{1 \le j \le n, i_j = 1} \exists r.P_j$$

eine Reduktion von C^*_n *bzgl.* $\mathcal{T}_n$.

Bei der Berechnung von Extension und Reduktion hat man also im allgemeinen jeweils exponentiell viele mögliche Ergebnisse. Der Algorithmus aus Abbildung 8.1 sollte daher als ein nichtdeterministischer Algorithmus betrachtet werden, der zunächst eine Extension rät und dann zu dieser eine Reduktion. Das folgende Theorem formuliert das in Unterabschnitt 8.3.2 zu beweisende Korrektheitsresultat für diesen Rewriting-Algorithmus. Dazu wird noch der Begriff der $\forall$-*Normalform* benötigt:

Definition 8.10 ($\forall$-Normalform) *Eine* $\mathcal{ALE}$*-Konzeptbeschreibung* C *ist in* $\forall$*-Normalform genau dann, wenn jede Konjunktion in* C *für jeden Rollennamen* $r \in N_R$ *höchstens eine Werterestriktion der Form* $\forall r.C'$ *enthält.*

Zu einer Konzeptbeschreibung C erhält man die zugehörige $\forall$-Normalform in polynomieller Zeit durch erschöpfende Anwendung der Regel $\forall r.C \sqcap \forall r.D \longrightarrow \forall r.(C \sqcap D)$.

Theorem 8.11 *Sei $\mathcal{T}$ eine $\mathcal{ALE}$-TBox und C eine $\mathcal{ALE}$-Konzeptbeschreibung in $\forall$-Normalform. Dann gilt für den Rewriting-Algorithmus aus Abbildung 8.1 mit Eingabe $C, \mathcal{T}$:*

1. *Jede Ausgabe des Algorithmus ist ein Rewriting von C bzgl. $\mathcal{T}$.*
2. *Die Menge aller durch den Algorithmus berechneten Rewritings enthält alle minimalen Rewritings von C bzgl. $\mathcal{T}$ modulo Kommutativität, Assoziativität und Idempotenz der Konjunktion und der Äquivalenz $C \sqcap \top \equiv C$.*

Berechnet man also nur eine Extension und zu dieser nur eine Reduktion, so liefert dies einen Algorithmus zur Berechnung eines Rewritings, welches i.a. natürlich nicht minimal sein wird. Allerdings bietet dieser Algorithmus die Möglichkeit, durch Verwendung von Heuristiken „kleine“ (statt minimale) Rewritings deterministisch in polynomieller Zeit (mit einem Orakel für Äquivalenz modulo TBox) zu berechnen (cf Abschnitt 8.4). Desweiteren wird sich aus dem Beweis von Theorem 8.11 und der formalen Definition des Begriffes ‘Reduktion’ ergeben, dass, wenn man alle Extensionen und zu jeder Extension eine Reduktion berechnet, diese Menge mindestens ein minimales Rewriting enthält.

Noch eine Bemerkung zur Berechnung der Menge aller minimalen Rewritings: wiederum auf Grund der Definition der Größe von $\bot$ und $\top$ als 0 ist es notwendig, die Menge der minimalen Rewritings nicht nur modulo Kommutativität und Assoziativität der Konjunktion zu betrachten, sondern auch modulo Idempotenz und der Äquivalenz $C \sqcap \top \equiv C$. Denn wegen $|\bot| = 0$ ist $\bot \sqcap \bot$ ebenso ein minimales Rewriting eines inkonsistenten Konzeptes C wie $\bot$. Ein „vernünftiger“ Rewriting-Algorithmus (und insbesondere der im folgenden Abschnitt vorgestellte Algorithmus zur Berechnung einer Reduktion) würde aber nur $\bot$ als minimales Rewriting (bzw. Reduktion) ausgeben. Also betrachten wir minimale Rewritings auch modulo Idempotenz (da $|F \sqcap \ldots \sqcap F| = |F| = 0$ für $F \in \{\top, \bot\}$) und modulo der Äquivalenz $C \sqcap \top \equiv C$ (da $|C \sqcap \top| = |C|$ für alle Konzeptbeschreibungen C). Trotz dieser Einschränkungen wird sich die gewählte Definition der Größe als sinnvoll erweisen, da sie zum einen eine intuitive aber dennoch formal saubere Definition und Berechnung von Reduktionen zulässt. Zum anderen wäre für andere „übliche“ Definitionen der Größe entweder die Aussage in Theorem 8.11 oder der Algorithmus aus Abbildung 8.1 komplexer zu formulieren (cf Beispiel 8.18 am Ende des folgenden Unterabschnitts).

8.3.1 Reduktionen von $\mathcal{ALE}$-Konzeptbeschreibungen

Eine Reduktion einer Konzeptbeschreibung C wurde oben informell beschrieben als eine Konzeptbeschreibung, die (i) äquivalent zu C modulo $\mathcal{T}$ ist und (ii) aus C durch

Eliminieren aller Redundanzen hervorgeht. Das Eliminieren von Teilen aus C fassen wir formal durch den Begriff der *Subbeschreibung*.[1] Intuitiv ist D eine Subbeschreibung von C, wenn D aus C erhalten werden kann durch

- Ersetzen von Inkonsistenzen durch $\bot$,
- Streichen von *primitiven* Konzeptnamen, Werte- und Existenzrestriktionen auf Toplevel von C und
- rekursivem Ersetzen der Konzeptbeschreibungen C' aus den verbliebenen Werte- und Existenzrestriktionen $\forall r.C'/\exists r.C'$ durch Subbeschreibungen von C'.

Die Einschränkung im ersten Punkt (nur inkonsistente Konzeptbeschreibungen dürfen durch $\bot$ ersetzt werden) soll sicherstellen, dass jede Subbeschreibung von C auch C subsumiert. Diese Eigenschaft wird im Beweis der Vollständigkeit des weiter unten vorgestellten Reduktionsalgorithmus benötigt. Leider reicht für die formale Definition einer Subbeschreibung ein Test der Form $C \equiv_{\mathcal{T}} \bot$ aber nicht aus, um alle Konzeptbeschreibungen zu erfassen, die gemäß dieser Intuition Subbeschreibung sind. Als Beispiel betrachte man die TBox $\mathcal{T} = \{A_1 \doteq \forall r.\neg P\}$ und die Konzeptbeschreibung $C = A_1 \sqcap \forall r.P$. Offensichtlich gilt für $D = A_1 \sqcap \forall r.\bot$, dass $C \equiv_{\mathcal{T}} D$, d.h. die Konzeptbeschreibung in der Werterestriktion auf Toplevel von C ist im Kontext der Werterestriktion im definierenden Konzept zu A_1 inkonsistent. Intuitiv sollte daher D eine Subbeschreibung von C sein. Aber D genügt nicht der obigen intuitiven Beschreibung, da $\bot$ keine zulässige Subbeschreibung von P wäre, im Rekursionsschritt also P nicht durch $\bot$ ersetzt werden dürfte. Dieses Problem wird in der formalen Definition dadurch gelöst, dass der Kontext F berücksichtigt wird, in dem eine Konzeptbeschreibung C auftritt, und $C \sqcap F$ auf Inkonsistenz modulo $\mathcal{T}$ getestet wird. So ist im obigen Beispiel der Kontext, in dem eine Subbeschreibung von P zu bestimmen ist, gleich $\neg P$. Man erhält $\bot$ als eine Subbeschreibung von P bzgl. dem Kontext $\neg P$ und insgesamt D als Subbeschreibung von C bzgl. dem Kontext $\top$.

Für die Definition des Begriffs Subbeschreibung und die formale Spezifikation des Reduktionsalgorithmus benötigen wir noch die folgenden Notationen: Sei $\mathcal{T}$ eine $\mathcal{ALE}$-TBox und C eine $\mathcal{ALE}$-Konzeptbeschreibung, die definierte Namen aus $\mathcal{T}$ enthalten darf. Die *aufgefaltete Konzeptbeschreibung* $\mathcal{T}(C)$ ist definiert als die Konzeptbeschreibung, die man aus C durch vollständiges Ersetzen definierter Namen in C durch die zugehörigen definierenden Konzepte in $\mathcal{T}$ erhält.[2] Die Menge aller definierten Konzeptnamen auf Toplevel von C wird mit *def*(C) bezeichnet und die Menge aller

[1] Bei der Betrachtung des Matching-Problems in $\mathcal{ALE}$ [BK00a, Küs00] wird der Begriff 'Subbeschreibung' ähnlich wie hier definiert, allerdings nur für Konzeptbeschreibungen ohne definierte Namen. Er wird dort im Rahmen der Formalisierung des Begriffes „interessanter Matcher“ verwendet.

[2] $\mathcal{T}(C)$ ist aufgrund der hier zugrunde gelegten Annahmen über TBoxen (azyklisch und ohne Mehrfachdefinitionen) wohldefiniert, kann aber exponentiell groß bzgl. der Größe von C sein (vgl. auch Abschnitt 3.2.2).

(negierten) primitiven Konzeptnamen mit $\mathit{prim}(C)$. Desweiteren definieren wir für einen Rollennamen $r \in N_R$

- $\mathit{val}_r(C)$ als die Konzeptbeschreibung D aus der eindeutigen Werterestriktion der Form $\forall r.D$ auf Toplevel der $\forall$-Normalform von C, wobei $\mathit{val}_r(C) := \top$, falls keine solche Werterestriktion existiert;

- $\mathit{exr}_r(C)$ als die Menge $\{C_1, \ldots, C_n\}$ von Konzeptbeschreibungen, die in Existenzrestriktionen der Form $\exists r.C_i$ auf Toplevel von C auftreten.

Definition 8.12 ($\mathcal{ALE}$-Subbeschreibung bzgl. $\mathcal{T}$ und F) *Sei $\mathcal{T}$ eine $\mathcal{ALE}$-TBox, F eine $\mathcal{ALE}$-Konzeptbeschreibung und C eine $\mathcal{ALE}$-Konzeptbeschreibung in $\forall$-Normalform, die definierte Konzeptnamen aus $\mathcal{T}$ enthalten darf. Die $\mathcal{ALE}$-Konzeptbeschreibung $\widehat{C}$ heißt* Subbeschreibung von C bzgl. $\mathcal{T}$ und F *genau dann, wenn*

1. *$\widehat{C} = C$, oder*

2. *$\widehat{C} = \bot$ und $C \sqcap F \equiv_{\mathcal{T}} \bot$; oder*

3. *$\widehat{C}$ aus C erhalten werden kann durch*

   (a) *Entfernen von (negierten) primitiven Konzeptnamen, Werte- und Existenzrestriktionen auf Toplevel von C,*

   (b) *Ersetzen der Konzeptbeschreibungen D aus den restlichen Werterestriktionen $\forall r.D$ durch Subbeschreibungen $\widehat{D}$ von D bzgl. $\mathcal{T}$ und $\mathit{val}_r(F \sqcap \mathcal{T}(F_1 \sqcap \ldots \sqcap F_m \sqcap A_1 \sqcap \ldots \sqcap A_n))$, wobei $\{F_1, \ldots, F_m\} = \mathit{def}(F)$ und $\{A_1, \ldots, A_n\} = \mathit{def}(C)$, und*

   (c) *Ersetzen der Konzeptbeschreibungen D aus den restlichen Existenzrestriktionen $\exists r.D$ durch Subbeschreibungen $\widehat{D}$ von D bzgl. $\mathcal{T}$ und $\mathit{val}_r(C \sqcap F \sqcap \mathcal{T}(F_1 \sqcap \ldots \sqcap F_m \sqcap A_1 \sqcap \ldots \sqcap A_n))$, wobei $\{F_1, \ldots, F_m\} = \mathit{def}(F)$ und $\{A_1, \ldots, A_n\} = \mathit{def}(C)$.*

Dass in einer Reduktion einer Konzeptbeschreibung C bzgl. einer TBox $\mathcal{T}$ *alle* Redundanzen eliminiert sein müssen, entspricht nun der Forderung nach einer *minimalen* Subbeschreibung von C bzgl. $\mathcal{T}$ und $\top$.

Definition 8.13 (Reduktion bzgl. $\mathcal{T}$) *Sei C eine $\mathcal{ALE}$-Konzeptbeschreibung in $\forall$-Normalform und $\mathcal{T}$ eine $\mathcal{ALE}$-TBox. Eine $\mathcal{ALE}$-Konzeptbeschreibung $\widehat{C}$ heißt* Reduktion von C bzgl. $\mathcal{T}$ *genau dann, wenn $\widehat{C}$ eine (bzgl. $|\cdot|$) minimale Subbeschreibung von C bzgl. $\mathcal{T}$ und $\top$ ist mit $\widehat{C} \equiv_{\mathcal{T}} C$.*

Zur Verdeutlichung betrachte man auch nochmals Beispiel 8.7. Dort ist die Konzeptbeschreibung $\widehat{C}$ eine Reduktion von C bzgl. $\mathcal{T}$ und $\top$, wohingegen die Konzeptbeschreibung $D = Q \sqcap \exists r.A_1 \sqcap \exists r.\forall r.\bot$ keine ist, da D keine Subbeschreibung von C bzgl. $\mathcal{T}$ und $\top$ ist: in einer Subbeschreibung dürfen ja keine *definierten* Konzeptnamen entfernt werden (sofern sie nicht in einer Werte- oder Existenzrestriktion auftreten, die vollständig entfernt wird).

Da im verbesserten Rewriting-Algorithmus die Reduktion stets nach der Extension berechnet wird, ist es durchaus sinnvoll, das explizite Entfernen einzelner definierter Konzeptnamen *nicht* zu erlauben. So wäre es zwar gegebenenfalls möglich, durch Entfernen definierter Konzeptnamen ein kleineres Rewriting zu erhalten, aber eben dieses Rewriting würde man auch erhalten, wenn die entsprechenden definierten Namen bei der Berechnung der Extension gar nicht erst hinzugefügt worden wären. Daher würde die Möglichkeit, auch einzelne definierte Namen bei der Berechnung von Reduktionen entfernen zu dürfen, lediglich den Grad des Nichtdeterminismus erhöhen ohne aber weitere Rewritings zu liefern.

Der im folgenden vorgestellte Algorithmus berechnet eine Reduktion $\widehat{C}$ von C bzgl. $\mathcal{T}$ in nicht-deterministisch polynomieller Zeit (mit einem Orakel für Äquivalenz modulo TBox). Intuitiv erfolgt die Berechnung top-down: Ist $C \equiv_{\mathcal{T}} \bot$, so $\widehat{C} := \bot$. Sonst sei $\forall r.C'$ die eindeutige (C sei in $\forall$-Normalform) Werterestriktion zu $r \in N_R$ auf Toplevel von C und $\{A_1, \ldots, A_n\}$ die Menge aller definierten Namen auf Toplevel von C. Im wesentlichen erhält man nun $\widehat{C}$ wie folgt:

1. Entferne das (negierte) primitive Konzept Q auf Toplevel von C, falls $A_1 \sqcap \ldots \sqcap A_m \sqsubseteq_{\mathcal{T}} Q$.
2. Entferne $\exists r.C_1$ auf Toplevel von C, falls
   (a) $A_1 \sqcap \ldots \sqcap A_m \sqcap \forall r.C' \sqsubseteq_{\mathcal{T}} \exists r.C_1$, oder
   (b) eine weitere Existenzrestriktion $\exists r.C_2$ auf Toplevel von C existiert mit $A_1 \sqcap \ldots \sqcap A_m \sqcap \forall r.C' \sqcap \exists r.C_2 \sqsubseteq_{\mathcal{T}} \exists r.C_1$.
3. Entferne $\forall r.C'$, falls $A_1 \sqcap \ldots \sqcap A_m \sqsubseteq_{\mathcal{T}} \forall r.C'$.
4. Alle Konzeptbeschreibungen D in den restlichen Werte- und Existenzrestriktionen werden rekursiv reduziert.

Die formale Spezifikation dieses Reduktionsalgorithmus ist aus zwei Gründen komplexer als die intuitive Beschreibung. Einen Grund hierfür liefert Fall 2(b): Es kann sein, dass die Subsumtionsbeziehung auch gilt, wenn C_1 und C_2 die Rollen tauschen. Liefert dann die rekursive Reduktion von C_2 eine kleinere Konzeptbeschreibung als die von C_1, so ist $\exists r.C_1$ zu entfernen. Sind die Reduktionen aber gleich groß, so muss durch eine nichtdeterministische Wahl entweder $\exists r.C_1$ oder $\exists r.C_2$ entfernt werden. Einen weiteren Grund liefert der 4. Schritt. Die Konzeptbeschreibungen D aus den verbliebenen Restriktionen können nicht ohne Berücksichtigung des Kontextes reduziert werden, in dem sie auftreten. Die Reduktion dieser Konzepte muss sowohl das Konzept C' als auch alle Konzepte D' aus Werterestriktionen der Form $\forall r.D'$ auf Toplevel der definierenden Konzepte zu $A_1, \ldots, A_n$ berücksichtigen. Beispielsweise wurde in Beispiel 8.7 der primitive Konzeptname P aus den Existenzrestriktionen auf Toplevel von C^* auf Grund der Werterestriktion $\forall r.P$ auf Toplevel von C^* entfernt. Die rekursive Formulierung des Reduktionsalgorithmus setzt einen dritten Parameter ein, der diesen Kontext aufnimmt. Formal berechnet ein solcher Algorithmus also Reduktionen bzgl. einer TBox $\mathcal{T}$ und einem Kontext gegeben durch eine Konzeptbeschreibung F (vgl. auch Definition 8.12).

Definition 8.14 (Reduktion bzgl. $\mathcal{T}$ und F) *Sei C eine $\mathcal{ALE}$-Konzeptbeschreibung in $\forall$-Normalform, $\mathcal{T}$ eine $\mathcal{ALE}$-TBox und F eine $\mathcal{ALE}$-Konzeptbeschreibung. Eine $\mathcal{ALE}$-Konzeptbeschreibung $\widehat{C}$ heißt* Reduktion von C bzgl. $\mathcal{T}$ und F *genau dann, wenn $\widehat{C}$ eine (bzgl. $|\cdot|$) minimale Subbeschreibung von C bzgl. $\mathcal{T}$ und F mit $\widehat{C} \sqcap F \equiv_{\mathcal{T}} C \sqcap F$ ist.*

Abbildung 8.2 enthält nun die formale Spezifikation des oben skizzierten Reduktionsalgorithmus für $\mathcal{ALE}$.

Offensichtlich arbeitet der Reduktionsalgorithmus nur auf Konzeptbeschreibungen und nicht wie der Algorithmus zur Berechnung des LCS auf Beschreibungsbäumen. Daher eignet sich die Charakterisierung der Subsumtion aus Kapitel 5 nicht für den Nachweis von Korrektheit und Vollständigkeit des Reduktionsalgorithmus. Der folgende Satz liefert eine Charakterisierung der Subsumtion (modulo TBox) in $\mathcal{ALE}$, die unmittelbar auf Konzeptbeschreibungen Bezug nimmt und sich im folgenden als geeignet herausstellen wird. Auf einen ausführlichen Beweis wird hier verzichtet, da er sehr ähnlich zum Beweis von Theorem 5.21 zu führen ist und keine im folgenden relevanten neuen Erkenntnisse liefert.

Satz 8.15

1. *Seien C, D zwei $\mathcal{ALE}$-Konzeptbeschreibungen in $\forall$-Normalform ohne definierte Namen. Dann gilt $C \sqsubseteq D$ genau dann, wenn $C \equiv \bot$ oder $D \equiv \top$ oder*
   - *$\mathrm{prim}(D) \subseteq \mathrm{prim}(C)$,*
   - *$\mathrm{val}_r(C) \sqsubseteq \mathrm{val}_r(D)$ für alle Rollennamen $r \in N_R$, und*
   - *für alle $D_i \in \mathrm{exr}_r(D)$ existiert ein $C_j \in \mathrm{exr}_r(C)$ mit $C_j \sqcap \mathrm{val}_r(C) \sqsubseteq D_i$.*
2. *Sei $\mathcal{T}$ eine $\mathcal{ALE}$-TBox und seien C, D zwei $\mathcal{ALE}$-Konzeptbeschreibungen, die definierte Namen aus $\mathcal{T}$ enthalten dürfen. Desweiteren seien $\{A_1, \ldots, A_n\} = \mathrm{def}(C)$ und $\{B_1, \ldots, B_m\} = \mathrm{def}(D)$. Dann gilt $C \sqsubseteq_{\mathcal{T}} D$ genau dann, wenn $C \equiv_{\mathcal{T}} \bot$ oder $D \equiv_{\mathcal{T}} \top$ oder*

Eingabe: Eine $\mathcal{ALE}$-Konzeptbeschreibung C in $\forall$-Normalform,
eine $\mathcal{ALE}$-TBox $\mathcal{T}$ und
eine $\mathcal{ALE}$-Konzeptbeschreibung F.

Algorithmus: $\mathsf{reduce}(C, \mathcal{T}, F)$

Falls $C \sqcap F \equiv_{\mathcal{T}} \bot$, dann $\widehat{C} := \bot$;
sonst
Sei $\{A_1, \ldots, A_m\} = \mathit{def}(C)$;
Sei $\{Q_1, \ldots, Q_\ell\} = \mathit{prim}(C) \setminus \mathit{prim}(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m))$;
Für jeden Rollennamen $r \in N_R$
Falls $\mathit{val}_r(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)) \sqsubseteq_{\mathcal{T}} \mathit{val}_r(C)$,
dann $D^r := \top$;
sonst $D^r := \mathsf{reduce}(\mathit{val}_r(C), \mathcal{T}, \mathit{val}_r(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)))$;
Sei $\mathcal{D}_r$ eine Teilmenge der Menge

$\mathcal{C}_r := \{\mathsf{reduce}(C_j, \mathcal{T}, \mathit{val}_r(C \sqcap \mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m))) \mid C_j \in \mathit{exr}_r(C)\}$

sodass

1. kein Paar $D_1, D_2 \in \mathcal{D}_r$ existiert mit $D_1 \neq D_2$ und
$\exists r.D_1 \sqcap \forall r.\mathit{val}_r(C) \sqcap F \sqcap A_1 \sqcap \ldots \sqcap A_m \sqsubseteq_{\mathcal{T}} \exists r.D_2$,
2. kein $D \in \mathcal{D}_r$ existiert mit $F \sqcap A_1 \sqcap \ldots \sqcap A_m \sqcap \forall r.\mathit{val}_r(C) \sqsubseteq_{\mathcal{T}} \exists r.D$,
3. für jedes $C_i \in \mathit{exr}_r(C)$ gilt $F \sqcap A_1 \sqcap \ldots \sqcap A_m \sqcap \forall r.\mathit{val}_r(C) \sqsubseteq_{\mathcal{T}} \exists r.C_i$;
oder es existiert ein $D \in \mathcal{D}_r$ mit
$\exists r.D \sqcap \forall r.\mathit{val}(C) \sqcap F \sqcap A_1 \sqcap \ldots \sqcap A_m \sqsubseteq_{\mathcal{T}} \exists r.C_i$, und
4. die Größe $\sum_{D \in \mathcal{D}_r}(|D| + 1)$ der Menge minimal ist unter den Größen aller Teilmengen von $\mathcal{C}_r$, die die Bedingungen (1)–(3) erfüllen.

Definiere $$\widehat{C} := Q_1 \sqcap \ldots \sqcap Q_\ell \sqcap A_1 \sqcap \ldots \sqcap A_m \sqcap \bigsqcap_{r \in N_R} \forall r.D^r \sqcap \bigsqcap_{D \in \mathcal{D}_r} \exists r.D,$$
wobei eine Werterestriktion $\forall r.D^r$ weggelassen wird, falls $D^r = \top$;
Gib $\widehat{C}$ zurück.

Abbildung 8.2: Der Reduktionsalgorithmus für $\mathcal{ALE}$.

- *prim*$(D \sqcap \mathcal{T}(B_1 \sqcap \ldots \sqcap B_m)) \subseteq$ *prim*$(C \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n))$,
- $\mathrm{val}_r(C \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n)) \sqsubseteq_{\mathcal{T}} \mathrm{val}_r(D \sqcap \mathcal{T}(B_1 \sqcap \ldots \sqcap B_m))$ *für alle Rollennamen* $r \in N_R$, *und*
- *für alle* $D_i \in \mathrm{exr}_r(D \sqcap \mathcal{T}(B_1 \sqcap \ldots \sqcap B_m))$ *existiert ein* $C_j \in \mathrm{exr}_r(C \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n))$ *mit* $C_j \sqcap \mathrm{val}_r(C \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n)) \sqsubseteq_{\mathcal{T}} D_i$.

Beweisskizze: Der Beweis der Rückrichtung von Satz 8.15(1) ist trivial. Der Beweis der Hinrichtung von Satz 8.15(1) ergibt sich leicht aus dem Beweis der Hinrichtung zu Theorem 5.21: aus der Annahme, dass eine der Bedingungen nicht erfüllt ist, kann durch Definition einer geeigneten Erweiterung der kanonischen Interpretation zu $\mathcal{G}_C$ ein Widerspruch zur Voraussetzung $C \sqsubseteq D$ abgeleitet werden. Wegen $C \sqsubseteq_{\mathcal{T}} D$ genau dann, wenn $\mathcal{T}(C) \sqsubseteq \mathcal{T}(D)$ folgt Satz 8.15(2) aus Satz 8.15(1). □

Für den Beweis von Vollständigkeit und Korrektheit des Reduktionsalgorithmus benötigen wir neben der geeigneten Charakterisierung der Subsumtion auch noch die im folgenden Lemma formulierte Hilfsaussage. Diese kann leicht mit einer Induktion nach der Tiefe der Subbeschreibung bewiesen werden.

Lemma 8.16 *Seien $\mathcal{T}$ eine $\mathcal{ALE}$-TBox, C eine $\mathcal{ALE}$-Konzeptbeschreibung in $\forall$-Normalform und F eine $\mathcal{ALE}$-Konzeptbeschreibung. Dann gilt für jede Subbeschreibung $\widehat{C}$ von C bzgl. $\mathcal{T}$ und F, dass $C \sqcap F \sqsubseteq_{\mathcal{T}} \widehat{C}$.*

Wir kommen nun zum Nachweis von Korrektheit und Vollständigkeit des Reduktionsalgorithmus aus Abbildung 8.2, d.h. wir zeigen: jede Ausgabe ist eine Reduktion des Eingabekonzeptes C bzgl. $\mathcal{T}$ und F und umgekehrt wird auch jede Reduktion (nichtdeterministisch) berechnet (falls der Kontext F ungleich $\perp$ ist). Man beachte, dass für den Fall $F \equiv_{\mathcal{T}} \perp$ formal auch $\top$ eine Reduktion wäre, die der Algorithmus bei Hinzunahme eines entsprechenden Spezialfalles liefern könnte. Der Einfachheit halber wird dieser Fall aber hier ausgeschlossen.

Lemma 8.17 *Jede Ausgabe $\widehat{C}$ von* $\mathsf{reduce}(C, \mathcal{T}, F)$ *ist eine Reduktion von C bzgl. $\mathcal{T}$ und F. Umgekehrt gilt: Ist $F \not\equiv_{\mathcal{T}} \perp$, so existiert für jede Reduktion E von C bzgl. $\mathcal{T}$ und F eine Ausgabe $\widehat{C}$ von* $\mathsf{reduce}(C, \mathcal{T}, F)$*, die gleich E ist (modulo Kommutativität, Assoziativität und Idempotenz der Konjunktion und der Äquivalenz $C \sqcap \top \equiv C$).*

Beweis der Vollständigkeit: Wir zeigen durch Induktion nach *depth*(E), dass, falls $F \not\equiv_{\mathcal{T}} \perp$, zu jeder Reduktion E von C bzgl. $\mathcal{T}$ und F eine Ausgabe $\widehat{C}$ von $\mathsf{reduce}(C, \mathcal{T}, F)$ existiert mit $\widehat{C} = E$ (modulo der genannten Äquivalenzen).

Angenommen, $E = \perp$. Dann gilt $C \sqcap F \equiv_{\mathcal{T}} \perp$ und die Ausgabe von $\mathsf{reduce}(C, \mathcal{T}, F)$ ist $\perp$, also gleich E.

Angenommen, $E \neq \bot$. Ist $C \sqcap F \equiv_{\mathcal{T}} \bot$, so folgt $F \equiv_{\mathcal{T}} \bot$; andernfalls wäre E wegen $|\bot| = 0$ und $C \sqcap F \equiv_{\mathcal{T}} E \sqcap F$ nicht minimal. Also ist nichts zu zeigen. Sei also $C \sqcap F \not\equiv_{\mathcal{T}} \bot$ und $\widehat{C}$ werde gemäß dem sonst-Zweig berechnet (s. Abbildung 8.2). Zeige:

1. Für alle Ausgaben $\widehat{C}$ gilt $\mathit{def}(E) = \mathit{def}(\widehat{C})$ und $\mathit{prim}(E) = \mathit{prim}(\widehat{C})$.

2. Wenn eine Werterestriktion $\forall r.E^r$ auf Toplevel von E existiert, dann existiert auch eine Werterestriktion $\forall r.C^r$ auf Toplevel von C und eine Ausgabe $\widehat{C^r}$ des rekursiven Aufrufs, die gleich E^r ist (modulo der Äquivalenzen). Wenn keine Werterestriktion $\forall r.E^r$ auf Toplevel von E existiert, so existiert auch eine Ausgabe $\widehat{C}$, die keine Werterestriktion der Form $\forall r.C'$ auf Toplevel enthält.

3. Wenn $\mathit{exr}_r(E) = \{E_1, \ldots, E_\mu\}$, $\mu > 0$, dann ist $\mathit{exr}_r(C) = \{C_1, \ldots, C_\nu\}$, $\nu \geq \mu$ und es existieren Ausgaben $\widehat{C_{i_j}}$, $1 \leq j \leq \mu$, der rekursiven Aufrufe, sodass $\widehat{C_{i_j}}$ gleich E_j ist, $1 \leq j \leq \mu$, und die Menge $\{\widehat{C_{i_j}} \mid 1 \leq j \leq \mu\}$ die Bedingungen (1)–(4) im Algorithmus erfüllt. Wenn $\mathit{exr}_r(E) = \emptyset$, so existiert auch eine Ausgabe $\widehat{C}$ mit $\mathit{exr}_r(\widehat{C}) = \emptyset$.

Aus den Punkten (1)-(3) folgt, dass eine Ausgabe $\widehat{C}$ von $\mathsf{reduce}(C, \mathcal{T}, F)$ existiert, die gleich E ist (modulo der Äquivalenzen).

Ad (1): Aus der Definition von Subbeschreibung folgt $\mathit{def}(C) = \mathit{def}(E)$ und auch $\mathit{def}(\widehat{C}) = \mathit{def}(E)$ für jede Ausgabe $\widehat{C}$ von $\mathsf{reduce}(C, \mathcal{T}, F)$.

Sei weiterhin $\mathit{prim}(E) = \{Q'_1, \ldots, Q'_\ell\}$ und $\mathit{def}(C) = \{A_1, \ldots, A_m\}$. Es gilt $\mathit{prim}(E) \subseteq \mathit{prim}(C) \setminus \mathit{prim}(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m))$, da E andernfalls nicht minimal wäre. Umgekehrt folgt mit Satz 8.15 aus $E \sqcap F \sqsubseteq_{\mathcal{T}} C \sqcap F$ auch $\mathit{prim}(C) \setminus \mathit{prim}(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)) \subseteq \mathit{prim}(E)$. Also folgt $\mathit{prim}(\widehat{C}) = \mathit{prim}(E)$ für jede Ausgabe $\widehat{C}$ von $\mathsf{reduce}(C, \mathcal{T}, F)$.

Ad (2): Sei $r \in N_R$. Mit Satz 8.15 folgt aus $C \sqcap F \equiv_{\mathcal{T}} E \sqcap F$

$$\mathit{val}_r(C \sqcap \mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)) \equiv_{\mathcal{T}} \mathit{val}_r(E \sqcap \mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)).$$

Hieraus folgt wegen $\mathit{val}_r(D_1 \sqcap D_2) \equiv \mathit{val}_r(D_1) \sqcap \mathit{val}_r(D_2)$ auch

$$\mathit{val}_r(C) \sqcap \mathit{val}_r(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)) \equiv_{\mathcal{T}} \mathit{val}_r(E) \sqcap \mathit{val}_r(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)).$$

Ist $\mathit{val}_r(E) = \top$, so folgt $\mathit{val}_r(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)) \sqsubseteq_{\mathcal{T}} \mathit{val}_r(C)$ und für jede Ausgabe $\widehat{C}$ gilt, dass keine Werterestriktion der Form $\forall r.D^r$ auf Toplevel von $\widehat{C}$ existiert (s. Abbildung 8.2).

Ist $\mathit{val}_r(E) \neq \top$, d.h. es existiert eine eindeutige Werterestriktion der Form $\forall r.E^r$ auf Toplevel von E, so existiert nach Definition einer Subbeschreibung auch eine eindeutige Werterestriktion der Form $\forall r.C^r$ auf Toplevel von C mit: E^r ist Subbeschreibung von C^r bzgl. $\mathcal{T}$ und $val_r(F \sqcap \mathcal{T}(F_1 \sqcap \ldots \sqcap F_n \sqcap A_1 \sqcap \ldots \sqcap A_m))$, wobei $\{F_1, \ldots, F_n\} = def(F)$. Sei $F^r := val_r(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m))$. Aus der Minimalität von E folgt, dass auch E^r eine minimale Subbeschreibung von C^r ist mit $C^r \sqcap F^r \equiv_{\mathcal{T}} E^r \sqcap F^r$. Also ist E^r eine Reduktion von C^r bzgl. $\mathcal{T}$ und F^r.

Um die Induktionsvoraussetzung anwenden zu können, müssen wir noch zeigen, dass $F^r \not\equiv \bot$. Angenommen, $F^r \equiv \bot$. Dann wäre E wegen $|\forall r.E^r| \geq 1$ keine minimale Subbeschreibung von C mit $C \sqcap F \equiv_{\mathcal{T}} E \sqcap F$, da durch eliminieren von $\forall r.E^r$ auf Toplevel von E eine Subbeschreibung von C entstehen würde, die auch noch die Äquivalenz erfüllen würde, aber kleiner als E wäre. Also ist $F^r \not\equiv_{\mathcal{T}} \bot$ und per Induktion folgt, dass eine Ausgabe $\widehat{C^r}$ von $\mathsf{reduce}(C^r, \mathcal{T}, F^r)$ existiert, die gleich E^r ist.

Ad (3): Sei $r \in N_R$, $def(F) = \{F_1, \ldots, F_n\}$ und $F^r := val_r(C \sqcap \mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m))$. Ist $exr_r(E) = \emptyset$, so folgt aus $E \sqcap F \sqsubseteq_{\mathcal{T}} C \sqcap F$ mit Satz 8.15 für alle $C_i \in exr_r(C)$, dass ein $D' \in exr_r(F \sqcap \mathcal{T}(F_1 \sqcap \ldots \sqcap F_n \sqcap A_1 \sqcap \ldots \sqcap A_m))$ existiert mit $D' \sqcap val_r(E \sqcap F \sqcap \mathcal{T}(F_1 \sqcap \ldots \sqcap F_n \sqcap A_1 \ldots \sqcap A_m)) \sqsubseteq_{\mathcal{T}} C_i$. Man sieht nun leicht, dass die leere Menge die Bedingungen (1)–(4) an die Menge $\mathcal{D}^r$ im Reduktionsalgorithmus erfüllt, d.h. es gibt eine Ausgabe $\widehat{C}$ mit $exr_r(\widehat{C}) = \emptyset$.

Ist $exr_r(E) = \{E_1, \ldots, E_\mu\}$, $\mu > 0$, so ist nach Definition einer Subbeschreibung $exr_r(C) = \{C_1, \ldots, C_\nu\}$ mit $\nu \geq \mu$. Außerdem ist $F^r \not\equiv \bot$, da sonst $C \sqcap F \equiv_{\mathcal{T}} \bot$ im Widerspruch zur Annahme $C \sqcap F \not\equiv_{\mathcal{T}} \bot$.

Zu zeigen ist nun: Für alle $1 \leq i \leq \mu$ existiert ein $C_{j_i} \in exr_r(C)$, sodass E_i Reduktion von C_{j_i} bzgl. $\mathcal{T}$ und F^r ist, d.h. $E_i \sqcap F^r \equiv_{\mathcal{T}} C_{j_i} \sqcap F^r$ und E_i ist minimale Subbeschreibung von C_{j_i} mit dieser Eigenschaft.

Sei dazu $\mathcal{C}_i := \{C' \in exr_r(C) \mid E_i \text{ ist Subbeschreibung von } C'\}$. Da E Subbeschreibung von C bzgl. $\mathcal{T}$ ist, ist $\mathcal{C}_i \neq \emptyset$. Mit Lemma 8.16 folgt $C' \sqcap F^r \sqsubseteq E_i \sqcap F^r$ für alle $C' \in \mathcal{C}_i$. Es gilt auch die Umkehrung $E_i \sqcap F^r \sqsubseteq_{\mathcal{T}} C' \sqcap F^r$ für alle $C' \in \mathcal{C}_i$. Denn angenommen, es gibt ein $C' \in \mathcal{C}_i$ mit $E_i \sqcap F^r \not\sqsubseteq_{\mathcal{T}} C' \sqcap F^r$. Wegen $E \sqcap F \sqsubseteq_{\mathcal{T}} C \sqcap F$, $E \sqcap F \not\equiv_{\mathcal{T}} \bot$ und $C \sqcap F \not\equiv_{\mathcal{T}} \top$ folgt mit Satz 8.15, dass ein $D' \in exr_r(E \sqcap F \sqcap \mathcal{T}(F_1 \sqcap \ldots \sqcap F_n \sqcap A_1 \sqcap \ldots \sqcap A_m))$ existiert mit

$$D' \sqcap val_r(E \sqcap F \sqcap \mathcal{T}(F_1 \sqcap \ldots \sqcap F_n \sqcap A_1 \sqcap \ldots \sqcap A_m)) \sqsubseteq_{\mathcal{T}} C'.$$

Mit (2) wissen wir bereits, dass

$$val_r(E \sqcap F \sqcap \mathcal{T}(F_1 \sqcap \ldots \sqcap F_n \sqcap A_1 \sqcap \ldots \sqcap A_m)) \equiv_{\mathcal{T}} F^r.$$

Damit folgt

$$(*)\ \ D' \sqcap F^r \sqsubseteq_{\mathcal{T}} C' \sqcap F^r \sqsubseteq_{\mathcal{T}} E_i \sqcap F^r.$$

Offenbar gilt nun $E_i \neq D'$. Dies liefert aber einen Widerspruch zur Minimalität von E: eliminiert man $\exists r.E_i$ auf Toplevel von E, so erhält man wegen $(*)$ eine zu

E äquivalente, aber kleinere Subbeschreibung, d.h. E wäre keine Reduktion von C bzgl. $\mathcal{T}$ und F.
Es folgt also, dass ein $C_{j_i} \in exr_r(C)$ existiert mit $C_{j_i} \sqcap F^r \equiv_\mathcal{T} E_i \sqcap F^r$ und E_i ist Subbeschreibung von C_{j_i} bzgl. $\mathcal{T}$ und F^r. E_i ist außerdem (bzgl. $|\cdot|$) minimal mit dieser Eigenschaft: Angenommen, E_i wäre nicht minimal mit dieser Eigenschaft. Dann existiert eine Subbeschreibung D' von C_{j_i} bzgl. $\mathcal{T}$ und F^r mit $D' \sqcap F^r \equiv_\mathcal{T} C_{j_i} \sqcap F^r$ und $|D'| < |E_i|$. Ersetzt man nun E_i durch D', so würde man eine Subbeschreibung E' von C bzgl. $\mathcal{T}$ und F erhalten mit $C \sqcap F \equiv_\mathcal{T} E' \sqcap F$ und $|E'| < |E|$ im Widerspruch zur Annahme, dass E minimal mit dieser Eigenschaft ist. Also ist E_i eine Reduktion von C_{j_i} bzgl. $\mathcal{T}$ und F^r. Oben wurde bereits gezeigt, dass $F^r \not\equiv_\mathcal{T} \bot$, sodass nun per Induktion folgt, dass eine Ausgabe $\widehat{C_{j_i}}$ von $\mathsf{reduce}(C_{j_i}, \mathcal{T}, F^r)$ existiert, die gleich E_i ist.
Es bleibt zu zeigen, dass die so erhaltene Menge $\{C_{j_1}, \ldots, C_{j_\mu}\}$ die Bedingungen (1)–(4) des Algorithmus erfüllt.

- Bedingungen (1) und (2) sind erfüllt, da E sonst nicht minimal wäre: das Eliminieren der redundanten Existenzrestriktion würde eine äquivalente und kleinere Subbeschreibung liefern.
- Bedingung (3) ist erfüllt, da andernfalls mit Satz 8.15(2) folgen würde, dass $E \sqcap F \not\equiv_\mathcal{T} C \sqcap F$.
- Bedingung (4) erfüllt, weil andernfalls E wiederum nicht minimal wäre.

Dies schließt den Beweis der Vollständigkeit ab.

Beweis der Korrektheit: Zu zeigen: Jede Ausgabe $\widehat{C}$ von $\mathsf{reduce}(C, \mathcal{T}, F)$ ist eine Reduktion von C bzgl. $\mathcal{T}$ und F.
Nach Konstruktion ist $\widehat{C}$ eine Subbeschreibung von C bzgl. $\mathcal{T}$ und F. Es bleibt also zu zeigen, dass

1. $\widehat{C} \sqcap F \equiv_\mathcal{T} C \sqcap F$ und
2. $\widehat{C}$ eine minimale Subbeschreibung ist, die 1. erfüllt.

Ad (1)*:* Angenommen, $\widehat{C} = \bot$. Nach Konstruktion ist dann $C \sqcap F \equiv_\mathcal{T} \bot$, also $\widehat{C} \sqcap F \equiv_\mathcal{T} C \sqcap F$.
Sei also $\widehat{C} \neq \bot$. Zeige

a) $\widehat{C} \sqcap F \sqsubseteq_\mathcal{T} C \sqcap F$ und

b) $\widehat{C} \sqcap F \sqsupseteq_\mathcal{T} C \sqcap F$.

Die Subsumtionsbeziehung *b)* folgt mit Lemma 8.16. Die Subsumtionsbeziehung *a)* folgt wegen $\mathit{def}(\widehat{C}) = \mathit{def}(C)$ unmittelbar aus den folgenden drei Punkten:

(*i*) für alle (negierten) primitiven Konzeptnamen $Q \in prim(C)$ gilt $\widehat{C} \sqcap F \sqsubseteq_{\mathcal{T}} Q$;

(*ii*) für alle Rollennamen $r \in N_R$ gilt $\widehat{C} \sqcap F \sqsubseteq_{\mathcal{T}} \forall r.val_r(C)$;

(*iii*) für alle Rollennamen $r \in N_R$ und alle $C_i \in exr_r(C)$ gilt $\widehat{C} \sqcap F \sqsubseteq_{\mathcal{T}} \exists r.C_i$.

Wir zeigen die Punkte (*i*)–(*iii*) durch Induktion nach $depth(C)$.
Ad (i): Sei $Q \in prim(C)$. Falls $Q \in prim(\widehat{C})$, dann $\widehat{C} \sqcap F \sqsubseteq_{\mathcal{T}} Q$. Sonst wurde Q vom Toplevel von C entfernt. Aus der Definition von $\widehat{C}$ folgt dann aber, dass Q auf Toplevel von $\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)$ auftritt. Da $def(C) = def(\widehat{C})$ folgt also $\widehat{C} \sqcap F \sqsubseteq_{\mathcal{T}} Q$.
Ad (ii): Sei $r \in N_R$ und $val_r(C) \neq \top$. Falls keine Werterestriktion $\forall r.D^r$ auf Toplevel von $\widehat{C}$ existiert, dann folgt aus der Definition von $\widehat{C}$, dass $val(\mathcal{T}(F, \sqcap A_1 \sqcap \ldots \sqcap A_m)) \sqsubseteq_{\mathcal{T}} val_r(C)$. Falls eine Werterestriktion der Form $\forall r.D^r$ auf Toplevel von $\widehat{C}$ existiert, dann ist nach Konstruktion $D^r = \mathsf{reduce}(val_r(C), \mathcal{T}, val_r(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m))$. Per Induktion folgt

$$D^r \sqcap val_r(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)) \sqsubseteq_{\mathcal{T}} val_r(C) \sqcap val_r(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)).$$

Dies impliziert

$$\forall r.D^r \sqcap \forall r.val_r(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)) \sqsubseteq_{\mathcal{T}} \forall r.val_r(C)$$

und insbesondere $\widehat{C} \sqcap F \sqsubseteq_{\mathcal{T}} \forall r.val_r(C)$.
Ad (iii): Sei $r \in N_R$ und $C_i \in exr_r(C)$. Aus der dritten Bedingung an die Menge $\mathcal{D}_r$ im Reduktionsalgorithmus folgt, dass entweder $F \sqcap A_1 \sqcap \ldots \sqcap A_m \sqcap \forall r.val_r(C) \sqsubseteq_{\mathcal{T}} \exists r.C_i$ oder ein $D \in exr_r(\widehat{C})$ existiert mit $\exists r.D \sqcap \forall r.val_r(C \sqcap \mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)) \sqsubseteq_{\mathcal{T}} \exists r.C_i$. In beiden Fällen folgt $\widehat{C} \sqcap F \sqsubseteq_{\mathcal{T}} \exists r.C_i$. Dies schließt den Beweis von (1) ab, d.h. wir haben gezeigt, dass $\widehat{C} \sqcap F \equiv_{\mathcal{T}} C \sqcap F$.

Ad (2): Ist $F \equiv_{\mathcal{T}} \bot$, so ist $C \sqcap F \equiv_{\mathcal{T}} \bot$ und damit $\bot$ die einzige Ausgabe von $\mathsf{reduce}(C, \mathcal{T}, F)$. Offensichtlich ist $\bot$ eine minimale Subbeschreibung von C mit $C \sqcap F \equiv_{\mathcal{T}} \bot$.
Ist $F \not\equiv_{\mathcal{T}} \bot$, so folgt (2) leicht unter Rückgriff auf den Beweis der Vollständigkeit: Durch Induktion nach $depth(C)$ folgt, dass jede Ausgabe von $\mathsf{reduce}(val_r(C), \mathcal{T}, val_r(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_n)))$ eine Reduktion von $val_r(C)$ ist. Insbesondere haben alle möglichen Ausgaben die gleiche, minimale Größe. Bedingung (4) an die Menge $\mathcal{D}_r$ im Reduktionsalgorithmus sichert zu, dass alle Mengen $\mathcal{D}_r$, die die Bedingungen (1)–(3) erfüllen, die gleiche, minimale Größe haben. Insgesamt hat also jede Ausgabe von $\mathsf{reduce}(C, \mathcal{T}, F)$ die gleiche Größe. Für $F \not\equiv_{\mathcal{T}} \bot$ wissen wir aus dem Beweis der Vollständigkeit, dass zu jeder Reduktion eine Ausgabe $\widehat{C}$ von $\mathsf{reduce}(C, \mathcal{T}, F)$ existiert mit $\widehat{C} = E$. Damit haben alle Ausgaben von $\mathsf{reduce}(C, \mathcal{T}, F)$ die minimale Größe $|E|$.
Dies schließt den Beweis von Lemma 8.17 ab. □

Man beachte, dass durch die Definition der Größe mit $|\bot| = 0$ erzwungen wird, dass immer $\bot$ eingesetzt wird, wenn $C \sqcap F \equiv_{\mathcal{T}} \bot$. Das folgende Beispiel zeigt, dass diese Eigenschaft entscheidend für die Vollständigkeit des Reduktionsalgorithmus und damit auch des Rewriting-Algorithmus ist.

Beispiel 8.18 *Definiert man die Größe von C als die Anzahl der Vorkommen von Konzept- und Rollennamen plus die Anzahl der Vorkommen von $\top$ und $\bot$ in C, so ist der Reduktionsalgorithmus aus Abbildung 8.2* nicht *vollständig im Sinne von Lemma 8.17. Betrachte beispielsweise die Instanz $C, \mathcal{T}$ mit*

$$\begin{aligned} C &= Q \sqcap \forall r.(P \sqcap \neg P) \\ \mathcal{T} &= \{A \doteq Q \sqcap \forall r.\neg P\}. \end{aligned}$$

Neben C selbst ist die einzige Extension von C bzgl. $\mathcal{T}$ gegeben durch $C^* = A \sqcap Q \sqcap \forall r.(P \sqcap \neg P)$. Nun liefert $\mathsf{reduce}(C, \mathcal{T}, \top)$ *als eindeutige Ausgabe $Q \sqcap \forall r.\bot$ und* $\mathsf{reduce}(C^*, \mathcal{T}, \top)$ *liefert $A \sqcap \forall r.\bot$, beide mit Größe 3 bzgl. des modifizierten Größenmaßes. Allerdings ist bzgl. des modifizierten Größenmaßes auch $A \sqcap \forall r.P$ eine Reduktion von C^* bzgl. $\mathcal{T}$. Diese wird aber offensichtlich nicht berechnet. Damit wäre nicht nur der Reduktionsalgorithmus nicht vollständig, sondern auch der Rewriting-Algorithmus würde bzgl. des modifizierten Größenmaßes nicht alle minimalen Rewritings berechnen. Folglich wäre für dieses modifizierte Größenmaß*

- *die Aussage in Theorem 8.11(2) abzuschwächen: die Menge aller berechneten Rewritings enthält ein minimales Rewriting; oder*
- *der Reduktionsalgorithmus geeignet zu modifizieren: Im Falle $C \sqcap F \equiv_{\mathcal{T}} \bot$ ist nicht nur $\bot$ zurückzugeben, sondern auch alle Konzeptbeschreibungen der Größe $|\bot|$, die in Konjunktion mit F Inkonsistenz liefern und der Definition einer Subbeschreibung genügen.*

Auch die Berücksichtigung des Kontextes bei der Definition einer Subbeschreibung ist notwendig für die Vollständigkeit des Reduktionsalgorithmus. Die Definition von Subbeschreibungen von $\mathcal{ALE}$-Konzeptbeschreibungen *ohne* definierte Namen in [BK00a] schränkt das Ersetzen von C durch $\bot$ nicht auf (in einem Kontext) inkonsistente Konzeptbeschreibungen ein. Würden wir Punkt (2) in Definition 8.12 ändern in

2.' $\widehat{C} = \bot$;

so würde zum einen Lemma 8.16 nicht mehr gelten und zum anderen wäre der Reduktionsalgorithmus nicht mehr vollständig, wie das folgende Beispiel zeigt.

Beispiel 8.19 *Betrachte*

$$\begin{aligned}
\mathcal{T} &= \{A_1 \doteq \forall r.P, A_2 \doteq \forall r.\neg P\},\\
C &= \exists r.(A_2 \sqcap \forall r.P) \sqcap \exists r.(A_1 \sqcap \forall r.Q),\\
E &= \exists r.(A_1 \sqcap \forall r.\bot), \text{ und}\\
E' &= \exists r.(A_2 \sqcap \forall r.\bot).
\end{aligned}$$

Man sieht leicht, dass $C \equiv_\mathcal{T} E$ und $C \equiv_\mathcal{T} E'$. Desweiteren sind E und E' bzgl. der modifizierten Definition Subbeschreibungen von C bzgl. $\mathcal{T}$ und $\top$. Insbesondere sind sie minimale und zu C modulo $\mathcal{T}$ äquivalente Subbeschreibungen, also Reduktionen von C bzgl. $\mathcal{T}$ und $\top$. Der Reduktionsalgorithmus liefert aber nur E' als Ausgabe. Ursache hierfür ist, dass $\exists r.(A_2 \sqcap \forall r.P) \sqsubseteq_\mathcal{T} \exists r.(A_1 \sqcap \forall r.Q)$ und damit $\mathcal{D}^r = \{A_2 \sqcap \forall r.\bot\}$ die eindeutige Teilmenge von $\mathcal{C}^r = \{A_2 \sqcap \forall r.\bot, A_1 \sqcap \forall r.Q\}$ ist, die die Bedingungen (1)–(4) des Algorithmus erfüllt. Dies wiederum liegt daran, dass die Konzeptbeschreibung Q im Rekursionsschritt bzgl. des Kontextes P betrachtet und daher nicht *durch $\bot$ ersetzt wird.*
Der Reduktionsalgorithmus aus Abbildung 8.2 wäre also bzgl. der modifizierten Definition einer Subbeschreibung nicht vollständig. Es sei noch angemerkt, dass dieses Beispiel kein *Gegenbeispiel zur Vollständigkeit des verbesserten Rewritingalgorithmus liefert, da C wegen $C \not\equiv_\mathcal{T} \exists r.P \sqcap \exists r.Q$ nicht als Extension bzgl. $\mathcal{T}$ einer $\mathcal{ALE}$-Konzeptbeschreibung auftrete kann.*

8.3.2 Zur Korrektheit des Rewriting-Algorithmus für $\mathcal{ALE}$

In diesem Unterabschnitt wird Theorem 8.11 bewiesen. Die erste Behauptung des Theorems folgt offensichtlich direkt aus den Definitionen von Extension und Reduktion. Es bleibt also Theorem 8.11(2) zu zeigen. Der Beweis erfolgt in zwei Schritten:

1. Zeige, dass zu jedem minimalen Rewriting E von C bzgl. $\mathcal{T}$ eine Extension C^E existiert, sodass E eine Subbeschreibung von C^E bzgl. $\mathcal{T}$ und $\top$ ist.

2. Zeige, dass zu dieser Extension eine Reduktion existiert, die gleich E ist, also vom Rewriting-Algorithmus als Rewriting von C bzgl. $\mathcal{T}$ berechnet wird.

Zum 1. Schritt: Informell erhält man die durch ein minimales Rewriting E von C bzgl. $\mathcal{T}$ induzierte Extension C^E wie folgt:

1. erweitere die Konjunktion auf Toplevel von C um alle definierten Namen, die auf Toplevel von E auftreten;

2. falls eine Werterestriktion der Form $\forall r.E'$ auf Toplevel von E existiert, so existiert auch eine Werterestriktion der Form $\forall r.C'$ auf Toplevel von C (andernfalls wäre E nicht äquivalent zu C modulo $\mathcal{T}$, da C ja keine definierten Namen enthält; s. Satz 8.15(2)); ersetze C' durch die rekursiv bestimmte, durch E' induzierte Extension $C^{E'}$ von C';

3. für jede Existenzrestriktion $\exists r.E_i$ auf Toplevel von E existiert eine Existenzrestriktion $\exists r.C_j$ auf Toplevel von C, sodass $C_j \sqcap \mathit{val}_r(C) \equiv_{\mathcal{T}} E_i \sqcap \mathit{val}_r(C)$ (andernfalls wäre E wiederum nicht äquivalent zu C modulo $\mathcal{T}$; cf Lemma 8.22); ersetze C_j durch die rekursiv bestimmte, durch E_i induzierte Extension $C_j^{E_i}$ von C_j.

Wie im Falle des Reduktionsalgorithmus ist auch in der formalen Definition der induzierten Extension der Kontext zu berücksichtigen, indem Konzeptbeschreibungen rekursiv verarbeitet werden. Um diesen Kontext wieder formal als Parameter zu fassen, erweitern wir den Begriff der Extension:

Definition 8.20 (Extension bzgl. $\mathcal{T}$ und F) *Sei C eine $\mathcal{ALE}$-Konzeptbeschreibung, $\mathcal{T}$ eine $\mathcal{ALE}$-TBox und F eine $\mathcal{ALE}$-Konzeptbeschreibung. Die $\mathcal{ALE}$-Konzeptbeschreibung C^* ist eine* Extension von C bzgl. $\mathcal{T}$ und F *genau dann, wenn $C \sqcap F \equiv_{\mathcal{T}} C^* \sqcap F$ und C^* aus C durch konjunktive Anbindung definierter Namen aus $\mathcal{T}$ an beliebigen Stellen in C erhalten werden kann.*

Für die formale Definition der durch E induzierten Extension benötigen wir noch den Begriff der *Reduziertheit bzgl. $\mathcal{T}$ und F*:

Definition 8.21 (reduziert bzgl. $\mathcal{T}$ und F) *Eine $\mathcal{ALE}$-Konzeptbeschreibung C heißt* reduziert bzgl. $\mathcal{T}$ und F *genau dann, wenn C eine Reduktion bzgl. $\mathcal{T}$ und F von sich selber ist.*

Abbildung 8.3 zeigt die rekursive Definition einer *durch E induzierten Extension C^E von C bzgl. $\mathcal{T}$ und F*. Diese Definition macht Sinn, da (i) für den Rekursionsschritt jeweils die Voraussetzungen erfüllt sind und (ii) stets eine geeignete Permutation der Existenzrestriktionen auf Toplevel von C existiert. Das folgende Lemma formalisiert diese Aussagen.

Lemma 8.22 *Unter Verwendung der Bezeichner aus der rekursiven Definition aus Abbildung 8.3 gilt:*

1. *D^r ist wohldefiniert, d.h. die Voraussetzungen der Definition sind für die rekursive Definition von D^r erfüllt.*
2. *Es existiert eine Permutation $\{j_1, \dots, j_m\}$ von $\{1, \dots, m\}$ sodass $C_{i_j}^{r,*}$, $1 \leq i \leq \ell$, wohldefiniert ist.*

Beweis: *Ad (1):* Zu zeigen:

(i) $\mathit{val}_r(C)$ ist in $\forall$-Normalform und enthält keine definierten Namen,

(ii) $\mathit{val}_r(F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n))$ enthält keine definierten Namen,

Gegeben: Eine $\mathcal{ALE}$-TBox $\mathcal{T}$ und $\mathcal{ALE}$-Konzeptbeschreibungen C, F, E, wobei

- C ist in $\forall$-Normalform und enthält keine definierten Konzeptnamen,
- F enthält keine definierten Konzeptnamen,
- E ist reduziert bzgl. $\mathcal{T}$ und F und
- $C \sqcap F \equiv_{\mathcal{T}} E \sqcap F$.

Rekursive Definition
der durch E induzierten Extension C^E von C bzgl. $\mathcal{T}$ und F:

Falls $E \sqcap F \equiv_{\mathcal{T}} \bot$, dann $C^E := C$;
sonst
  Sei $\{Q_1, \ldots, Q_k\} := \mathit{prim}(C)$;
  Sei $\{A_1, \ldots, A_n\} := \mathit{def}(E)$;
  Für jeden Rollennamen $r \in N_R$
    Sei D^r die rekursiv definierte, durch $\mathit{val}_r(E)$ induzierte
      Extension von $\mathit{val}_r(C)$ bzgl. $\mathcal{T}$ und $\mathit{val}_r(F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n)))$;
    Sei $\mathit{exr}_r(C) = \{C_1^r, \ldots, C_m^r\}$ und $\mathit{exr}_r(E) = \{E_1, \ldots, E_\ell\}$;
    Sei $\{j_1, \ldots, j_m\}$ eine Permutation von $\{1, \ldots, m\}$
      sodass, für alle $1 \leq i \leq \ell$,
      $C_{j_i}^r \sqcap \mathit{val}_r(C \sqcap F) \equiv_{\mathcal{T}} E_i \sqcap \mathit{val}_r(E \sqcap F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n))$;
    Für $1 \leq i \leq \ell$ sei $C_{j_i}^{r,*}$ die durch E_i induzierte, rekursiv bestimmte
      Extension von $C_{j_i}^r$ bzgl. $\mathcal{T}$ und $\mathit{val}_r(C \sqcap F)$;
  Definiere nun C^E durch

$$\begin{aligned} C^E \quad := \quad & Q_1 \sqcap \ldots \sqcap Q_k \sqcap A_1 \sqcap \ldots \sqcap A_n \sqcap \\ & \bigsqcap_{r \in N_R} \forall r.D^r \sqcap \bigsqcap_{1 \leq i \leq \ell} \exists r.C_{j_i}^{r,*} \sqcap \bigsqcap_{\ell+1 \leq i \leq m} \exists r.C_{j_i}^r, \end{aligned}$$

  wobei die Werterestriktion $\forall r.D^r$ weggelassen wird, falls keine
  Werterestriktion der Form $\forall r.C'$ auf Toplevel von C existiert.

Abbildung 8.3: Die rekursive Definition der durch E induzierten Extension C^E von C bzgl. $\mathcal{T}$ und F.

(iii) $val_r(E)$ ist reduziert bzgl. $\mathcal{T}$ und $val_r(F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n))$ und

(iv) $val_r(C) \sqcap val_r(F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n)) \equiv_{\mathcal{T}} val_r(E) \sqcap val_r(F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n))$.

Ad (i): Falls für $r \in N_R$ keine Werterestriktion der Form $\forall r.C'$ auf Toplevel von C existiert, so ist nach Definition $val_r(C) = \top$ und $val_r(C)$ ist in $\forall$-Normalform. Sonst existiert eine eindeutige Werterestriktion der Form $\forall r.C'$ auf Toplevel von C und C' ist in $\forall$-Normalform; andernfalls wäre C nicht in $\forall$-Normalform. Da C keine definierten Namen enthält, enthält auch $val_r(C)$ keine definierten Namen.

Ad (ii): Da F keine definierten Namen enthält und nach Konstruktion eine aufgefaltete Konzeptbeschreibung keine definierten Namen mehr enthält, treten auch keine definierten Namen in $val_r(F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n))$ auf.

Ad (iii): $val_r(E)$ ist reduziert bzgl. $\mathcal{T}$ und $val_r(F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n))$, da andernfalls E nicht reduziert bzgl. $\mathcal{T}$ und F wäre.

Ad (iv): Aus $C \sqcap F \equiv_{\mathcal{T}} E \sqcap F$ folgt mit Satz 8.15(2)

$$(*)\quad val_r(C \sqcap F) \equiv_{\mathcal{T}} val_r(E \sqcap F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n)),$$

wobei $\{A_1, \ldots, A_n\} = def(E)$. Wegen $val_r(C_1 \sqcap C_2) \equiv_{\mathcal{T}} val_r(C_1) \sqcap val_r(C_2)$ und $C_1 \equiv_{\mathcal{T}} C_2 \sqcap C_3 \implies C_1 \sqcap C_3 \equiv_{\mathcal{T}} C_2 \sqcap C_3$ folgt daraus

$$val_r(C) \sqcap val_r(F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n)) \equiv_{\mathcal{T}} val_r(E) \sqcap val_r(F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n)).$$

Also sind die Voraussetzungen der rekursiven Definition für $val_r(C)$, $val_r(F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n))$ und $val_r(E)$ erfüllt und D^r ist nach Induktion wohldefiniert.

Ad (2): Für $r \in N_R$ und $exr_r(E) = \{E_1, \ldots, E_\ell\}$ definieren wir die Permutation von $\{1, \ldots, m\}$ wie folgt: Für $1 \leq i \leq \ell$ folgt mit Satz 8.15 aus $C \sqcap F \sqsubseteq_{\mathcal{T}} E \sqcap F$, dass ein $C_j \in exr_r(C \sqcap F)$ existiert mit $C_j \sqcap val_r(C \sqcap F) \sqsubseteq_{\mathcal{T}} E_i$. Es gilt sogar $C_j \in exr_r(C)$. Andernfalls wäre $F \sqcap \forall r.val_r(C \sqcap F) \sqsubseteq_{\mathcal{T}} \exists r.E_i$. Mit (iv) aus dem ersten Teil des Beweises würde hieraus aber $F \sqcap \forall r.val_r(E \sqcap F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n)) \sqsubseteq_{\mathcal{T}} \exists r.E_i$ folgen. Damit liefert das Entfernen von $\exists r.E_i$ auf Toplevel von E aber eine Subbeschreibung E' von E, die kleiner als E ist und $E' \sqcap F \equiv_{\mathcal{T}} E \sqcap F$ erfüllt im Widerspruch dazu, dass E bzgl. $\mathcal{T}$ und F reduziert ist.

Also existiert ein $C_j \in exr_r(C)$ mit

$$(**)\quad C_j \sqcap val_r(C \sqcap F) \sqsubseteq_{\mathcal{T}} E_i.$$

Definiere $j_i := j$. Da $i \in \{1, \ldots, \ell\}$ beliebig gewählt war, sollten nun die ersten ℓ Elemente der gesuchten Permutation festgelegt sein. Dazu ist aber noch zu zeigen, dass für $i, k \in \{1, \ldots, \ell\}$ mit $i \neq k$ auch $j_i \neq j_k$ ist. Wir zeigen dazu

$$(+)\quad C_{j_i} \sqcap val_r(C \sqcap F) \equiv_{\mathcal{T}} E_i \sqcap val_r(E \sqcap F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n)).$$

Aus $(*)$ und $(**)$ folgt die eine Richtung dieser Äquivalenz:

$$(++)\ C_{j_i} \sqcap \mathit{val}_r(C \sqcap F) \sqsubseteq_{\mathcal{T}} E_i \sqcap \mathit{val}_r(E \sqcap F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n)).$$

Angenommen, $C_{j_i} \sqcap \mathit{val}_r(C \sqcap F) \sqsupseteq_{\mathcal{T}} E_i \sqcap \mathit{val}_r(E \sqcap F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n))$ gilt nicht. Aus $C \sqcap F \sqsupseteq_{\mathcal{T}} E \sqcap F$ folgt mit Satz 8.15, dass es ein $D \in \mathit{exr}_r(E \sqcap F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n))$ gibt mit $D \sqcap \mathit{val}_r(E \sqcap F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n)) \sqsubseteq_{\mathcal{T}} C_{j_i}$. Es gilt sogar $D \in \mathit{exr}_r(E)$. Andernfalls gilt

$$A_1 \sqcap \ldots \sqcap A_n \sqcap F \sqcap \forall r.\mathit{val}_r(E \sqcap F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n)) \sqsubseteq_{\mathcal{T}} \exists r.C_{j_i}.$$

Mit $(*)$ würde folgen

$$A_1 \sqcap \ldots \sqcap A_n \sqcap F \sqcap \forall r.\mathit{val}_r(E \sqcap F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n)) \sqsubseteq_{\mathcal{T}} \exists r.C_{j_i} \sqcap \forall r.\mathit{val}_r(C \sqcap F),$$

woraus mit $(++)$ folgen würde

$$A_1 \sqcap \ldots \sqcap A_n \sqcap F \sqcap \forall r.\mathit{val}_r(E \sqcap F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n)) \sqsubseteq_{\mathcal{T}} \exists r.E_i.$$

Wie oben ergibt sich hieraus aber ein Widerspruch dazu, dass E reduziert bzgl. $\mathcal{T}$ und F ist.

Es existiert also ein $D \in \mathit{exr}_r(E)$ mit $D \sqcap \mathit{val}_r(E \sqcap F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n)) \sqsubseteq_{\mathcal{T}} C_{j_i}$. Es ist $D = E_i$, da sonst wiederum wie oben ein Widerspruch zur Reduziertheit von E bzgl. $\mathcal{T}$ und F folgen würde. Also folgt Behauptung $(+)$. Mit $(+)$ wiederum folgt aus der Annahme $j_i = j_k$ für $i \neq k$ wieder wie oben ein Widerspruch zur Reduziertheit von E bzgl. $\mathcal{T}$ und F. Es gilt also $j_i \neq j_k$ für alle $i \neq k$ aus $\{1, \ldots, \ell\}$.

Sei nun $\{j_{\ell+1}, \ldots, j_m\}$ eine beliebige Permutation von $\{1, \ldots, m\} \setminus \{j_1, \ldots, j_\ell\}$. Dann ist $\{j_1, \ldots, j_m\}$ eine Permutation von $\{1, \ldots, m\}$. Für die Wohldefiniertheit von $C_{j_i}^{r,*}$ bleibt für $1 \leq i \leq \ell$ zu zeigen:

(i) C_{j_i} ist in $\forall$-Normalform und enthält keine definierten Namen,

(ii) $\mathit{val}_r(C \sqcap F)$ enthält keine definierten Namen,

(iii) E_i ist reduziert bzgl. $\mathcal{T}$ und $\mathit{val}_r(C \sqcap F)$ und

(iv) $C_{j_i} \sqcap \mathit{val}_r(F \sqcap C) \equiv_{\mathcal{T}} E_i \sqcap \mathit{val}_r(E \sqcap F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n))$.

Ad (i): Da C in $\forall$-Normalform ist und keine definierten Namen enthält, ist auch C_{j_i} in $\forall$-Normalform und enthält keine definierten Namen.

Ad (ii): Da C und F keine definierten Namen enthalten, treten auch keine definierten Namen in $\mathit{val}_r(C \sqcap F)$ auf.

Ad (iii): E_i ist reduziert bzgl. $\mathcal{T}$ und $\mathit{val}_r(E \sqcap F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n))$, da andernfalls E nicht reduziert bzgl. $\mathcal{T}$ und F wäre. Mit $(*)$ folgt, dass E_i auch reduziert bzgl. $\mathcal{T}$ und $\mathit{val}_r(C \sqcap F)$ ist.

Ad (iv): Folgt unmittelbar aus (+).

Dies schließt den Beweis von Lemma 8.22 ab. □

Um nun auch den Beweis von Theorem 8.11 für $\mathcal{ALE}$ abschließen zu können, bleibt noch zu zeigen, dass die induzierte Extension auch tatsächlich eine Extension des Eingabekonzeptes bzgl. der Eingabe-TBox ist und das minimale Rewriting auch tatsächlich eine Subbeschreibung der induzierten Extension.

Lemma 8.23 *Seien $\mathcal{T}$ eine $\mathcal{ALE}$-TBox und C, F, E $\mathcal{ALE}$-Konzeptbeschreibungen, die die Voraussetzungen zur rekursiven Definition der durch E induzierten Extension in Abbildung 8.3 erfüllen. Sei C^E eine durch diese Definition bestimmte $\mathcal{ALE}$-Konzeptbeschreibung. Dann gilt*

1. *C^E ist eine Extension von C bzgl. $\mathcal{T}$ und F.*
2. *E ist eine Subbeschreibung von C^E bzgl. $\mathcal{T}$ und F.*

Beweis: *Ad (1):* Nach Konstruktion erhält man C^E aus C durch Hinzufügen definierter Konzeptnamen in C. Insbesondere gilt für jeden definierten Konzeptnamen A, der auf Toplevel zu C hinzugefügt wird, $C \sqcap F \sqsubseteq_\mathcal{T} A$ und damit $C \sqcap A \sqcap F \equiv_\mathcal{T} C \sqcap F$. Durch Induktion nach $depth(C)$ folgt leicht $C^E \sqcap F \equiv_\mathcal{T} C \sqcap F$.

Ad (2): Wir zeigen die Behauptung durch Induktion nach $depth(E)$.

Falls $E = \bot$, so ist $C \sqcap F \equiv_\mathcal{T} \bot$ und damit E eine Subbeschreibung von $C^E = C$ bzgl. $\mathcal{T}$ und F.

Für $E \neq \bot$ zeigen wir, dass E der dritten Bedingung in Definition 8.12 genügt:

- Nach Definition ist $def(E) = def(C^E)$.
- Zeige $prim(E) \subseteq prim(C^E)$: Sei $Q \in prim(E)$. Da $C \sqcap F \equiv_\mathcal{T} E \sqcap F$ und C und F keine definierten Konzeptnamen enthalten, folgt mit Satz 8.15, dass $Q \in prim(C \sqcap F)$. Desweiteren ist $Q \notin prim(F)$, da andernfalls E nicht reduziert bzgl. $\mathcal{T}$ und F wäre. Also ist $Q \in prim(C)$.
- Sei $\forall r.E^r$ eine Werterestriktion auf Toplevel von E. Zeige: es existiert eine Werterestriktion der Form $\forall r.D^r$ auf Toplevel von C^E mit E^r ist Subbeschreibung von D^r bzgl. $\mathcal{T}$ und $val_r(F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n))$, wobei $\{A_1, \ldots, A_n\} = def(E)$.

  Wegen $C \sqcap F \sqsubseteq_\mathcal{T} E \sqcap F$ folgt mit Satz 8.15 $val_r(C \sqcap F) \sqsubseteq_\mathcal{T} E^r$. Insbesondere existiert auch eine Werterestriktion der Form $\forall r.C^r$ auf Toplevel von C. Andernfalls würde aus $val_r(F) \sqsubseteq_\mathcal{T} E^r$ ein Widerspruch zur Reduziertheit von E bzgl. $\mathcal{T}$ und F folgen. Mit Lemma 8.22(1) folgt, dass die durch E^r induzierte, rekursiv definierte Extension D^r von C^r bzgl. $\mathcal{T}$ und $val_r(F \sqcap (\mathcal{T}(A_1 \sqcap \ldots \sqcap A_n))$ wohldefiniert ist. Per Induktion folgt, dass E^r eine Subbeschreibung von D^r ist.

- Sei $\exists r.E^r$ eine Existenzrestriktion auf Toplevel von E. Zeige: es existiert eine Existenzrestriktion der Form $\exists r.D^r$ auf Toplevel von C^E mit E^r ist Subbeschreibung von D^r bzgl. $\mathcal{T}$ und $val_r(C \sqcap F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n))$, wobei $\{A_1, \ldots, A_n\} = def(E)$.

  Mit Lemma 8.22(2) folgt, dass eine Existenzrestriktion der Form $\exists r.C^r$ auf Toplevel von C existiert, sodass C^r, E^r und $val_r(C \sqcap F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n))$ die Voraussetzungen der rekursiven Definition der induzierten Extension erfüllen. Für die dabei rekursiv definierte Konzeptbeschreibung D^r folgt per Induktion, dass E^r eine Subbeschreibung von D^r bzgl. $\mathcal{T}$ und $val_r(C \sqcap F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n))$ ist.

Zusammengenommen liefern die oben aufgeführten Punkte, dass E gemäß dem dritten Fall in Definition 8.12 eine Subbeschreibung von C^E bzgl. $\mathcal{T}$ und F ist. $\square$

Nachdem die notwendigen Begriffe eingeführt und die benötigten Hilfsaussagen bewiesen sind, kommen wir nun

Zum Beweis von Theorem 8.11(2): Sei E ein minimales Rewriting von C bzgl. $\mathcal{T}$. Dann ist E auch reduziert bzgl. $\mathcal{T}$ und $\top$, da andernfalls E kein *minimales* Rewriting von C bzgl. $\mathcal{T}$ wäre. Sei C^E eine durch E induzierte Extension von C bzgl. $\mathcal{T}$ und $\top$. Mit Lemma 8.23 folgt, dass E eine Subbeschreibung von C^E bzgl. $\mathcal{T}$ und $\top$ ist. E ist sogar eine Reduktion von C^E bzgl. $\mathcal{T}$ und $\top$. Denn angenommen, E wäre keine Reduktion von C^E bzgl. $\mathcal{T}$ und $\top$. Da wir bereits wissen, dass $C^E \equiv_{\mathcal{T}} E$ und E eine Subbeschreibung von C^E bzgl. $\mathcal{T}$ und $\top$ ist, wäre E nicht minimal mit dieser Eigenschaft. Dann existiert aber eine Subbeschreibung E' von C^E bzgl. $\mathcal{T}$ und $\top$ mit $E' \equiv_{\mathcal{T}} C^E \equiv_{\mathcal{T}} C$ und $|E'| < |E|$. Dann wäre E aber kein minimales Rewriting von C bzgl. $\mathcal{T}$ im Widerspruch zur Annahme. Also ist E Reduktion von C^E bzgl. $\mathcal{T}$ und $\top$. Nun folgt mit Lemma 8.17, dass $\mathsf{reduce}(C^E, \mathcal{T}, \top)$ nichtdeterministisch eine Ausgabe gleich E berechnet. Also liefert auch der minimale Rewriting-Algorithmus nichtdeterministisch eine Ausgabe, die gleich E ist.

8.4 Ein heuristischer Rewriting-Algorithmus für $\mathcal{ALE}$

Der nun vorgestellte heuristische Algorithmus zur Berechnung eines (nicht notwendigerweise minimalen) Rewritings in $\mathcal{ALE}$ berechnet ein Rewriting einer $\mathcal{ALE}$-Konzeptbeschreibung C bzgl. einer $\mathcal{ALE}$-TBox $\mathcal{T}$ in deterministisch polynomieller Zeit (mit einem Orakel zur Entscheidung von Subsumtion modulo $\mathcal{T}$ in $\mathcal{ALE}$). Dieser Algorithmus wurde für die Anwendung prototypisch implementiert (cf Kapitel 9).

Die grundlegende Idee lässt sich wie folgt beschreiben: Anstatt zuerst eine Extension und dann eine Reduktion dieser Extension zu berechnen, werden diese beiden

Schritte verzahnt und ein Rewriting wird in einem einzigen, rekursiven Lauf durch das Eingabekonzept berechnet. Dabei werden sowohl Extension als auch Reduktion gemäß einer *Greedy-Heuristik* bestimmt. Genauer: In jedem Rekursionsschritt berechnet man lokal eine *maximale* Extension, d.h. auf Toplevel des Eingabekonzeptes C wird die Konjunktion $A_1 \sqcap \ldots \sqcap A_n$ *aller* (bzgl. $\sqsubseteq_{\mathcal{T}}$) minimalen definierten Namen A_i, die C subsumieren, hinzugefügt. Dann werden *alle* (negierten) primitiven Konzeptnamen, Werte- und Existenzrestriktionen auf Toplevel von C entfernt, die bzgl. $A_1 \sqcap \ldots \sqcap A_n$ und dem gegebenen Kontext F redundant sind (F ergibt sich aus den Werterestriktionen der vorangehenden Rekursionsschritte). Schließlich wird zu den verbliebenen Werte- und Existenzrestriktionen rekursiv ein Rewriting berechnet, wobei wie im Falle des Reduktionsalgorithmus neben dem Konzept C und der Eingabe-TBox $\mathcal{T}$ auch noch der Kontext F, in dem C zu betrachten ist, an den Algorithmus übergeben wird.

Um tatsächlich ein Rewriting in polynomieller Zeit zu berechnen, ist noch das Auffalten von Konzeptbeschreibungen geeignet einzuschränken: Hierzu bezeichne $\mathcal{T}^*(C)$ die Konzeptbeschreibung, die man aus C erhält, indem alle definierten Namen *auf Toplevel von C* erschöpfend durch ihre definierenden Konzepte aus $\mathcal{T}$ ersetzt. Man beachte, dass im Unterschied zu $\mathcal{T}(C)$ die Konzeptbeschreibung $\mathcal{T}^*(C)$ stets in polynomieller Zeit (bzgl. der Größe von C und $\mathcal{T}$) berechnet werden kann, wenn für alle definierten Namen A stets alle Konjunktionen der Form $A \sqcap A$ modulo Kommutativität und Assoziativität der Konjunktion bestimmt und zu A vereinfacht werden.

Die formale Spezifikation des heuristischen Rewritingalgorithmus für $\mathcal{ALE}$ zeigt Abbildung 8.4. Das folgende Theorem formuliert Korrektheit und Komplexität des heuristischen Rewriting-Algorithmus. Die Korrektheitsaussage ergibt sich unter Verwendung der Charakterisierung der Subsumtion modulo TBox in $\mathcal{ALE}$ aus Satz 8.15 leicht mit einer Induktion nach der Rollentiefe von C. Das Komplexitätsresultat folgt unmittelbar aus dem Algorithmus.

Theorem 8.24 *Seien $\mathcal{T}$ eine $\mathcal{ALE}$-TBox, C und F $\mathcal{ALE}$-Konzeptbeschreibungen, die keine definierten Namen enthalten und $\widehat{C}$ das Ergebnis von $\mathsf{rewrite}(C, \mathcal{T}, F)$. Dann gilt*

1. $\widehat{C} \sqcap F \equiv_{\mathcal{T}} C \sqcap F$ *und*
2. $\widehat{C}$ *wird in deterministisch polynomieller Zeit (mit einem Orakel zur Entscheidung der Subsumtion modulo TBox in $\mathcal{ALE}$) berechnet.*

Mit dem ersten Teil des Theorems folgt also insbesondere, dass die Ausgabe $\widehat{C}$ von $\mathsf{rewrite}(C, \mathcal{T}, \top)$ stets ein Rewriting von C bzgl. $\mathcal{T}$ ist. Leider ist dieses Rewriting i.a. kein minimales Rewriting, wie das folgende Beispiel zeigt.

Eingabe: $\mathcal{ALE}$-Konzeptbeschreibungen C und F und eine $\mathcal{ALE}$-TBox $\mathcal{T}$, wobei C in $\forall$-Normalform ist und keine definierten Namen enthält.

Algorithmus: $\mathsf{rewrite}(C, \mathcal{T}, F)$
Falls $C \sqcap F \equiv_{\mathcal{T}} \bot$, dann $\widehat{C} := \bot$;
Falls $F \sqsubseteq_{\mathcal{T}} C$, dann $\widehat{C} := \top$;
Sonst
Sei $\{A_1, \ldots, A_n\}$ die Menge aller bzgl. $\sqsubseteq_{\mathcal{T}}$ minimalen definierten Namen A_i mit $C \sqcap F \sqsubseteq_{\mathcal{T}} A_i$;
Sei $\{Q_1, \ldots, Q_\ell\} = \mathit{prim}(C) \setminus \mathit{prim}(\mathcal{T}^*(F \sqcap A_1 \sqcap \ldots \sqcap A_n))$;
Für jedes $r \in N_R$
$D^r := \mathsf{rewrite}(\mathit{val}_r(C), \mathcal{T}, \mathit{val}_r(\mathcal{T}^*(F \sqcap A_1 \sqcap \ldots \sqcap A_n)))$;
Sei $\{D_1, \ldots, D_m\} := \mathit{exr}_r(C)$ und $\mathcal{D}^r := \{D_1, \ldots, D_m\}$;
Für $i = 1, \ldots, m$
Falls (1) $D \in \mathcal{D}^r \setminus \{D_i\}$ existiert mit $D \sqcap \mathit{val}_r(C \sqcap \mathcal{T}^*(F)) \sqsubseteq D_i$ oder
(2) $A_1 \sqcap \ldots \sqcap A_n \sqcap F \sqcap \mathit{val}_r(C) \sqsubseteq \exists r.D_i$
dann $\mathcal{D}^r := \mathcal{D}^r \setminus \{D_i\}$;
Definiere
$$\begin{aligned} \widehat{C} \;:=\; & Q_1 \sqcap \ldots \sqcap Q_\ell \sqcap A_1 \sqcap \ldots \sqcap A_n \sqcap \\ & \forall r.D^r \sqcap \bigsqcap_{D \in \mathcal{D}^r} \exists r.\mathsf{rewrite}(D, \mathcal{T}, \mathit{val}_r(C \sqcap \mathcal{T}^*(F))), \end{aligned}$$
wobei $\forall r.D^r$ weggelassen wird, falls $D^r = \top$;
Gib $\widehat{C}$ zurück.

Abbildung 8.4: Der heuristische Rewriting-Algorithmus für $\mathcal{ALE}$.

Beispiel 8.25 *Für eine natürliche Zahl $n > 2$ seien*

$$\begin{aligned} C_n &:= \forall r.(P_1 \sqcap \ldots \sqcap P_n), \\ \mathcal{T}_n &:= \{\, A_i \doteq \forall r.P_i \mid 1 \le i \le n\} \cup \\ & \quad\; \{A_{n+1} \doteq P_1 \sqcap \ldots \sqcap P_n\}. \end{aligned}$$

Der heuristische Rewriting-Algorithmus aus Abbildung 8.4 liefert als Rewriting von C_n bzgl. $\mathcal{T}_n$ die Konzeptbeschreibung $\widehat{C}_n := A_1 \sqcap \ldots \sqcap A_n$ der Größe n. Das eindeutige minimale Rewriting von C_n bzgl. $\mathcal{T}_n$ ist aber gegeben durch die Konzeptbeschreibung $E_n := \forall r.A_{n+1}$ der Größe 2. Folglich kann die Differenz zwischen der Größe des heuristisch berechneten Rewritings und der Größe eines minimalen Rewritings beliebig groß werden.

Im obigen Beispiel liefert der heuristische Algorithmus kein minimales Rewriting, da zu viele definierte Namen auf Toplevel von C_n eingeführt werden. Diese definierten Namen erlauben es dann, die Werterestriktion auf Toplevel von C_n zu entfernen, sodass es unmöglich wird, in einem Rekursionsschritt festzustellen, dass auf einem „tieferen Level“ eine möglicherweise bessere Extension hätte gefunden werden können.

Ein weiterer Unterschied zwischen minimalen und den heuristisch berechneten Rewritings ergibt sich aus der Bedingung, jeweils nur minimale definierte Namen zur Bestimmung der Extension zu berücksichtigen.

Beispiel 8.26 *Es sei $\mathcal{T} = \{A_1 \doteq P_1 \sqcap P_2, A_2 \doteq P_1 \sqcap P_3, A_3 \doteq P_3\}$ und $C = P_1 \sqcap P_2 \sqcap P_3$. Der heuristische Algorithmus liefert als Rewriting $\widehat{C} = A_1 \sqcap A_2$. Dieses Rewriting ist offensichtlich minimal. Im Vergleich mit dem ebenfalls minimalen Rewriting $E = A_1 \sqcap A_3$ enthält $\widehat{C}$ zwar nicht mehr definierte Konzeptnamen, wohl aber speziellere.*

Im Prinzip sind aber die beiden oben aufgezeigten Unterschiede die einzigen zwischen dem heuristisch berechneten und einem minimalen Rewriting. Formal lassen sie sich sehr schön mit dem Begriff *Quasi-Subbeschreibung* fassen. Dieser unterscheidet sich vom Begriff 'Subbeschreibung' in drei Punkten: zum einen dürfen in einer Quasi-Subbeschreibung keine Konzeptbeschreibungen durch $\perp$ ersetzt werden und zum anderen dürfen definierte Namen hinzugefügt oder durch speziellere definierte Namen ersetzt werden.

Definition 8.27 (Quasi-Subbeschreibung) *Sei $\mathcal{T}$ eine $\mathcal{ALE}$-TBox und C eine $\mathcal{ALE}$-Konzeptbeschreibung, die definierte Namen aus $\mathcal{T}$ enthalten kann. Die $\mathcal{ALE}$-Konzeptbeschreibung $\widehat{C}$ ist eine* Quasi-Subbeschreibung von C bzgl. $\mathcal{T}$ *genau dann, wenn man $\widehat{C}$ aus C erhält durch*

1. *Entfernen (negierter) primitiver Konzeptnamen, Werte- und Existenzrestriktionen auf Toplevel von C,*

2. *Hinzufügen definierter Konzeptnamen aus $\mathcal{T}$ auf Toplevel von C,*

3. *Ersetzen definierter Konzeptnamen auf Toplevel von C durch (bzgl. $\sqsubseteq_{\mathcal{T}}$) speziellere definierte Konzeptnamen aus $\mathcal{T}$ und*

4. *Ersetzen der Konzeptbeschreibungen D aus den restlichen Werte- und Existenzrestriktionen $\forall r.D/\exists r.D$ durch Quasi-Subbeschreibungen $\widehat{D}$ bzgl. $\mathcal{T}$.*

Man beachte, dass sowohl das Entfernen als auch das Hinzufügen auf Toplevel optional sind, d.h. weder muss etwas auf Toplevel von C hinzugefügt, noch muss etwas entfernt werden.

Betrachtet man nochmals obige Beispiele, so sieht man leicht, dass das durch den heuristischen Algorithmus berechnete Rewriting jeweils eine Quasi-Subbeschreibung der angegebenen minimalen Rewritings ist. Beispiel 8.25 hat dabei gezeigt, dass die Möglichkeit bestehen muss, Werterestriktionen aus einem minimalen Rewriting E zu entfernen, um ein heuristisch berechnetes Rewriting $\widehat{C}$ als Quasi-Subbeschreibung von E zu erhalten. Das folgende Beispiel zeigt, dass (insbesondere für das hier betrachtete Größenmaß) dasselbe auch für Existenzrestriktionen gilt.

Beispiel 8.28 *Sei* $C = \forall r.P \sqcap \exists r.P$ *und* $\mathcal{T} = \{A \doteq \exists r.P\}$. *Der heuristische Rewriting-Algorithmus liefert* $\widehat{C} = \forall r.P \sqcap A$. *Wegen* $|A| = |\exists r.\top| = 1$ *ist aber auch* $E := \forall r.P \sqcap \exists r.\top$ *ein minimales Rewriting von* C *bzgl.* $\mathcal{T}$. *Offensichtlich ist nun* $\widehat{C}$ *eine Quasi-Subbeschreibung von* E, *da* $\widehat{C}$ *aus* E *durch Hinzufügen von* A *und Entfernen von* $\exists r.\top$ *(jeweils auf Toplevel von* E*) erhalten werden kann.*

Theorem 8.29 *Sei* C *eine* $\mathcal{ALE}$*-Konzeptbeschreibung in* $\forall$*-Normalform, die keine definierten Namen enthält,* $\mathcal{T}$ *eine* $\mathcal{ALE}$*-TBox und* E *ein minimales Rewriting von* C *bzgl.* $\mathcal{T}$. *Dann ist das Ergebnis* $\widehat{C}$ *von* $\mathsf{rewrite}(C, \mathcal{T}, \top)$ *eine Quasi-Subbeschreibung von* E.

Beweis: Die Behauptung des Theorems ergibt sich leicht aus der folgenden

Behauptung: Sei $\mathcal{T}$ eine $\mathcal{ALE}$-TBox und seien $C, E, F_1, F_2, \widehat{C}$ $\mathcal{ALE}$-Konzeptbeschreibungen mit

- C ist in $\forall$-Normalform und enthält keine definierten Namen,
- $F_1 \sqsubseteq_{\mathcal{T}} F_2$,
- E ist reduziert bzgl. $\mathcal{T}$ und F_2,
- E enthält keine redundanten definierten Namen, d.h. geht E' aus E durch entfernen definierter Namen hervor, so gilt $E' \sqcap F_2 \not\equiv_{\mathcal{T}} E \sqcap F_2$,
- $C \sqcap F_1 \equiv_{\mathcal{T}} E \sqcap F_2$ und
- $\widehat{C}$ ist die Ausgabe von $\mathsf{rewrite}(C, \mathcal{T}, F_1)$.

Dann gilt $\widehat{C} = \top$ oder $\widehat{C} = \bot$ oder

1. $prim(\widehat{C}) \subseteq prim(E)$,
2. für alle $A \in def(E)$ existiert ein $A' \in def(\widehat{C})$ mit $A' \sqsubseteq_{\mathcal{T}} A$,
3. für jede Werterestriktion $\forall r.\widehat{C}'$ auf Toplevel von $\widehat{C}$ existiert eine Werterestriktion $\forall r.E'$ auf Toplevel von E, sodass $\widehat{C}'$ Quasi-Subbeschreibung von E' ist und
4. für jede Existenzrestriktion der Form $\exists r.\widehat{C}'$ auf Toplevel von $\widehat{C}$ existiert ein $E_j \in exr_r(E)$, sodass $\widehat{C}'$ Quasi-Subbeschreibung von E_j ist.

Als minimales Rewriting von C bzgl. $\mathcal{T}$ ist E reduziert bzgl. $\mathcal{T}$ und $\top$ und enthält keine redundanten definierten Namen. Mit $F_1 = F_2 = \top$ folgt daher aus der Behauptung, dass $\widehat{C} = \top$ oder $\widehat{C} = \bot$ oder die Punkte (1)–(4) erfüllt sind. Ist $\widehat{C} = \top$, so folgt aus $C \equiv_{\mathcal{T}} \widehat{C}$ auch $E \equiv_{\mathcal{T}} \top$. Da E minimales Rewriting ist und $|\top| = 0$, folgt $E = \top$ (modulo Idempotenz der Konjunktion). Ist $\widehat{C} = \bot$, so folgt wiederum auch $E = \bot$. Gelten die Punkte (1)–(4) der Behauptung, so besagen diese gerade, dass $\widehat{C}$ eine Quasi-Subbeschreibung von E ist.

Zum Beweis der Behauptung: Ist $\widehat{C} = \top$ oder $\widehat{C} = \bot$, so ist nichts zu zeigen. Sei also $\widehat{C} \notin \{\top, \bot\}$.

Ad (1): Sei $Q \in \mathit{prim}(\widehat{C})$. Angenommen, $Q \notin \mathit{prim}(E)$. Dann folgt mit Satz 8.15 aus $C \sqcap F_1 \equiv_{\mathcal{T}} E \sqcap F_2$, dass Q auf Toplevel von $\mathcal{T}(F_2)$ oder $\mathcal{T}(A)$ für ein $A \in \mathit{def}(E)$ auftritt. Wegen $F_1 \sqsubseteq_{\mathcal{T}} F_2$ und Bedingung (2) folgt in beiden Fällen ein Widerspruch zur Definition der Menge $\{Q_1, \dots, Q_\ell\}$ im Rewriting-Algorithmus aus Abbildung 8.4.

Ad (2): Sei $A \in \mathit{def}(E)$. Aus $C \sqcap F_1 \equiv_{\mathcal{T}} E \sqcap F_2$ folgt $C \sqcap F_1 \sqsubseteq_{\mathcal{T}} A$. Ist $A \in \mathit{def}(\widehat{C})$, so ist nichts zu zeigen. Ist $A \notin \mathit{def}(\widehat{C})$, so ist A kein minimaler Konzeptname mit der Eigenschaft $C \sqcap F_1 \sqsubseteq_{\mathcal{T}} A$. Dann existiert aber ein A' mit $A' \sqsubset_{\mathcal{T}} A$, $C \sqcap F_1 \sqsubseteq_{\mathcal{T}} A'$ und $A' \in \mathit{def}(\widehat{C})$ und es folgt das Gewünschte.

Ad (3): Sei $\forall r.\widehat{C^r}$ eine Werterestriktion auf Toplevel von $\widehat{C}$. Nach Konstruktion ist $\widehat{C^r} \neq \top$ und es existiert eine (eindeutige) Werterestriktion $\forall r.C^r$ auf Toplevel von C, sodass $\widehat{C^r}$ das Ergebnis von $\mathsf{rewrite}(C^r, \mathcal{T}, \mathit{val}_r(\mathcal{T}^*(F_1 \sqcap A_1 \sqcap \ldots \sqcap A_n)))$ ist, wobei $\{A_1, \dots, A_n\} = \mathit{def}(\widehat{C})$. Wir zeigen, dass

(a) eine Werterestriktion der Form $\forall r.E^r$ auf Toplevel von E existiert;

(b) E^r reduziert bzgl. $\mathcal{T}$ und $\mathit{val}_r(\mathcal{T}^*(F_2 \sqcap B_1 \sqcap \ldots \sqcap B_m))$ ist, wobei $\{B_1, \dots, B_m\} = \mathit{def}(E)$;

(c) $C^r \sqcap \mathit{val}_r(\mathcal{T}^*(F_1 \sqcap A_1 \sqcap \ldots \sqcap A_n)) \equiv_{\mathcal{T}} E^r \sqcap \mathit{val}_r(\mathcal{T}^*(F_2 \sqcap B_1 \sqcap \ldots \sqcap B_m))$; und

(d) $\mathit{val}_r(\mathcal{T}^*(F_1 \sqcap A_1 \sqcap \ldots \sqcap A_n)) \sqsubseteq_{\mathcal{T}} \mathit{val}_r(\mathcal{T}^*(F_2 \sqcap B_1 \sqcap \ldots \sqcap B_m))$.

Aus diesen Punkten folgt per Induktion, dass $\widehat{C^r} = \top$ oder $\widehat{C^r} = \bot$ oder die Bedingungen (1)–(4) für $\widehat{C^r}$ und E^r erfüllt sind. $\widehat{C^r} = \top$ liefert einen Widerspruch zur Konstruktion. Ist $\widehat{C^r} = \bot$, so folgt aus (a) und (c) und der Reduziertheit von E auch $E^r = \bot$, womit also $\widehat{C^r}$ eine Quasi-Subbeschreibung von E^r ist. Gelten die Bedingungen (1)–(4) für $\widehat{C^r}$ und E^r, so folgt ebenfalls, dass $\widehat{C^r}$ eine Subbeschreibung von E^r ist.

Zum Beweis von Punkt (3) bleiben also die Punkte (a)–(d) zu zeigen.

Ad (a): Angenommen, es existiert keine Werterestriktion der Form $\forall r.E^r$ auf Toplevel von E. Dann folgt aus der Äquivalenz $C \sqcap F_1 \equiv_{\mathcal{T}} E \sqcap F_2$, dass $\mathit{val}_r(\mathcal{T}^*(F_2 \sqcap B_1 \sqcap \ldots \sqcap B_m)) \equiv_{\mathcal{T}} C^r \sqcap \mathit{val}_r(\mathcal{T}^*(F_1))$. Wegen $F_1 \sqsubseteq_{\mathcal{T}} F_2$ und Punkt (2) der Behauptung folgt hieraus aber $\mathit{val}_r(\mathcal{T}^*(F_1 \sqcap A_1 \sqcap \ldots \sqcap A_n)) \sqsubseteq_{\mathcal{T}} C^r$ im Widerspruch dazu, dass eine Werterestriktion der Form $\forall r.\widehat{C^r}$ auf Toplevel von $\widehat{C}$ existiert.

Ad (b): E^r ist reduziert bzgl. $\mathcal{T}$ und $\mathit{val}_r(\mathcal{T}^*(F_2 \sqcap B_1 \sqcap \ldots \sqcap B_m))$, da andernfalls E nicht reduziert bzgl. $\mathcal{T}$ und F_2 wäre.

Ad (c): Die Äquivalenz folgt mit Satz 8.15 aus der Äquivalenz $C \sqcap F_1 \equiv_{\mathcal{T}} E \sqcap F_2$.

Ad (d): Die Subsumtionsbeziehung folgt offensichtlich wegen $F_1 \sqsubseteq_{\mathcal{T}} F_2$ und Punkt (2) der Behauptung.

Zum Abschluss des Beweises von Theorem 8.29 ist noch Punkt (4) aus der Behauptung zu zeigen.

Ad (4): Sei $\exists r.\widehat{C'}$ eine Existenzrestriktion auf Toplevel von $\widehat{C}$. Nach Konstruktion existiert eine Existenzrestriktion der Form $\exists r.C_i$ auf Toplevel von C, sodass $\widehat{C'}$ das Ergebnis von $\mathsf{rewrite}(C_i, \mathcal{T}, \mathit{val}_r(\mathcal{T}^*(F_1 \sqcap A_1 \sqcap \ldots \sqcap A_n)))$ ist. Wir zeigen, dass eine Existenzrestriktion der Form $\exists r.E_j$ auf Toplevel von E existiert, sodass

(a) E_j reduziert bzgl. $\mathcal{T}$ und $\mathit{val}_r(\mathcal{T}^*(E \sqcap F_2))$ ist;

(b) $C_i \sqcap \mathit{val}_r(C \sqcap \mathcal{T}^*(F_1)) \equiv_{\mathcal{T}} E_j \sqcap \mathit{val}_r(\mathcal{T}^*(E \sqcap F_2))$; und

(c) $\mathit{val}_r(C \sqcap \mathcal{T}^*(F_1)) \equiv_{\mathcal{T}} \mathit{val}_r(\mathcal{T}^*(E \sqcap F_2))$.

Aus diesen Punkten folgt per Induktion, dass $\widehat{C'} = \top$ oder $\widehat{C'} = \bot$ oder die Bedingungen (1)–(4) für $\widehat{C'}$ und E_j erfüllt sind. $\widehat{C'} = \bot$ steht im Widerspruch zur Annahme $\widehat{C} \neq \bot$. Ist $\widehat{C'} = \top$, so gilt $\mathit{val}_r(C \sqcap \mathcal{T}^*(F_1 \sqcap A_1 \sqcap \ldots \sqcap A_n)) \sqsubseteq_{\mathcal{T}} C_i$. Also existiert ein E_j mit

$$\mathit{val}_r(C \sqcap \mathcal{T}^*(F_1)) \sqsubseteq_{\mathcal{T}} E_j \sqcap \mathit{val}_r(\mathcal{T}^*(E \sqcap F_2)).$$

Mit (c) folgt nun $\mathit{val}_r(\mathcal{T}^*(E \sqcap F_2)) \sqsubseteq_{\mathcal{T}} E_j$. Dann enthält E_j aber nur definierte Namen, da sonst E nicht reduziert bzgl. $\mathcal{T}$ und F_2 wäre. Dies liefert aber einen Widerspruch zur Voraussetzung, dass E keine redundanten definierten Namen enthält, denn würde man die definierten Namen aus E_j entfernen, würde man eine modulo $\mathcal{T}$ äquivalente Konzeptbeschreibung erhalten. Also folgt auch für $\widehat{C'} = \top$ ein Widerspruch, sodass nach Induktion die Bedingungen (1)–(4) erfüllt sind. Aus diesen folgt, dass $\widehat{C'}$ eine Quasi-Subbeschreibung von E_j ist.

Zum Beweis von Punkt (4) bleiben also obige Punkte (a)–(c) zu zeigen.

Punkt (c) folgt aus Punkt (c) im Beweis von Punkt (3) und wegen $\mathit{val}_r(C \sqcap \mathcal{T}^*(F_1)) \sqsubseteq_{\mathcal{T}} \mathit{val}_r(\mathcal{T}^*(A_1 \sqcap \ldots \sqcap A_n))$.

Es bleiben also die Punkte (a) und (b) zu zeigen. Aus der Äquivalenz $C \sqcap F_1 \equiv_{\mathcal{T}} E \sqcap F_2$ folgt mit Satz 8.15, dass ein $D \in \mathit{exr}_r(\mathcal{T}^*(E \sqcap F_2))$ existiert mit $D \sqcap \mathit{val}_r(\mathcal{T}^*(E \sqcap F_2)) \sqsubseteq_{\mathcal{T}} C_i$.

Angenommen, $D \notin \mathit{exr}_r(E)$, also $D \in \mathit{exr}_r(\mathcal{T}^*(F_2 \sqcap B_1 \sqcap \ldots \sqcap B_m))$, wobei $\{B_1, \ldots, B_m\} = \mathit{def}(E)$. Dann folgt wegen Punkt (2) der Behauptung und $F_1 \sqsubseteq_{\mathcal{T}} F_2$, dass ein $D' \in \mathit{exr}_r(\mathcal{T}^*(F_1 \sqcap A_1 \sqcap \ldots \sqcap A_n))$ existiert mit $D' \sqsubseteq_{\mathcal{T}} D$. Nach Konstruktion wäre dann aber C_i aus $\mathcal{D}_r$ entfernt worden im Widerspruch zur Annahme $\widehat{C'} \in \mathit{exr}_r(\widehat{C})$.

Also ist $D = E_j$ für ein $E_j \in \mathit{exr}_r(E)$. Mit (c) folgt

(∗) $E_j \sqcap \mathit{val}_r(C \sqcap \mathcal{T}^*(F_1)) \sqsubseteq_{\mathcal{T}} C_i \sqcap \mathit{val}_r(C \sqcap \mathcal{T}^*(F_1))$.

Wiederum aus der Äquivalenz $C \sqcap F_1 \equiv_{\mathcal{T}} E \sqcap F_2$ folgt mit Satz 8.15, dass ein $D \in \mathit{exr}_r(C \sqcap \mathcal{T}^*(F_1))$ existiert mit

$$D \sqcap \mathit{val}_r(C \sqcap \mathcal{T}^*(F_1)) \sqsubseteq_\mathcal{T} E_j.$$

Mit $(*)$ folgt $D \sqcap \mathit{val}_r(C \sqcap \mathcal{T}^*(F_1)) \sqsubseteq_\mathcal{T} C_i \sqcap \mathit{val}_r(C \sqcap \mathcal{T}^*(F_1))$. Angenommen, es gilt $D \sqcap \mathit{val}_r(C \sqcap \mathcal{T}^*(F_1)) \sqsubset_\mathcal{T} C_i \sqcap \mathit{val}_r(C \sqcap \mathcal{T}^*(F_1))$. Dann wäre C_i aus $\mathcal{D}_r$ entfernt worden im Widerspruch zur Annahme $\widehat{C'} \in \mathit{exr}_r(\widehat{C})$. Also folgt $C_i \sqcap \mathit{val}_r(C \sqcap \mathcal{T}^*(F_1)) \equiv_\mathcal{T} D \sqcap \mathit{val}_r(C \sqcap \mathcal{T}^*(F_1))$. Mit Punkt (c) folgt $C_i \sqcap \mathit{val}_r(C \sqcap \mathcal{T}^*(F_1)) \equiv_\mathcal{T} E_j \sqcap \mathit{val}_r(\mathcal{T}^*(E \sqcap F_2))$, also Punkt (b). Punkt (a) folgt wie Punkt (b) im Beweis von Punkt (3). □

Bezüglich der in Abschnitt 5.3 eingeführten $\mathcal{ALE}$-Beschreibungsbäume lässt sich das Resultat aus Lemma 8.29 auch wie folgt interpretieren: Sei $\mathcal{G}(E)$ der $\mathcal{ALE}$-Beschreibungsbaum zu einem minimalen Rewriting E von C bzgl. $\mathcal{T}$ und sei $\mathcal{G}(\widehat{C})$ der $\mathcal{ALE}$-Beschreibungsbaum zum Ergebnis $\widehat{C}$ von $\mathsf{rewrite}(C, \mathcal{T}, \top)$. Dabei dürfen die Knotenlabel nun natürlich auch definierte Konzeptnamen enthalten. Dann gilt:

- es existiert eine injektive Abbildung φ von $\mathcal{G}(\widehat{C})$ nach $\mathcal{G}(E)$, d.h. die Baumstruktur von $\mathcal{G}(\widehat{C})$ ist eventuell kleiner als die von $\mathcal{G}(E)$;
- das Label eines Knoten v in $\mathcal{G}(\widehat{C})$ kann weniger (negierte) primitive Konzeptnamen;
- zu jedem definierten Namen A im Label eines Knotens $\varphi(v)$ in $\mathcal{G}(E)$ enthält das Label von v in $\mathcal{G}(\widehat{C})$ einen definierten Namen A' mit $A' \sqsubseteq_\mathcal{T} A$, kann aber mehr definierte Konzeptnamen als das Label von $\varphi(v)$ enthalten.

Kapitel 9

Implementierung und Erfahrungen

Dieses Kapitel fasst einen Teil der Erkenntnisse und Ergebnisse über den Einsatz von BL-Systemen in der Prozesstechnik zusammen, die wir im Rahmen unserer Zusammenarbeit mit den Mitarbeitern vom Lehrstuhl für Prozesstechnik gewonnen haben. Abschnitt 9.1 beschreibt kurz die prototypische Implementierung der in den vorherigen Kapiteln entwickelten Algorithmen. Die Abschnitte 9.2 und 9.3 geben Erfahrungen, die bei Entwicklung, Anbindung und Einsatz des Prototyps in der Prozesstechnik gemacht wurden, wieder.

9.1 Implementierung der Nicht-Standardinferenzdienste

Im Rahmen dieser Arbeit wurden die in den vorherigen Kapiteln eingeführten Algorithmen zu den Nicht-Standardinferenzen prototypisch in Common Lisp implementiert. Grund für die Wahl von Common Lisp als Implementierungssprache ist, dass das sehr leistungsfähige BL-System FaCT [Hor98] ebenfalls in Common Lisp implementiert ist und dieses System bereits erfolgreich zur Klassifizierung der Wissensbasis eingesetzt wurde. Die Nicht-Standardinferenzdienste können außerdem durch eine Erweiterung des Quellcodes direkt auf dieses System aufgesetzt werden (FaCT steht unter der URL `http://www.cs.man.ac.uk/~horrocks/FaCT/` zum Download zur Verfügung). Ein sehr großer Vorteil hiervon ist, dass der sehr effizient implementierte Test auf Subsumtion modulo TBox in FaCT ohne zusätzlichen Aufwand als Hilfsprozedur für die neuen Dienste eingesetzt werden kann und somit auf die Implementierung eines Subsumtionstests für den Prototyp verzichtet werden konnte. Die Wissensbasis wird jeweils in Form von Listen gespeichert: je eine Liste für die Mengen der Rollennamen, der Konzeptnamen und der Individuen. Konzeptdefinitionen $A \doteq C$ werden als Tupel der Form $(A\ C)$ dargestellt und in einer Liste gespeichert. Der Zugriff auf die definierenden Konzepte beim Auffalten von Konzeptbeschreibungen erfolgt über eine geeignete Indexierung der Liste von Konzeptnamen. Zu jedem

Individuum a wird genau eine Konzeptassertion in Form eines Tupel $(a\ C)$ in einer Liste gespeichert. Tritt a in mehreren Konzeptassertionen $a : C_1, \ldots, a : C_n$ auf, so werden diese zu einer Konzeptassertion $a : \sqcap_{1 \leq i \leq n} C_i$ zusammengefasst. Tritt a in keiner Konzeptassertion auf, so ist die zugehörige Konzeptbeschreibung gleich $\top$. Die Rollenassertionen der Form $(a, b) : r$ werden als Tripel $(a\ b\ r)$ ebenfalls in einer Liste gespeichert. Im folgenden sind die wesentlichen Ideen und Besonderheiten der Implementierung der neuen Dienste zusammengestellt. Erfahrungen mit Beispielen bzgl. Laufzeitverhalten und Platzaufwand werden zusammenhängend in Abschnitt 9.3 beschrieben.

Berechnung des LCS. Die Berechnung des LCS von n $\mathcal{ALE}$-Konzeptbeschreibungen $C_1, \ldots, C_n$ erfolgt durch iteriertes Anwenden der binären LCS-Operation, wobei $\perp$ für $n = 0$ und C_1 für $n = 1$ als Ergebnis zurückgegeben wird. Die binäre LCS-Operation erwartet als Eingabe zwei aufgefaltete $\mathcal{ALE}$-Konzeptbeschreibungen C_1, C_2. Gilt $C_1 \sqsubseteq C_2$ ($C_2 \sqsubseteq C_1$), so wird C_2 (C_1) zurückgegeben. Sonst werden die $\mathcal{ALE}$-Normalformen zu C_1 und C_2 und aus diesen der LCS berechnet. Durch die Listendarstellung von Konzeptbeschreibungen ist es sehr leicht, den LCS, d.h. die Konzeptbeschreibung zum Produkt der $\mathcal{ALE}$-Beschreibungsbäume zu den $\mathcal{ALE}$-Normalformen, rekursiv zu berechnen.

Da das vorrangige Ziel der Implementierung darin bestand, den Einsatz von Nicht-Standardinferenzdiensten in unserer Prozesstechnikanwendung qualitativ zu bewerten, wurden zunächst keine großen Anstrengungen für eine effiziente Implementierung der LCS-Operation unternommen. Die erreichten Laufzeiten der binären LCS-Operation liegen im Bereich weniger Sekunden und der Platzaufwand war mit einigen hunderttausend Zellen großen Listen recht groß. Hier ist aber durch geeignete Optimierungen noch eine deutliche Verbesserung zu erwarten.

Berechnung der LCS-Hierarchie. Die LCS-Hierarchie wird wie in Kapitel 6 beschrieben mit Hilfe des Objektexplorationsalgorithmus berechnet. Die Implementierung dieses Algorithmus folgt dabei der Beschreibung in Abbildung 6.2. Da die LCS-Operation einen (insbesondere im Vergleich zum Subsumtionstest) hohen Platz- und Zeitbedarf hat, wurde darauf geachtet, die Zahl der LCS-Operationen möglichst gering zu halten. Beispielsweise wird die Anfrage an den Experten, ob die Implikation $A_i \to A_i'' \setminus A_i$ gültig ist, d.h. ob der LCS zur Menge A_i den LCS zur Menge $A_i'' \setminus A_i$ subsumiert, beantwortet, indem nur der LCS zur Menge A_i berechnet und dann $C \sqsubseteq \mathsf{lcs}(A_i)$ für alle $C \in A_i'' \setminus A_i$ getestet wird (vgl. Beweis von Lemma 6.11). Desweiteren kann durch Wiederverwenden bereits berechneter LCS jede n-äre LCS-Operation auf eine binäre LCS-Operation reduziert werden: Wird der LCS zur Teilmenge $\{C_{i_1}, \ldots, C_{i_m}\}$, $m > 2$, berechnet, so garantiert die Aufzählung der Teilmengen von $\{C_1, \ldots, C_n\}$ gemäß der lektischen Ordnung, dass, wenn C_{i_m} als letztes Element hinzugefügt wurde, der LCS zur Menge $\{C_{i_m}, \ldots, C_{i_{m-1}}\}$ bereits berechnet wurde. Der LCS von $\{C_{i_1}, \ldots, C_{i_m}\}$ kann folglich durch die binäre LCS-Operation $\mathsf{lcs}(\mathsf{lcs}(C_{i_1}, \ldots, C_{i_{m-1}}), C_{i_m})$ berechnet werden.

Berechnung der Approximation des MSC. Für eine erste qualitative Bewertung der Nicht-Standardinferenzen betrachten wir als Approximation des MSC nur die Konjunktion der Konzeptbeschreibungen C aus Konzeptassertionen der Form $a : C$ in der gegebenen ABox $\mathcal{A}$. Es ist aber im Rahmen der Fortführung der Kooperation (neben weiterführenden theoretischen Untersuchungen des MSC in $\mathcal{ALE}$) die Implementierung der am Ende von Abschnitt 7.4 beschriebenen Approximation der k-Approximation in $\mathcal{ALE}$ geplant.

Berechnung eines Rewritings. Hier wurde der in Abbildung 8.4 eingeführte deterministische Algorithmus zur Berechnung eines (nicht notwendig) minimalen Rewritings implementiert. Als Eingabe erwartet der Rewriting-Dienst aufgefaltete $\mathcal{ALE}$-Konzeptbeschreibungen (wie sie ja insbesondere durch die LCS-Operation geliefert werden). Bei der Berechnung der Mengen $\{A_1, \ldots, A_n\}$ definierter Namen, die die Eingabe subsumieren, wird auf die von FaCT eigentlich für die Klassifizierung und das Browsen der Wissensbasis zur Verfügung gestellte Prozedur zur Berechnung aller Oberkonzepte, d.h. aller definierten Namen, die eine gegebene Konzeptbeschreibung subsumieren, zurückgegriffen. Durch den Rückgriff auf diese Prozedur sowie den sehr effizienten Subsumtionstest von FaCT haben wir eine sehr effiziente Implementierung des heuristischen Rewriting-Algorithmus für $\mathcal{ALE}$ erhalten.

9.2 Anbindung von FaCT an ModKit

Nach [MvWB99] ist ModKit in der zweiten Version als eine mächtige Modellierungsumgebung zu sehen, in der verschiedenste Werkzeuge zur Prozessmodellierung und Simulation integriert sind (s. Abbildung 9.1 rechts). Wie in der ersten Version [BLM96, BLM97, BLM00] stehen Editoren zur Modellierung von Prozessen auf konzeptuellem Level zur Verfügung. Im Zuge des Re-Engineering wurde aber die Wissensbasis als eigenständige Komponente namens ROME (*Repository Of a Modeling Environment*) [vWM00] realisiert, die nun von ModKit aus verwaltet wird (Abbildung 9.1). Das Repository umfasst neben dem strukturierten Katalog von Klassen und Bausteinen auch Modelldokumentationen, Repräsentationen in werkzeugspezifischen Formaten und Metadaten (s. Abbildung 9.2). ROME dient somit als Basis für den Datenaustausch in der gesamten Modellierungsumgebung [MvWB99, vWM00].

Design, Architektur und Implementierung von ROME wird in [vWM00] beschrieben. Hier beschränken wir uns auf eine kurze Beschreibung der Architektur (s. Abbildung 9.3). In ROME werden die Daten in einer *objektorientierten Datenbank* gespeichert. Der Zugriff und die Verwaltung der Daten erfolgt mit Hilfe von Operationen, die

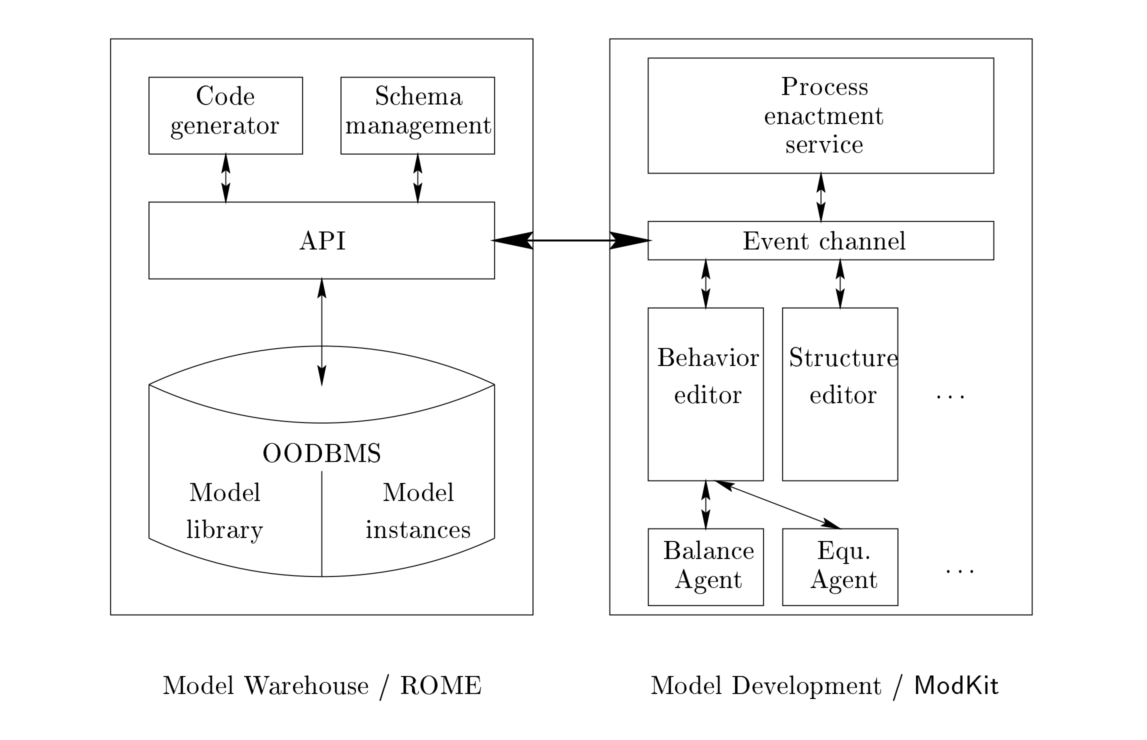


Abbildung 9.1: Ein Überblick über die Modellierungsumgebung ModKit und ROME [MvWB99].

durch den *Repository Manager* bereitgestellt werden. Dieser ist in C++ implementiert und über *CORBA-Schnittstellen* ansprechbar. Die Anbindung von Klienten an ROME erfolgt über ein *API* (wie im Falle von ModKit) oder über einen *Code-Generator*, der aus Eingabedateien, die vom Klienten geschickt werden, den Zugriffscode erzeugt. Änderungen in der Datenbank lösen ein *Event* aus, über das angebundene Werkzeuge durch den zugehörigen *Event Service* informiert werden.

Wie in Abbildung 9.2 angedeutet verwaltet ROME die Spezialisierungshierarchie der Klassen und Bausteine. Im Rahmen der Arbeit von Ulrike Sattler [Sat98] wurde das BL-System Crack über ein geeignetes API direkt an ModKit angebunden. Im Unterschied dazu wird die geplante Anbindung von FaCT an die neue Modellierungsumgebung sowohl ModKit als auch ROME einbeziehen. Da ModKit aber die Rolle des Verwalters für das Repository spielt, wird der Modellierer nach wie vor über ModKit Zugriff auf die Daten in ROME erhalten und auch über ModKit die Dienste von FaCT bzw. die Nicht-Standardinferenzdienste nutzen. Der dazu notwendige Datenaustausch zwischen ROME und FaCT kann mit Hilfe von CORBA erfolgen, wobei die Daten in geeigneten XML-Formaten ausgetauscht werden. Dieser Ansatz zur Anbindung von BL-Systemen an Anwendungen wird auch durch die Entwickler von FaCT und seinen Folgesystemen unter Leitung von Ian Horrocks unterstützt [BHPST99].

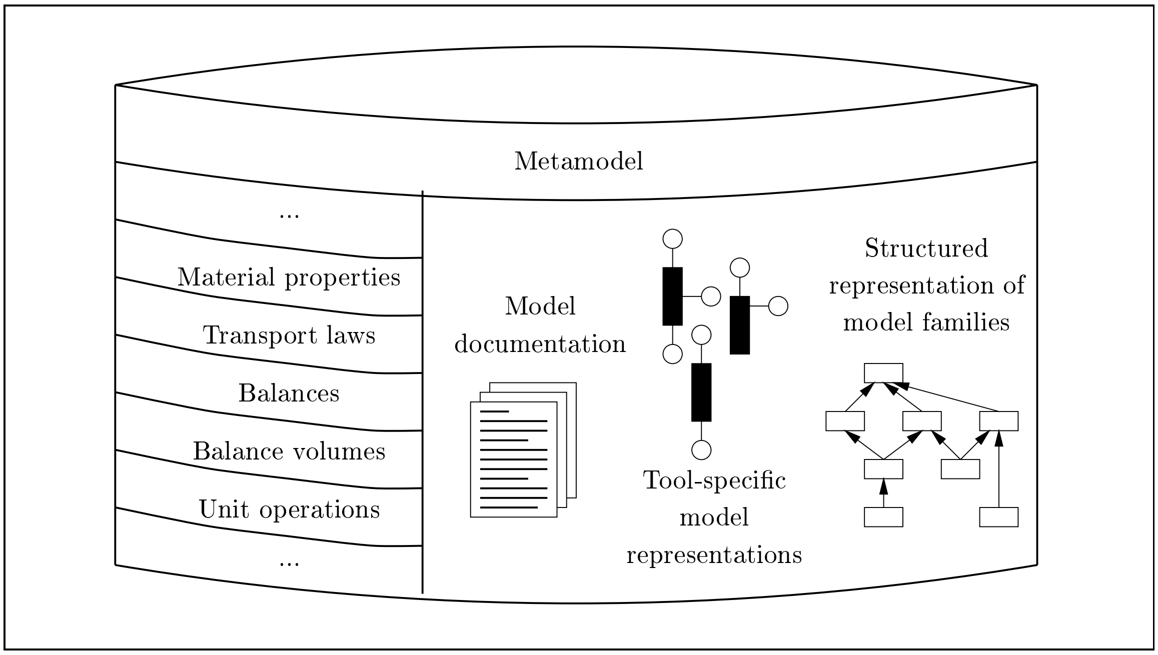


Abbildung 9.2: Ein Überblick über die von ROME bereitgestellten Daten [MvWB99,vWM00].

Im Rahmen dieser Arbeit war eine Realisierung der oben beschriebenen Anbindung aufgrund der noch laufenden Implementierungsarbeiten sowohl an ROME als auch an einem CORBA-fähigen BL-System nicht möglich. Da das ROME zugrundeliegende Datenmodell jedoch im wesentlichen mit VeDa übereinstimmt, konnte für die Untersuchung des Verhaltens der Nicht-Standardinferenzen in der Anwendung auf einen Auszug aus VeDa zurückgegriffen. Genauer wurden ein Auszug aus dem in [SM98a] beschriebenen strukturellen Datenmodell von Hand in eine FaCT-Wissensbasis übersetzt und Klassen (z.B. von Reaktoren) mit Hilfe der in dieser Wissensbasis definierten Konzepte beschrieben. Die gewonnenen Erfahrungen und erzielten Ergebnisse werden nun im folgenden Unterabschnitt beschrieben.

9.3 Erfahrungen

Zum Einsatz der Standardinferenzdienste

Die Erfahrungen, die bei der Anbindung des BL-Systems Crack und der Nutzung seiner Standardinferenzdienste gemacht wurden, sind in [Sat98] dokumentiert, werden aber hier der Vollständigkeit halber nochmals zusammengefasst. Es ergaben sich im wesentlichen zwei positive Effekte.

Der erste positive Effekt trat bereits während der Anbindung von Crack an ModKit auf. Bei der Implementierung und dem Testen der Übersetzung von Klassenbeschrei-

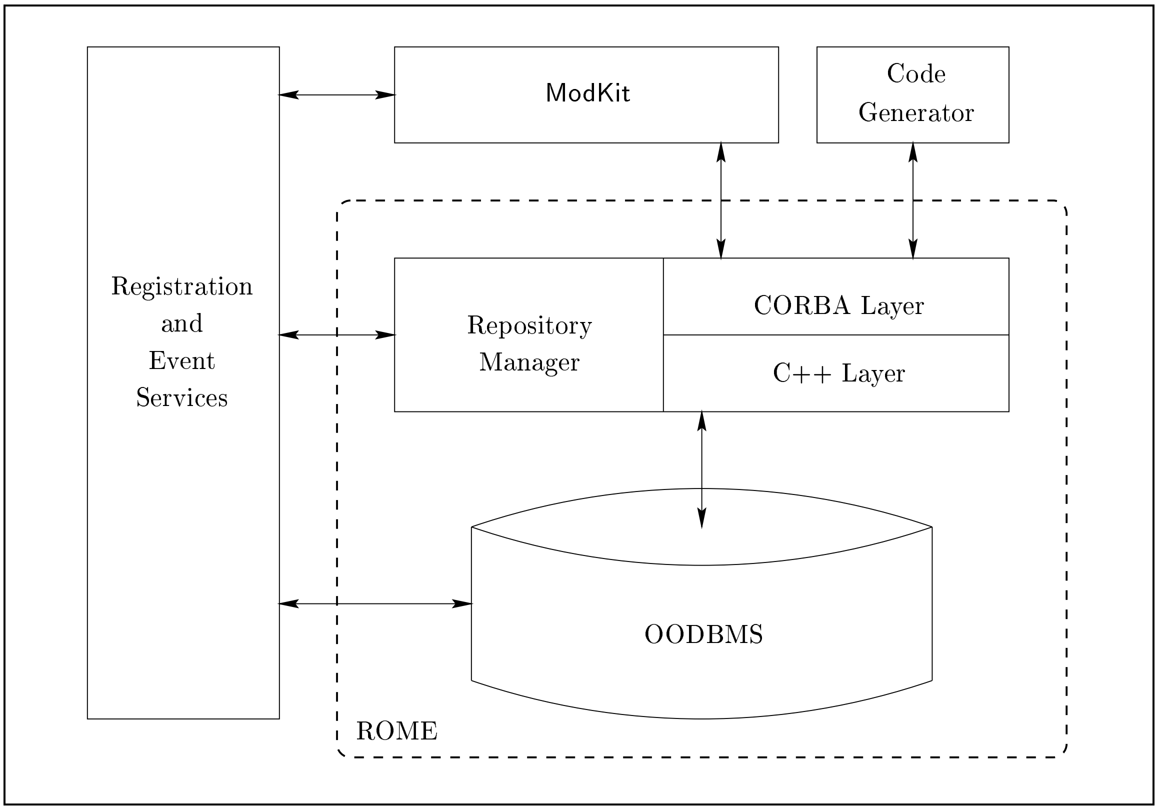


Abbildung 9.3: Die Architektur von ROME nach [vWM00].

bungen in Konzeptbeschreibungen wurden zum einen Inkonsistenzen aufgedeckt und zum anderen viele Klassen gefunden, deren Beschreibung nur aus einer Liste von Oberklassen bestand. Der Grund für ihre Einführung und ihre Bedeutung ergaben sich jeweils nur intuitiv aus dem Namen. Für diese Klassen wurden dann bessere, deklarative Beschreibungen eingeführt, die nicht nur zu einer tieferen und damit besseren Klassenhierarchie führten, sondern durch die die Hierarchie und die Klassen für die Modellierer auch verständlicher und leichter lesbar wurden.

Der zweite positive Effekt ist die Unterstützung der Analyse und Pflege der Wissensbasis: durch Berechnung und Visualisierung der Klassenhierarchie wird die Suche nach Klassen und Bausteinen unterstützt. Darüberhinaus wird sichergestellt, dass manuelle Erweiterungen der Klassenhierarchie auf Seiten von ModKit konsistent mit der zugehörigen Subsumtionshierarchie sind. Stimmen die Positionen einer neuen Klasse in der Klassenhierarchie und der Subsumtionshierarchie nicht überein, so wird dies erkannt und die Position bzw. die Klassenbeschreibung können korrigiert werden.

Zur Übersetzung von Klassenbeschreibungen

Wie bereits in Kapitel 4 erwähnt haben genauere Untersuchungen gezeigt, dass die meisten der von den Prozesstechnikern für die Repräsentation in einem BL-System als relevant eingestuften Aspekte der Klassenbeschreibungen in Konjunktionen von Existenz- und Werterestriktionen, also in $\mathcal{ALE}$-Konzeptbeschreibungen übersetzt werden können. Die Idee der Übersetzung lässt sich wie folgt beschreiben: Zu einer Klasse f, die durch einen Frame F beschrieben wird, erhält man eine Konzeptdefinition der Form $f \doteq C(F)$, wobei sich $C(F)$ aus ausgewählten Einträgen in F ergibt. Diese Einträge sind im wesentlichen von der Form $\langle attribute \rangle$ $\langle domain \rangle$ $\langle flags \rangle$. Sichern die *flags*-Werte zu jeder Instanz von f die Existenz eines Attributes aus dem *domain* zu, so liefert ein solcher Eintrag eine Existenzrestriktion der Form $\exists attribute.domain$. Folgt aus den *flags*-Werten, dass alle Instanziierungen des Attributes Instanzen von *domain* sein müssen, so liefert der Eintrag (ggf. zusätzlich zu einer Existenzrestriktion) eine Werterestriktion der Form $\forall attribute.domain$.

Als Beispiel betrachte man die Klassenbeschreibung zur Klasse DEVICE-MODEL und die zugehörige Konzeptdefinition in Abbildung 9.4. Aus Gründen der Lesbarkeit erfolgt die Beschreibung der Klasse in der Framesprache von VeDa und beschränkt sich auf die aus struktureller Sicht relevanten Aspekte und dokumentierende Einträge. Die vollständige Klassenbeschreibung findet sich in [SM98a]. Die einzige Oberklasse MODEL liefert den (definierten) Konzeptnamen MODEL auf Toplevel der rechten Seite der Konzeptdefinition. Die erste Existenzrestriktion ergibt sich aus dem ersten Eintrag, da durch das auf t (*true*) gesetzte Flag **:req** (*req* für *required*) zugesichert wird, dass es zu jeder Instanz von DEVICE-MODEL eine Instanziierung des Attributes *implementation* gibt. Der Eintrag **:dom** $\# \geq 1\{$DEVICE-INTERFACE$\}$ besagt zum einen, dass zu jeder Instanz von DEVICE-MODEL mindestens eine Instanziierung des Attributes *active-interfaces* existiert und zum anderen, dass jede Instanziierung des Attributs ein DEVICE-INTERFACE ist. Folglich wird dieser Eintrag in ein Paar aus Existenz- und Werterestriktion zum Attribut *active-interfaces* übersetzt.

Für die Untersuchung der Inferenzdienste in unserer Anwendung haben wir nach der oben skizzierten Vorgehensweise die in [SM98a] eingeführten *structural modeling objects* von VeDa, d.h. die grundlegenden Objekte zur strukturellen Beschreibung von Prozessmodellen, in eine $\mathcal{ALE}$-TBox übersetzt. Wir erhielten zunächst eine zyklische TBox. Da die Nicht-Standardinferenzdienste keine zyklischen TBoxen verarbeiten können, mussten die zyklischen Abhängigkeiten eliminiert werden. Eine genaue Untersuchung der betroffenen definierten Konzepte ergab, dass sich alle zyklischen Abhängigkeiten aus Rückverweisen der folgenden Form ergaben: im definierenden Konzept zu einer Klasse A tritt auf Toplevel eine Existenzrestriktion der Form $\exists r.B$ auf und im definierenden Konzept zu B gibt es auf Toplevel eine Existenzrestriktion der Form $\exists r'.A$. Beispielsweise erhält man zur Klasse DEVICE-IMPLEMENTATION die Konzeptdefintion

DEVICE-IMPLEMENTATION $\doteq$ MODEL-IMPLEMENTATION $\sqcap$

class: DEVICE-MODEL
documentation: “an entity which is a model of a device”
superclasses: MODEL
individual-relational-attributes:
implementation: **:dom** DEVICE-IMPLEMENTATION
:req t
:doc “the implementation of the device-model”
active-interfaces: **:dom** $\# \geq 1\{\text{DEVICE-INTERFACE}\}$
:doc “all the interfaces by means of which the device-model interacts with other models”

$$\begin{aligned}\text{DEVICE-MODEL} \doteq\ & \text{MODEL} \sqcap \\ & \exists\ \text{implementation.DEVICE-IMPLEMENTATION} \sqcap \\ & \exists\ \text{active-interfaces.DEVICE-INTERFACE} \sqcap \\ & \forall\ \text{active-interfaces.DEVICE-INTERFACE.}\end{aligned}$$

Abbildung 9.4: Beispiel für die Übersetzung von Klassenbeschreibungen in Konzeptdefinitionen.

$$\begin{aligned}& \exists\ \text{is-implementing DEVICE-MODEL} \sqcap \\ & \forall\ \text{optional-interfaces.DEVICE-INTERFACE} \sqcap \\ & \forall\ \text{required-interfaces.DEVICE-INTERFACE} \sqcap \\ & \exists\ \text{required-interfaces.DEVICE-INTERFACE} \sqcap \\ & \exists\ \text{has-Behavior.DEVICE-BEHAVIOR} \sqcap \\ & \forall\ \text{quantities.VARIABLE}\end{aligned}$$

und damit eine zyklische Abhängigkeit der oben beschriebenen Form zum definierten Konzeptnamen DEVICE-MODEL (s. Abbildung 9.4). Diese Rückverweise sind in dem Sinne redundant, dass sie bei der Entwicklung von VeDa eingeführt wurden, um Abhängigkeiten in beiden Richtungen explizit zu machen und bei einem beliebigen Einstiegspunkt in die Hierarchie alle Abhängigkeiten verfolgen zu können. Verzichtet man also auf die Rückverweise, so geht de facto keine relevante Information verloren. So konnten für den Einsatz der neuen Dienste die zyklischen Abhängigkeiten in der TBox einfach dadurch aufgelöst werden, dass die entsprechende Rückverweise aus den Konzeptbeschreibungen eliminiert wurden. Als Ergebnis erhielten wir eine azyklische $\mathcal{ALE}$-TBox mit rund 50 atomaren Konzeptnamen, 60 Rollennamen und 60 Konzeptdefinitionen. Die zugehörige Subsumtionshierarchie ist auszugsweise in Abbildung 9.5 zu sehen. Einen vollständigen Überblick findet man in [SM98a].

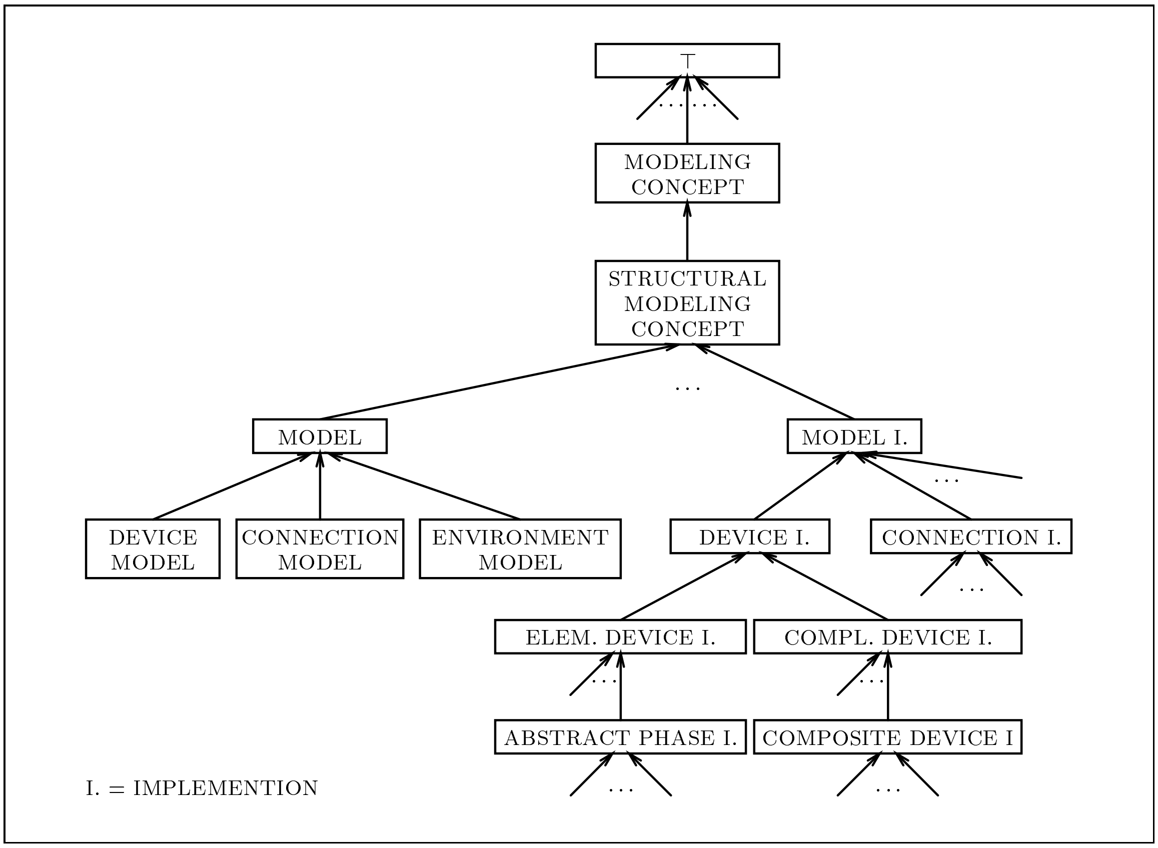


Abbildung 9.5: Ein Auszug aus der Subsumtionshierarchie zu den *structural modeling objects* von VeDa.

Ausgehend von dieser azyklischen $\mathcal{ALE}$-TBox haben wir das Verhalten der Nicht-Standardinferenzdienste untersucht. Die dabei gewonnenen Erfahrungen werden im folgenden Unterabschnitt an einem Beispiel zusammenhängend erläutert.

Zum Einsatz der Nicht-Standardinferenzdienste

Die ersten Erfahrungen mit den neuen Diensten wurden bei der Definition einer Klasse für Reaktoren mit Kühlphase gemacht. Ausgangspunkt war eine Menge von sieben Beschreibungen solcher Reaktoren. Die Beschreibungen der Reaktoren lagen in frame-basierter Darstellung vor und wurden in $\mathcal{ALE}$-Konzeptbeschreibungen $R_1, \ldots, R_7$ übersetzt.

Mit Hilfe des Objektexpolrationsalgorithmus wurde die Subsumtionshierarchie aller LCS zu Teilmengen von $\mathcal{O} := \{R_1, \ldots, R_7\}$ berechnet. Abbildung 9.6 zeigt die durch das System Anaconda [Nav96] generierte Visualisierung dieser Hierarchie. Das größte Element des Verbandes entspricht dem LCS aller 7 Reaktoren, das klein-

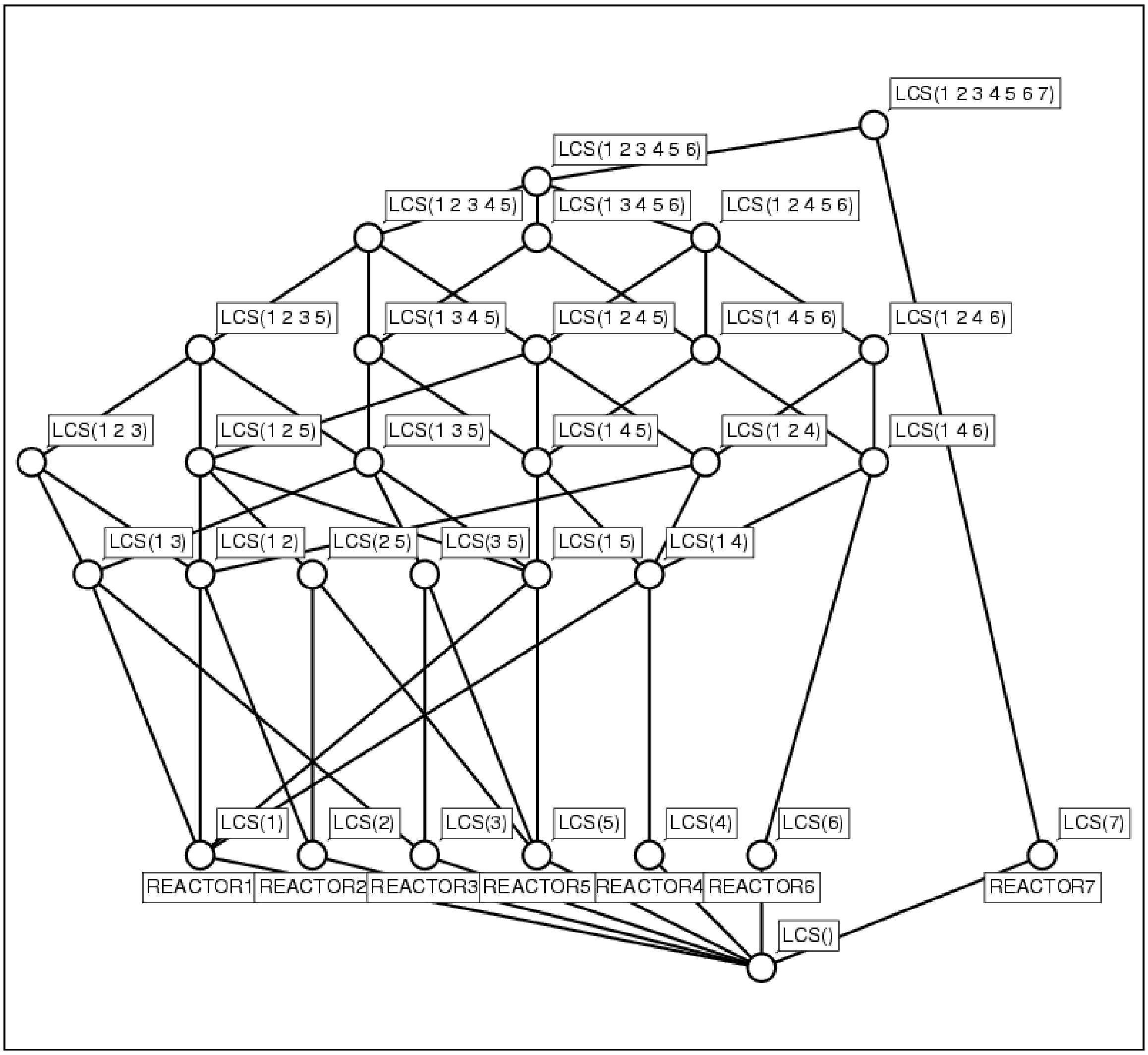


Abbildung 9.6: Die Subsumtionshierarchie der LCS zu den sieben Reaktorbeschreibungen.

ste Element dem LCS der leeren Menge, also $\perp$. Ein Knoten mit Label $\mathsf{lcs}(i_1 \dots i_m)$ entspricht dem formalen Konzept mit Extension $R_{i_1}, \dots, R_{i_m}$, also dem LCS von $R_{i_1}, \dots, R_{i_m}$. Umgekehrt ist der LCS zu einer beliebigen Menge $\{R_{i_1}, \dots, R_{i_n}\}$ gegeben durch die kleinste obere Schranke der Elemente $\mathsf{lcs}(R_{i_1}), \dots, \mathsf{lcs}(R_{i_n})$. Intuitiv entspricht diese genau dem Knoten, der (von unten nach oben den Kanten folgend) als erster Knoten über jedem der Knoten $\mathsf{lcs}(R_{i_1}), \dots, \mathsf{lcs}(R_{i_n})$ liegt. Beispielsweise ist der LCS von R_3, R_5, R_6 gleich $\mathsf{lcs}(R_1, R_3, R_4, R_5, R_6)$.

Statistische Informationen: Im Beispiel enthält die Duquennes-Guigues-Basis 15 Implikationen und der Konzeptverband 30 formale Konzepte. Um mögliche Kandidaten für eine neue Klassenbeschreibung zu bestimmen, sind von dieser Menge die

„trivialen“ LCS $\bot, R_1, \ldots, R_7$ abzuziehen. Es bleiben also 22 Kandidaten und von diesen wurden im Lauf des Algorithmus nur 11 explizit berechnet.

Der Experte benötigte 255 Subsumtionstests und 25 n-äre LCS-Operationen, die aber durch Wiederverwenden bereits berechneter LCS jeweils durch eine binäre LCS-Operation berechnet wurden. Die Zahl der Gegenbeispiele betrug ebenfalls 25.

Schließlich wurde noch die Laufzeit bestimmt, die für die einzelnen Teilaufgaben benötigt wurde, also für die LCS-Operationen, für die Subsumtionstests und für den „Overhead“ der Objektexploration (z.B. Berechnung der $\cdot''$-Operation, Berechnung der Pseudo-Hüllen, etc). Es hat sich gezeigt, dass ca. 84% der Zeit für LCS-Operationen benötigt wurden, 15% für Subsumtionstests und nur rund 1% für den Rest. Dies deutet darauf hin, dass der Explorationsalgorithmus keinen nennenswerten Mehraufwand verursacht. Zur Tatsache, dass die LCS-Operationen soviel mehr Zeit erfordern als die Subsumtionstests, sei noch angemerkt, dass der Subsumtionstest in FaCT sehr effizient implementiert ist und die LCS-Operation noch nicht optimiert wurde.

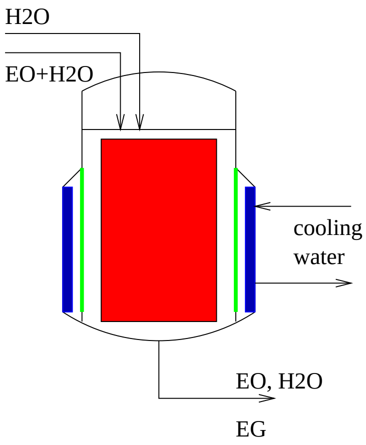
Erkenntnisse aus dem Konzeptverband: Unmittelbar lassen sich zwei Dinge ablesen:

1. Es gibt keine Subsumtionsbeziehungen zwischen den Beschreibungen der Reaktoren, da jede Einermenge als Extension eines formalen Konzeptes auftritt.
2. Beschreibung R_7 unterscheidet sich sehr stark von den übrigen Beschreibungen, da der LCS von R_7 mit irgendeiner anderen Beschreibung das größte Element des Verbandes liefert.

Aus der zweiten Beobachtung folgt, dass R_7 bei der Generierung eines Vorschlags für die neue Klassenbeschreibung nicht berücksichtigt werden sollte. Tatsächlich hat sich bei genauer Betrachtung der Beschreibung R_7 gezeigt, dass diese einen Reaktor auf einer anderen Abstraktionsebene beschreibt als die übrigen Reaktorbeschreibungen. Dieser Unterschied führt insbesondere dazu, dass der LCS von allen sieben Reaktoren ein sehr allgemeines Konzept ist, nämlich äquivalent zum bereits definierten Konzept DEVICE-MODEL.

Offensichtlich ist es nicht sinnvoll, dem Modellierer nun jedes Element aus dem Verband als gleichwertigen Kandidaten für die neue Klasse zu präsentieren. Vielmehr soll durch geeignete Kriterien eine Vorauswahl getroffen, d.h. eine kleinere Menge besonders guter Kandidaten bestimmt werden. Im Beispiel wurden hierzu die beiden folgenden Kriterien eingesetzt:

1. Kombinationen von Eingabekonzepten, die zu sehr allgemeinen Konzeptbeschreibungen (Sprüngen in der Hierarchie) führen, werden nicht berücksichtigt.
2. Ein Kandidat soll ca. 50% der Eingabekonzepte subsumieren.



$$\begin{aligned}
\mathsf{lcs}(R_1, R_4, R_5) = {} & \text{COMPOSITE-DEVICE-IMPLEMENTATION} \sqcap \\
& \exists\ \text{has-part}. \\
& \quad (\text{ABSTRACT-PHASE-IMPLEMENTATION} \sqcap \\
& \quad\ \exists\ \text{phenomena.REACTION} \sqcap \\
& \quad\ \exists\ \text{phenomena.CAPACITANCE} \sqcap \\
& \quad\ \exists\ \text{is-coupled-to}. \\
& \quad\ (\text{TWO-FILM-IMPLEMENTATION} \sqcap \\
& \quad\quad \exists\ \text{phenomena.HEAT-TRANSFER} \sqcap \\
& \quad\quad \exists\ \text{generalizedforces.TEMPERATURE-GRADIENT} \sqcap \\
& \quad\quad \exists\ \text{is-coupled-to}. \\
& \quad\quad\ (\text{ABSTRACT-PHASE-IMPLEMENTATION} \sqcap \\
& \quad\quad\quad \exists\ \text{phenomena.CAPACITANCE} \sqcap \\
& \quad\quad\quad \exists\ \text{materialstructure.HOMOGENEOUS}))).
\end{aligned}$$

Abbildung 9.7: Die graphische Repräsentation eines Reaktors mit Kühlphase und die gelernte $\mathcal{ALE}$-Konzeptbeschreibung.

Durch das erste Kriterium werden zu allgemeine Kandidaten ausgeschlossen. Im Beispiel schließt dies alle LCS zu Mengen aus, die R_7 oder R_6 enthalten.[1] Das zweite Kriterium schließt zu spezielle Kandidaten aus, die also z.B. nicht zum gewünschten Abstraktionsschritt von Bausteinen zu Klassen führen würden.

Damit erhielten wir im Beispiel 9 Kandidaten, nämlich die LCS mit Extensionen der Mächtigkeit 3, 4 und 5 aus der Menge $\{R_1, \ldots, R_5\}$. Diese wurden daraufhin untersucht, ob sie eine aussagekräftige, nachvollziehbare und lesbare Beschreibung für die neue Klasse liefern. Diese Kandidaten waren allesamt Konzeptbeschreibungen, die circa 800 atomare Konzept- und Rollennamen enthielten (ausgedruckt füllt jedes dieser Konzepte knapp sechs DIN-A4 Seiten). Die mit dem heuristischen Rewriting-Algorithmus berechneten Rewritings zu den genannten Kandidaten enthielten jeweils nur noch circa 15 Konzept- und Rollennamen. Die Größen aller berechneten LCSs lagen im Bereich von 600 bis 1000 Konzept- und Rollennamen. Die zugehörigen Rewritings kamen im Schnitt auf eine Größe von 10 bis 20 Konzept- und Rollennamen. Insbesondere trat in keinem der Rewritings das in Beispiel 8.26 skizzierte Problem auf, dass aufgrund der Greedy-Heuristik zu viele definierte Namen eingesetzt wurden, im Gegenteil: es wurde in jedem Rekursionsschritt immer nur genau ein definierter Name bestimmt, der im Rewriting eingesetzt werden konnte. In dieser Anwendung scheint also das Problem nicht aufzutreten, dass durch den heuristischen Algorithmus Rewritings berechnet werden, die aufgrund redundanter definierter Namen merklich

[1]Die Beschreibung R_6 führt zu einem weniger offensichtlichen Sprung in der Hierarchie als R_7 (s. die Kante von $\mathsf{lcs}(R_6)$ nach $\mathsf{lcs}(R_1, R_4, R_6)$ in Abbildung 9.6). Ursache für diesen Sprung ist, dass zwar nicht der gesamte Reaktor aber die Verbindung zwischen Reaktions- und Kühlphase in R_6 auf einem anderen Abstraktionsniveau beschrieben wurde als in den übrigen Beschreibungen.

größer als minimale Rewritings sind.

Durch eine Prüfung *aller* Kandidaten, d.h. aller berechneter LCSs, wurde für dieses Beispiel nachgewiesen, dass durch die Kriterien der Vorauswahl nicht versehentlich auch gute Kandidaten ausgeschlossen wurden. Aus der Bewertung der 9 (und aller) Kandidaten ging schließlich der LCS der Beschreibungen R_1, R_4 und R_5 als der beste Kandidat hervor. Abbildung 9.7 zeigt rechts die zugehörige Konzeptbeschreibung und links sieht man die graphische Repräsentation einer Instanz der neuen Klasse, eines Reaktors mit Kühlphase. Der Zusammenhang zwischen Beschreibung und Reaktor lässt sich grob wie folgt beschreiben: Der Reaktor ist ein zusammengesetztes Objekt (COMPOSITE-DEVICE-IMPLEMENTATION), der eine Reaktionsphase enthält ($\exists$ has-part.(ABSTRACT-PHASE-IMPLEMENTATION $\sqcap \ldots$)), die von einer Wand umgeben ist ($\exists$ is-coupled-to.(TWO-FILM-IMPLEMENTATION $\sqcap \ldots$)), die wiederum mit einer Kühlphase verbunden ist ($\exists$ is-coupled-to.(ABSTRACT-PHASE-IMPLEMENTATION $\sqcap \ldots$)).

Zusammenfassend haben wir insbesondere durch das oben beschriebene Beispiel den Eindruck gewonnen, dass der Einsatz des BL-Systems FaCT und der Nicht-Standardinferenzen eine sinnvolle Unterstützung bei der Strukturierung und Verwaltung der Wissensbasis bieten kann. Dabei liefern die neuen Dienste Beschreibungen, die für die Modellierer lesbar und verständlich sind und somit als Ausgangspunkt für die Definition neuer Klassen dienen können. Dadurch wird die Pflege der Wissensbasis, genauer ihrer Struktur, bei zunehmender Anzahl von Klassen und Bausteinen erleichtert, was wiederum eine effektivere Wiederverwendung von Bausteinen bei der Prozessmodellierung ermöglicht.

Kapitel 10

Zusammenfassung

Das Ziel der Kooperation zwischen dem Lehr- und Forschungsgebiet Theoretische Informatik und dem Lehrstuhl für Prozesstechnik besteht darin, die Einsetzbarkeit von BL-Systemen und ihrer Standard- sowie Nicht-Standardinferenzdiensten zur Unterstützung der Strukturierung und Pflege der Wissensbasis in der von den Prozesstechnikern entwickelten Modellierungsumgebung ModKit zu untersuchen.

Im Rahmen des dieser Arbeit zugrundeliegenden Forschungsprojektes lag das Hauptaugenmerk auf der Entwicklung und Umsetzung eines Szenarios zur Unterstützung der Erweiterung der Wissensbasis, genauer: zur Unterstützung der Definition neuer Klassen von Bausteinen. Die entwickelten Szenarien wurden ausführlich in Kapitel 4 vorgestellt. Sie beschreiben das Zusammenspiel der Nicht-Standardinferenzen MSC, LCS und Rewriting bei der Definition einer neuen Klasse bzw. der zugehörigen Konzeptbeschreibung. Für die formale Untersuchung dieser drei Inferenzen bzgl. Existenz, Berechenbarkeit und Komplexität wurde als Zielsprache die BL $\mathcal{ALE}$ gewählt, da in dieser BL wesentliche Aspekte der Klassen- und Bausteinbeschreibungen ausgedrückt werden können. Darüberhinaus wurde argumentiert, dass die Berechnung des LCS zu einer Menge von Konzeptbeschreibungen $\{C_1, \ldots, C_n\}$ im allgemeinen keine ausreichende Unterstützung bei der Erweiterung der Wissensbasis liefert. Vielmehr ist es häufig nützlicher, dem Modellierer die Subsumtionshierarchie aller LCS zu allen Teilmengen von $\{C_1, \ldots, C_n\}$ zur Auswahl geeigneter Kandidaten für neue Klassen anzubieten.

In Kapitel 5 wurde Existenz und Berechenbarkeit des LCS für n $\mathcal{ALE}$-Konzeptbeschreibungen nachgewiesen. Die Basis für diesen Nachweis bildete die Charakterisierung der Subsumtion in $\mathcal{ALE}$ durch Homomorphismen zwischen $\mathcal{ALE}$-Beschreibungsbäumen. Darüberhinaus wurde gezeigt, dass bereits der LCS von zwei $\mathcal{ALE}$-Konzeptbeschreibungen exponentiell groß in der Größe der Ausgangskonzepte sein kann. In Kapitel 6 wurde der Objektexplorationsalgorithmus aus der Formalen Begriffsanalyse vorgestellt, der dafür bekannt ist, dass er in vielen praktischen Anwendungen eine effiziente Berechnung des Konzeptverbandes zu einem formalen Kontext erlaubt. Um diesen Algorithmus für eine effiziente Berechnung der Subsumtionshierarchie der LCS zu allen Teilmengen einer Menge $\{C_1, \ldots, C_n\}$ von Konzeptbeschreibungen einzusetzen, wurde zur Menge $\{C_1, \ldots, C_n\}$ ein formaler Kontext definiert und für dessen Konzeptverband nachgewiesen, dass er isomorph zur gesuchten LCS-Hierarchie ist.

Wie in Kapitel 7 gezeigt existiert das MSC zu einem Individuum a bzgl. einer $\mathcal{ALE}$-ABox im allgemeinen nicht. Um dennoch eine möglichst spezielle Beschreibung zu a zu erhalten, wurde der Begriff der k-Approximation eingeführt. Leider gelang der Nachweis der Existenz und Berechenbarkeit einer solchen k-Approximation für $\mathcal{ALE}$ nur für endliche Signaturen und mit Hilfe eines inpraktikablen Algorithmus. Nur für die Teilsprachen $\mathcal{EL}$ und $\mathcal{EL}_\neg$ konnte eine Charakterisierung der k-Approximation gefunden werden, die die Entwicklung eines praktikablen Algorithmus erlaubt. Grundlage hierfür war jeweils die Entwicklung einer Charakterisierung der Instanz durch Homomorphismen zwischen Beschreibungsbäumen und Beschreibungsgraphen. Aus der Charakterisierung der k-Approximation wurde für $\mathcal{EL}$ und $\mathcal{EL}_\neg$ auch jeweils eine Charakterisierung des MSC für *azyklische* Aboxen abgeleitet. Sowohl für $\mathcal{ALE}$ als auch die Teilsprachen wurde schließlich nachgewiesen, dass das MSC und die k-Approximation exponentiell groß in der Größe der ABox (und k) sein kann.

Die Untersuchung des minimalen Rewriting Problems in Kapitel 8 wurde in zwei Teile geteilt. Zunächst wurde die Komplexität des induzierten Entscheidungsproblems für die BLen $\mathcal{FL}_0$, $\mathcal{ALN}$, $\mathcal{ALE}$ und $\mathcal{ALC}$ untersucht. Der zweite Teil konzentrierte sich auf das eigentliche Berechnungsproblem für $\mathcal{ALE}$. Aufbauend auf den zu diesem Zweck eingeführten Begriffen Extension und Reduktion bzgl. einer TBox wurde ein nichtdeterministischer Algorithmus entwickelt, der die Berechnung eines bzw. aller minimalen Rewritings zu einer $\mathcal{ALE}$-Konzeptbeschreibung bzgl. einer $\mathcal{ALE}$-TBox erlaubt. Es wurde gezeigt, dass sich ein minimales Rewriting mit polynomiellem Platzaufwand berechnen lässt, die möglicherweise exponentiell große Menge aller minimalen Rewritings aber nur mit exponentiellem Zeitaufwand. Aufbauend auf dem nichtdeterministischen Algorithmus wurde schließlich noch ein Algorithmus entwickelt, der durch den Einsatz einer Greedy-Heuristik die Berechnung (nicht notwendig minimaler) Rewritings in deterministisch polynomieller Zeit mit einem Orakel für Subsumtion modulo TBox in $\mathcal{ALE}$ erlaubt.

Im Anschluss an die technischen Kapitel wurde in Kapitel 9 die prototypische Implementierung der entwickelten Algorithmen sowie die Anbindung des derartig erweiterten BL-Systems FaCT an die Modellierungsumgebung ModKit beschrieben. Die bei der Umsetzung der entwickelten Szenarien in der Anwendung gemachten positiven Erfahrungen wurden dann anhand eines Beispiels dargestellt.

Zum Abschluss dieser inhaltlichen Zusammenfassung sei zu den bisher erzielten Resultaten für Nicht-Standardinferenzen noch angemerkt, dass insgesamt—gerade auch durch die am Lehr- und Forschungsgebiet Theoretische Informatik erzielten Ergebnisse [BN98, BK98, BKM99a, BKBM99, BK00a, BKM00, BM00, Küs00]—ein Stand der Forschung erreicht ist, der bereits über die Ergebnisse zu den Standardinferenzen in der zweiten Phase hinausgeht (vgl. die historische Einteilung in vier Phasen aus Abschnitt 3.1).

Offene Probleme

Als das am Ende dieser Arbeit wesentliche offene Problem bzgl. der gemäß den Szenarien aus Kapitel 4 für $\mathcal{ALE}$ zu untersuchenden Nicht-Standardinferenzen sei hier das MSC in $\mathcal{ALE}$ bzw. in den Subsprachen $\mathcal{EL}$ und $\mathcal{EL}_\neg$ erwähnt. Wünschenswert ist eine Charakterisierung des MSC durch zyklische Konzeptbeschreibungen, wie sie in [BK98] für $\mathcal{ALN}$ entwickelt wurde. Dies wiederum erfordert zunächst geeignete Charakterisierungen der Semantik zyklischer $\mathcal{ALE}$-Konzeptbeschreibungen sowie der Instanz und Subsumtion modulo zyklischer $\mathcal{ALE}$-TBoxen. Gelingt der Nachweis von Existenz und Berechenbarkeit des MSC für zyklische Konzeptbeschreibungen, so wären für die Umsetzung der Szenarien auch die Nicht-Standardinferenzen LCS und Rewriting für zyklische $\mathcal{ALE}$-Konzeptbeschreibungen als offene Probleme zu betrachten. Zu diesen Problemen liegen aber noch keine vielversprechenden Ansätze aus der Literatur oder angrenzenden Forschungsbereichen vor.

Ausblick

Im folgenden werden Ansätze und Ideen für weitere Anwendungsszenarien beschrieben, die bei der Fortführung unserer Kooperation im Rahmen des DFG-Forschungsprojektes „Neuartige Schlussfolgerungsverfahren zur Unterstützung des Aufbaus und der Wartung von Wissensbasen in Beschreibungslogiken“ in den kommenden Jahren verfolgt werden sollen.

Erweiterung der Ergebnisse

Hierzu zählt zum einen die Optimierung der Implementierung der bisher entwickelten Algorithmen sowie die Anbindung des BL-Systems an ModKit mit CORBA. Ziel ist es, die Durchführung von Fallstudien zu erleichtern, um so einen fundierteren Eindruck von den Möglichkeiten und den Grenzen der Einsetzbarkeit von BL-Systemen mit Nicht-Standardinferenzen in der Anwendung zu erhalten.

Zum anderen wird eine Erweiterung der theoretischen Ergebnisse auf ausdrucksstärkere BLen angestrebt. Besonders interessant wäre hier die Untersuchung des LCS für die BLen $\mathcal{ALEN}$ und $\mathcal{ALQ}$,[1] da diese BLen speziellere Übersetzungen von Klassenbeschreibungen zulassen würden. Beispielsweise kann der Frame-Eintrag

$$\textit{device-models}\ \mathbf{:dom}\ \#\geq 2\{\text{DEVICE-MODEL}\}\ \mathbf{:req}\ t$$

[1] $\mathcal{ALEN}$ erweitert $\mathcal{ALE}$ um Zahlenrestriktionen. $\mathcal{ALQ}$ erlaubt neben Konjunktion, primitiver Negation und Werterestriktionen noch sogenannte *qualifizierte Zahlenrestriktionen* der Form $(\geq n\ r\ C), (\leq n\ r\ C)$ (s. z.B. [HB91] für eine formale Definition von Syntax und Semantik dieses Konstruktors).

aus der VeDa-Klassenbeschreibung zu COMPOSITE-DEVICE-IMPLEMENTATION in die $\mathcal{ALQ}$-Konzeptbeschreibung

$$(\geq 2 \text{ device-models DEVICE-MODEL})$$

übersetzt werden, der in $\mathcal{ALEN}$ bzw. $\mathcal{ALE}$ durch die allgemeinere Konzeptbeschreibung

$$\exists \text{ device-models.DEVICE-MODEL} \sqcap (\geq 2 \text{ device-models})$$

bzw.

$$\exists \text{ device-models.DEVICE-MODEL}$$

beschrieben wird. Man beachte, dass $\mathcal{ALEN}$ eine Teilsprache von $\mathcal{ALQ}$ ist, da $\exists r.C$ äquivalent zu $(\geq 1\ r\ C)$ ist. Zunächst $\mathcal{ALEN}$ zu betrachten macht jedoch Sinn, um so einen Eindruck und erste Lösungsansätze von den insgesamt in $\mathcal{ALQ}$ zu lösenden Problemen bei der Berechnung des LCS zu erhalten. In [Man99] finden sich erste Ansätze für die Charakterisierung des LCS in diesen BLen, jedoch sind die vorgestellten Algorithmen nicht vollständig und die Beweise zu Existenz und Berechenbarkeit sind noch lückenhaft.

Anwendungsszenario zum Entdecken und Vermeiden von Redundanzen

Die Wissensbasis von ModKit wird mittlerweile seit einigen Jahren von mehreren Modellierern aufgebaut und ständig erweitert. Hieraus ergibt sich das Problem, dass von Zeit zu Zeit (und bei zunehmender Größe der Wissensbasis immer häufiger) redundante Klassen in die Wissensbasis eingefügt werden, d.h., intuitiv gleiche Konzepte treten unter verschiedenen Namen oder Definitionen mehrfach in der Wissensbasis auf. Zur Lösung dieses Problems ist ein Systemdienst zum *Entdecken und Vermeiden von Redundanzen* wünschenswert, d.h. ein Dienst, der zu einer gegebenen Klassenbeschreibung prüft, ob bereits eine äquivalente Klasse in der Wissensbasis existiert.

Intuitiv würde man versuchen, einen solchen Dienst unter Verwendung des Äquivalenztests des BL-Systems zu realisieren: die Klassenbeschreibung der neu einzufügenden Klasse wird in eine Konzeptbeschreibung übersetzt und in die bereits existierende Hierarchie eingeordnet. Ist sie äquivalent zu einem bereits definierten Konzept, so ist die neue Klasse redundant und sollte nicht eingefügt werden. Andernfalls wird sie in die Wissensbasis aufgenommen.

Dieser Ansatz ist jedoch nicht immer ausreichend [Küs00, BN98]. Die Ursache hierfür ist, dass in der neuen und den existierenden Beschreibungen möglicherweise für ein atomares Konzept verschiedene Namen verwendet werden oder sie auf unterschiedlichen Abstraktionsniveaus beschrieben werden (s. z.B. die Einleitung von [BK00a] für eine ausführlichere Beschreibung des Problems). In verschiedenen Arbeiten [BKBM99,

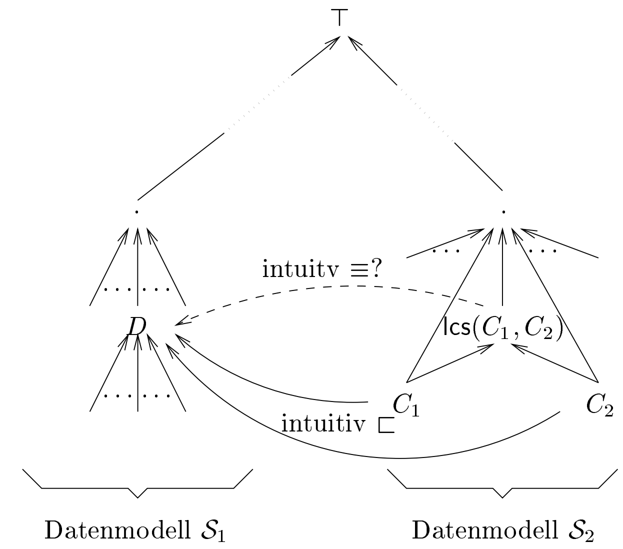


Abbildung 10.1: Idee eines Szenarios zur Unterstützung der Integration von Datenmodellen.

BK00a, Küs00] wird der folgende auf der Nicht-Standardinferenz *Matching von Konzeptbeschreibungen* basierende Lösungsansatz vorgestellt: Angenommen, ein Modellierer möchte eine neue Konzeptbeschreibung in die Wissensbasis einfügen, ist sich aber nicht sicher, ob für Teile der Konzeptbeschreibung bereits eine und wenn ja welche Beschreibung vorliegt. Dann hat er die Möglichkeit, ein sogenanntes *Konzeptpattern* zu definieren, in dem diese Teile durch sogenannte *Konzeptvariablen* beschrieben sind. Ein Matching-Algorithmus sucht dann eine Substitution der Variablen, sodass man eine Konzeptbeschreibung erhält, die äquivalent zu einer bereits existierenden Konzeptbeschreibung ist. Gibt es eine solche Substitution, so erhält der Modellierer zum einen die möglichen Beschreibungen der durch Variablen ersetzten Teile der Konzeptbeschreibung und weiß zum anderen, dass das neue Konzept redundant wäre, es also nicht eingefügt werden sollte.
Umfassende theoretische Resultate zum Matching in Classic [BKBM99] und in $\mathcal{ALE}$ [BK00a, Küs00] liegen bereits vor. Da die insbesondere für $\mathcal{ALE}$ entwickelten Algorithmen sehr komplex sind, wäre hier jedoch zunächst zu untersuchen, ob sich das skizzierte Szenario mit diesen Algorithmen umsetzen lässt oder ob alternative Verfahren für den Einsatz in der Anwendung entwickelt werden müssen.

Anwendungsszenario zur Integration von Datenmodellen

In [BSM00] argumentieren die Autoren, dass in den vergangenen Jahren viele rechnergestützte Ansätze zur Unterstützung der Modellierung chemischer Prozesse vorgestellt worden sind, eine signifikante Verbesserung der Unterstützung aber nur noch erreicht werden kann, wenn der Modellierungsprozess in seiner Gesamtheit unterstützt wird. Dies wiederum erfordert die Integration von Software-Werkzeugen in einer Entwicklungsumgebung. Da jedoch die Software-Werkzeuge auf verschiedene Datenmodelle aufsetzen, die die Daten der jeweils unterstützten Teilprozesse modellieren, ist intuitiv klar, dass für eine erfolgreiche Integration der Werkzeuge ein automatischer Datenaustausch zu gewährleisten ist. Dieser wiederum erfordert die *Integration von Datenmodellen.*

Für dieses insbesondere aus dem Datenbank- und Data Warehouse-Bereich bekannte Problem wurden bereits viele Ansätze und Techniken entwickelt, die sich nach [CdGL+99] in zwei Kategorien einteilen lassen: (1) Techniken zur Konstruktion *eines* integrierten Datenmodells und (2) Techniken zur Definition von Transformationen zwischen verschiedenen Datenmodellen.

Der bekannteste Ansatz zur Unterstützung der Integration von Datenmodellen durch BLen ist der sogenannte *assertionale Ansatz* [BC86, CL93, CGL+98b, CdGL+99]. Hierbei werden die zu integrierenden Datenmodelle in (hinreichend ausdrucksstarke) BLen übersetzt und zu einem integrierten Datenmodell vereinigt, wobei zusätzlich Beziehungen zwischen den Datenmodellen durch sogenannte *integration assertions* in der erhaltenen Terminologie formuliert werden. Diese Assertionen reichen von Namensäquivalenzen der Form $A \equiv B$ mit Namen A, B aus verschiedenen Datenmodellen über Abbildungen $A \equiv C$ von Namen eines Datenmodells auf komplexe Beschreibungen C in einem anderen Datenmodell bis hin zu generellen Inklusionsbeziehungen $C \sqsubseteq D$ zwischen komplexen Beschreibungen C, D verschiedener Datenmodelle. Standardinferenzdienste erlauben in diesem Ansatz die automatische Konsistenzprüfung und Berechnung von Subsumtionsbeziehungen innerhalb des erhaltenen Datenmodells.

In [BK00b] stellen die Autoren ein Rahmenwerk vor, in dem Matching und Unifikation von Konzeptbeschreibungen eingesetzt werden kann, um geeignete Abbildungen zwischen Namen und Konzeptbeschreibungen verschiedener Datenmodelle zu berechnen. Dabei wird zum einen versucht, den Begriff 'geeignete Abbildung' formal zu fassen. Zum anderen wird gezeigt, dass der Nutzung struktureller Informationen, d.h. von Subsumtionsbeziehungen innerhalb der verschiedenen Datenmodelle, bei der Suche nach solchen Abbildungen Grenzen gesetzt sind.

Zu untersuchen, inwiefern sich diese beiden Ansätze bei der Integration der Datenmodelle in der Prozesstechnik einsetzen lassen, könnte Gegenstand künftiger gemeinsamer Arbeiten sein. Ausgehend von einer Übersetzung der Datenmodelle und insbesondere des integrierten Modells in eine BL-Terminologie ist außerdem eine Unterstützung durch die in dieser Arbeit untersuchten Nicht-Standardinferenzen denkbar: Dem ersten Szenario aus Abschnitt 4.2 folgend, lassen sich beispielsweise LCS und Rewriting einsetzen, um die Struktur des integrierten Datenmodells zu verfeinern. Darüberhinaus ist das Folgende eine grobe Idee eines Szenarios zur Unterstützung der händischen Integration von Datenmodellen und insbesondere der Definition der oben erwähnten Assertionen: Angenommen, dem Modellierer liegen zwei Datenmodelle $\mathcal{S}_1$ und $\mathcal{S}_2$ in Form einer vereinigten BL-Terminologie vor und er findet zwei Konzepte C_1, C_2 des einen Modells, die intuitiv Unterkonzepte eines Konzeptes D im anderen Modell sind, wobei keines dieser Konzepte intuitiv gleich mit einem Konzept des jeweils anderen Datenmodells ist (s. Abbildung 10.1). Die Berechnung des LCS von C_1, C_2 liefert nun ein Konzept C, das Oberkonzept von C_1 und C_2 ist und damit auch intuitiv ähnlicher zu D. Erkennt der Modellierer sogar eine (intuitive) Spezialisierungs- oder Äquivalenzbeziehung zwischen C und D, kann er die entsprechende Assertion definieren. Man beachte, dass aufgrund der unterschiedlichen Namen in den verschiedenen Datenmodellen ein Subsumtionstest zwischen C und D im allgemeinen keinen Sinn macht (vgl. auch die Motivation für das Matching von Konzeptbeschreibungen), sodass hier der Modellierer mit seinem Expertenwissen die Spezialisierungsbeziehung erkennen und als Assertion von Hand festschreiben muss. Dieses Szenario könnte in künftigen Arbeiten zunächst genauer spezifiziert und dann bzgl. seiner Anwendbarkeit in der Prozesstechnik untersucht werden.

Fazit

Insgesamt ist der Einsatz von BLen in der Prozesstechnik, wie er im Rahmen der Kooperation zwischen dem Lehr- und Forschungsgebiet Theoretische Informatik und dem Lehrstuhl für Prozesstechnik betrieben wird, positiv zu bewerten.

So sind einerseits für die Unterstützung der Prozessmodellierung BL-Systeme und neue Dienste als Mittel zur Strukturierung und Pflege der Wissensbasis bereit gestellt worden. Grundlage hierfür war der Nachweis, dass relevante Aspekte der betrachteten Klassen und Bausteine in einer BL-Wissensbasis repräsentiert werden können und die Inferenzdienste für die Modellierer verwertbare Ergebnisse wie z.B. die Spezialisierungshierarchie liefern.

Umgekehrt ist für den Bereich BL die Prozesstechnik als ein komplexer, technischer Anwendungsbereich erschlossen worden. Zum einen sind, motiviert durch die dort vorgefundenen Anforderungen, Ausdrucksstärke und Komplexität von BLen sowie Standard- und Nicht-Standardinferenzdienste formal untersucht worden, wodurch insgesamt ein tieferes Verständnis dieser Familie von Wissensrepräsentationsformalismen sowie seiner Einsetzbarkeit geschaffen worden ist. Zum anderen konnte auch gezeigt werden, dass BL-Systeme in solch komplexen Anwendungsszenarien sinnvoll eingesetzt werden können.

Danksagung

Diese Arbeit entstand in der Zeit von Oktober 1997 bis September 2000. Sie zu erstellen wurde mir durch ein Stipendium der Deutschen Forschungsgemeinschaft ermöglicht, dass ich als Mitglied des Graduiertenkollegs „Informatik & Technik“ an der RWTH Aachen erhielt.

Mein fachlicher Dank gilt Prof. Franz Baader und Prof. Wolfgang Marquardt, die als Betreuer nicht nur Ziele und Rahmen des dieser Arbeit zugrundeliegenden Forschungsvorhabens festgelegt, sondern mich auch stets bei der fokussierten und zügigen Umsetzung des Vorhabens unterstützt haben. Darüberhinaus danke ich Prof. Baader sowie Ralf Küsters für die vielen fruchtbaren Diskussionen und die konstruktive Kritik bei unserer wissenschaftlichen Zusammenarbeit. Namentlich möchte ich auch noch Dr. Ian Horrocks von der University of Manchester und Lars von Wedel vom Lehrstuhl für Prozesstechnik für ihre Erläuterungen zu FaCT, VeDa und ModKit und für ihre Unterstützung bei den Implementierungsarbeiten danken.

Mein persönlicher Dank gilt zunächst meinen Eltern, die nicht nur durch die Finanzierung meines Studiums den Grundstein für diese Arbeit legten, sondern mir auch stets Halt und Kraft gegeben haben. Meinen Geschwistern, Freunden und Kollegen danke ich für die vielen gemeinsam verbrachten, entspannenden und motivierenden, fröhlichen und ernsten Stunden, die mich insbesondere in schwierigen Phasen immer wieder aufgebaut haben.

Literaturverzeichnis

[AHU87] A.V. Aho, J.E. Hopcroft und J.D. Ullman: *Data Structures and Algorithms.* Computer Science and Information Processing. Addison Wesley, 1987.

[AHV95] S. Abiteboul, R. Hull und V. Vianu: *Foundations of Databases.* Addison-Wesley, 1995.

[Baa91] F. Baader: Augmenting Concept Languages by Transitive Closure of Roles: An Alternative to Terminological Cycles. In: J. Mylopoulos und R. Reiter (Herausgeber): *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI'91)*, Seiten 446–451. Morgan Kaufmann, 1991.

[Baa95] F. Baader: Computing a Minimal Representation of the Subsumption Lattice of all Conjunctions of Concepts Defined in a Terminology. In: *Proceedings of the International Symposium on Knowledge Retrieval, Use, and Storage for Efficiency (KRUSE'95)*, Seiten 168–178, 1995.

[Baa96] F. Baader: Using Automata Theory for Characterizing the Semantics of Terminological Cycles. *Annals of Mathematic and Artificial Intelligence*, 18:175–219, 1996.

[Bau00] M. Baumeister: *Ein flexibles und abstrahierendes Objektmodell für die Modellierung chemischer Prozesse.* Doktorarbeit, RWTH Aachen, 2000.

[BBH+91] F. Baader, H.-J. Bürkert, J. Heinsohn, B. Hollunder, J. Müller, B. Nebel, W. Nutt und H.-J Profitlich: Terminological Knowledge Representation: A Proposal for a Terminological Logic. Technischer Bericht TM-90-04, Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Kaiserslautern, Germany, 1991.

[BBH96] F. Baader, M. Buchheit und B. Hollunder: Cardinality restrictions on concepts. *Artificial Intelligence*, 88(1–2):195–213, 1996.

[BBN+93] F. Baader, H.-J. Bürckert, B. Nebel, W. Nutt und G. Smolka: On the Expressivity of Feature Logics with Negation, Functional Uncertainty, and Sort Equations. *Journal of Logic, Language and Information*, 2:1–18, 1993.

[BC86] J. Biskup und B. Convent: A Formal View Integration Method. In: C. Zaniolo (Herausgeber): *Proceedings of the 1986 ACM SIGMOD International Conference on Management of Data*, Seiten 398–407. ACM Press, 1986.

[BFH+94] F. Baader, E. Franconi, B. Hollunder, B. Nebel und H.J. Profitlich: An Empirical Analysis of Optimization Techniques for Terminological Representation Systems, or: Making KRIS get a move on. *Applied Artificial Intelligence*, 4:109–132, 1994.

[BFL83] R. J. Brachman, R. Fikes und H. J. Levesque: Krypton: A Functional Approach to Knowledge Representation. *IEEE Computer*, 16(10):67–73, 1983.

[BFT95] P. Bresciani, E. Franconi und S. Tessaris: Implementing and testing expressive Description Logics: a preliminary report. In: *Proceedings of the International Symposium on Knowledge Retrieval, Use, and Storage for Efficiency (KRUSE'95)*, 1995.

[BH91a] F. Baader und P. Hanschke: A Schema for Integrating Concrete Domains into Concept Languages. In: J. Mylopoulos und R. Reiter (Herausgeber): *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI'91)*, Seiten 452–457. Morgan Kaufmann, 1991.

[BH91b] F. Baader und B. Hollunder: A Terminological Knowledge Representation System with Complete Inference Algorithm. In: H. Boley und M.M. Richter (Herausgeber): *Proceedings of the Workshop on Processing Declarative Knowledge, PDK-91*, Band 567 der Reihe *Lecture Notes in Artificial Intelligence*, Seiten 67–86. Springer-Verlag, 1991.

[BHPST99] S. Bechhofer, I. Horrocks, P. F. Patel-Schneider und S. Tessaris: A Proposal for a Description Logic Interface. In: P. Lambrix, A. Borgida, M. Lenzerini, R. Möller und P. Patel-Schneider (Herausgeber): *Proceedings of the 1999 International Workshop on Description Logics (DL'99)*, Nummer 22 in *CEUR-WS*, Seiten 33–36, 1999. Proceedings online available from http://SunSITE.Informatik.RWTH-Aachen.DE/Publications/CEUR-WS/Vol-22/.

[BK98] F. Baader und R. Küsters: Computing the Least Common Subsumer and the Most Specific Concept in the Presence of Cyclic $\mathcal{ALN}$-Concept

Descriptions. In: O. Herzog und A. Günter (Herausgeber): *Proceedings of the 22nd Annual German Conference on Artificial Intelligence (KI'98)*, Band 1504 der Reihe *Lecture Notes in Computer Science*, Seiten 129–140. Springer-Verlag, 1998.

[BK00a] F. Baader und R. Küsters: Matching in Description Logics with Existential Restrictions. In: A.G. Cohn, F. Giunchiglia und B. Selman (Herausgeber): *Principles of Knowledge Representation and Reasoning: Proceedings of the 7th International Conference (KR2000)*, Seiten 261–272. Morgan Kaufmann, 2000.

[BK00b] A. Borgida und R. Küsters: What's not in a Name? Initial Explorations of a Structural Approach to Integrating Large Concept Knowledge Bases. In: F. Baader und U. Sattler (Herausgeber): *Proceedings of the 2000 International Workshop on Description Logics (DL2000)*, Band 33 der Reihe *CEUR-WS*. Verlag Mainz, 2000. Proceedings online available from http://SunSite.Informatik.RWTH-Aachen.de/Publications/CEUR-WS/Vol-33/.

[BKBM99] F. Baader, R. Küsters, A. Borgida und D. McGuinness: Matching in Description Logics. *Journal of Logic and Computation*, 9(3):411–447, 1999.

[BKM99a] F. Baader, R. Küsters und R. Molitor: Computing Least Common Subsumers in Description Logics with Existential Restrictions. In: T. Dean (Herausgeber): *Proceedings of the 16th International Joint Conference on Artificial Intelligence 1999 (IJCAI'99)*, Seiten 96–101. Morgan Kaufmann, 1999.

[BKM99b] F. Baader, R. Küsters und R. Molitor: Rewriting Concepts Using Terminologies – Revisited. LTCS-Report 99-12, LuFG Theoretische Informatik, RWTH Aachen, 1999. Siehe http://www-lti.informatik.rwth-aachen.de/Forschung/Reports.html.

[BKM00] F. Baader, R. Küsters und R. Molitor: Rewriting Concepts Using Terminologies. In: A.G. Cohn, F. Giunchiglia und B. Selman (Herausgeber): *Principles of Knowledge Representation and Reasoning: Proceedings of the 7th International Conference (KR2000)*, Seiten 297–308. Morgan Kaufmann, 2000.

[BL84] R. Brachman und H.J. Levesque: The Tractability of Subsumption in Frame-based Description Languages. In: *Proceedings of the 4th National Conference on Artificial Intelligence (AAAI'84)*, Seiten 34–37. ACM Press, 1984.

[BLM96] R. Bogusch, B. Lohmann und W. Marquardt: Computer-Aided Process Modeling with ModKit. In: *Conference Proceedings of Chemputers Europe III*, 1996.

[BLM97] R. Bogusch, B. Lohmann und W. Marquardt: Ein System zur rechnergestützten Modellierung in der Verfahrenstechnik. In: *Jahrbuch 1997 der VDI-Gesellschaft Verfahrenstechnik und Chemieingenieurwesen*, Seiten 22–53. VDI-Verlag, 1997.

[BLM00] R. Bogusch, B. Lohmann und W. Marquardt: Computer-aided Process Modeling with ModKit. Submitted to Computers & Chemical Engineering, 2000.

[BLR97] C. Beeri, A.Y. Levy und M.-C. Rousset: Rewriting queries using views in description logics. In: L. Yuan (Herausgeber): *Proceedings of the 16th ACM SIG-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'97)*, Seiten 99–108. ACM Press, 1997.

[BM96] A. Borgida und D.L. McGuinness: Asking Queries about Frames. In: L.C. Aiello, J. Doyle und S. Shapiro (Herausgeber): *Principles of Knowledge Representation and Reasoning: Proceedings of the 5th International Conference (KR'96)*, Seiten 340–349. Morgan Kaufmann, 1996.

[BM98a] M. Baumeister und W. Marquardt: The Chemical Engineering Data Model VeDa. Part 1: VDDL: The Language Definition. Interner Bericht LPT-1998-01, Lehrstuhl für Prozesstechnik, RWTH Aachen, 1998. Siehe http://www.lfpt.rwth-aachen.de/Publication.

[BM98b] R. Bogusch und W. Marquardt: The Chemical Engineering Data Model VeDa. Part 4: Behavioral Modeling Objects. Interner Bericht LPT-1998-04, Lehrstuhl für Prozesstechnik, RWTH Aachen, 1998. Siehe http://www.lfpt.rwth-aachen.de/Publication.

[BM00] F. Baader und R. Molitor: Building and Structuring Description Logic Knowledge Bases Using Least Common Subsumers and Concept Analysis. In: B. Ganter und G. Mineau (Herausgeber): *Proceedings of the 8th International Conference on Conceptual Structures (ICCS2000)*, Band 1867 der Reihe *Lecture Notes in Artificial Intelligence*, Seiten 292–305. Springer-Verlag, 2000.

[BMPS+91] R.J. Brachman, D. McGuinness, P. Patel-Schneider, L. Resnick und A. Borgida: Living with CLASSIC: When and how to use a KL-ONE-like language. In: John F. Sowa (Herausgeber): *Principles of Semantic Networks*. Morgan Kaufmann, 1991.

[BMT98] F. Baader, R. Molitor und S. Tobies: On the Relation between Descripion Logics and Conceptual Graphs. LTCS-Report 98-11, LuFG Theoretische Informatik, RWTH Aachen, 1998. Siehe http://www-lti.informatik.rwth-aachen.de/Forschung/Reports.html.

[BMT99] F. Baader, R. Molitor und S. Tobies: Tractable and Decidable Fragments of Conceptual Graphs. In: W. Tepfenhart (Herausgeber): *Proceedings of the 7th International Conference on Conceptual Structures (ICCS'99)*, Band 1640 der Reihe *Lecture Notes in Computer Science*, Seiten 480–493. Springer-Verlag, 1999.

[BN98] F. Baader und P. Narendran: Unification of Concept Terms in Description Logics. In: H. Prade (Herausgeber): *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI-98)*, Seiten 331–335. John Wiley & Sons Ltd, 1998.

[Bor96] A. Borgida: On the Relative Expressiveness of Description Logics and Predicate Logics. *Artificial Intelligence*, 82(1-2):353–367, 1996.

[BP94] A. Borgida und P. Patel-Schneider: A Semantics and Complete Algorithm for Subsumption in the CLASSIC Description Logic. *Journal of Artificial Intelligence Research*, 1:277–308, 1994.

[Bra77] R.J. Brachman: What's in a Concept: Structural Foundations for Semantic Networks. *International Journal of Man-Machine Studies*, 9(2):127–152, 1977.

[Bra78] R.J. Brachman: Structured inheritance networks. In: W.A. Woods und R.J. Brachman (Herausgeber): *Research in Natural Language Understanding*, Quarterly Progress Report No. 1, BBN Report No. 3742, Seiten 36–78. Bolt, Beranek and Newman Inc., 1978.

[Bra83] R.J. Brachman: What IS-A is and isn't: An Analysis of Taxonomic Links in Semantic Networks. *IEEE Computer*, 16(10):30–36, 1983.

[BS85] R.J. Brachman und J. Schmolze: An Overview of the KL-ONE Knowledge Representation System. *Cognitive Science*, 9(2):171–216, 1985.

[BS96a] F. Baader und U. Sattler: Description Logics with Symbolic Number Restrictions. In: W. Wahlster (Herausgeber): *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI'96)*, Seiten 283–287. John Wiley & Sons Ltd, 1996.

[BS96b] F. Baader und U. Sattler: Knowledge Representation in Process Engineering. In: L. Padgham, E. Franconi, M. Gehrke, D. McGuinness und

P. Patel-Schneider (Herausgeber): *Proceedings of the 1996 International Workshop on Description Logic (DL'96)*, Seiten 74–78. AAAI Press, 1996.

[BS96c] F. Baader und U. Sattler: Number Restrictions on Complex Roles in Description Logics. In: L.C. Aiello, J. Doyle und S. Shapiro (Herausgeber): *Principles of Knowledge Representation and Reasoning: Proceedings of the 5th International Conference*, Seiten 446–451. Morgan Kaufmann, 1996.

[BS98] F. Baader und U. Sattler: Description Logics with Concrete Domains and Aggregation. In: H. Prade (Herausgeber): *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI-98)*, Seiten 336–340. John Wiley & Sons Ltd, 1998.

[BS00] F. Baader und U. Sattler: Tableau Algorithms for Description Logics. In: R. Dyckhoff (Herausgeber): *Proceedings of the International Conference on Automated Reasoning with Tableaux and Related Methods (Tableaux 2000)*, Band 1847 der Reihe *Lecture Notes in Artificial Intelligence*, Seiten 1–18. Springer-Verlag, 2000.

[BSM00] B. Bayer, R. Schneider und W. Marquardt: Integration of Data Models for Process Design – First Steps and Experiences. In: *Proceedings of the 7th International Symposium on Process Systems Engineering*, Band 24 der Reihe *Computers and Chemical Engineering*, Seiten 599–605, 2000.

[Cal96] D. Calvanese: Reasoning with Inclusion Axioms in Description Logics: Algorithms and Complexity. In: W. Wahlster (Herausgeber): *Proceedings of the 12th European Conference on Artificial Intelligence (ECAI-96)*, Seiten 303–307. John Wiley & Sons, 1996.

[CBH92] W.W. Cohen, A. Borgida und H. Hirsh: Computing Least Common Subsumers in Description Logics. In: W. Swartout (Herausgeber): *Proceedings of the 10th National Conference on Artificial Intelligence*, Seiten 754–760. MIT Press, 1992.

[CdGL+99] D. Calvanese, D. de Giacomo, M. Lenzerini, D. Nardi und R. Rosati: Source Integration. In: M. Jarke, M. Lenzerini, Y. Vassiliou und P. Vassiliadis (Herausgeber): *Fundamentals of Data Warehouses*, Seiten 27–45. Springer-Verlag, 1999.

[CGL98a] D. Calvanese, G. de Giacomo und M. Lenzerini: What can Knowledge Representation do for Semi-Structured Data? In: *Proceedings of the 16th National Conference of the American Association for Artificial Intelligence (AAAI-98)*, Seiten 205–210. AAAI Press/The MIT Press, 1998.

[CGL+98b] D. Calvanese, G. de Giacomo, M. Lenzerini, D. Nardi und R. Rosati: Description Logic Framework for Information Integration. In: A.G. Cohn, L.K. Schubert und S.C. Shapiro (Herausgeber): *Principles of Knowledge Representation and Reasoning: Proceedings of the 6th International Conference (KR'98)*, Seiten 2–13. Morgan Kaufmann, 1998.

[CGL+98c] D. Calvanese, G. de Giacomo, M. Lenzerini, D. Nardi und R. Rosati: Information Integration: Conceptual Modeling and Reasoning Support. In: *Proceedings of the 6th International Conference on Cooperative Information Systems (CoopIS-98)*, Seiten 280–291, 1998.

[CGL99] D. Calvanese, G. de Giacomo und M. Lenzerini: Modeling and Querying Semi-Structured Data. *Network and Information Systems*, 2(2):253–273, 1999.

[CH94a] W.W. Cohen und H. Hirsh: The Learnability of Description Logics with Equality Constraints. *Machine Learning*, 17, 1994.

[CH94b] W.W. Cohen und H. Hirsh: Learning the CLASSIC Description Logic: Theoretical and Experimental Results. In: J. Doyle, E. Sandewall und P. Torasso (Herausgeber): *Principles of Knowledge Representation and Reasoning: Proceedings of the 4th International Conference (KR'94)*, Seiten 121–132. Morgan Kaufmann, 1994.

[Cha00] H. Chalupsky: OntoMorph: A Translation System for Symbolic Knowledge. In: A.G. Cohn, F. Giunchiglia und B. Selman (Herausgeber): *Principles of Knowledge Representation and Reasoning: Proceedings of the 7th International Conference (KR2000)*, Seiten 471–482. Morgan Kaufmann, 2000.

[CL93] T. Catarci und M. Lenzerini: Interschema Knowledge in Cooperative Information Systems. *Intelligent and Cooperative Information Systems*, 2(4):375–398, 1993.

[CLN99] D. Calvanese, M. Lenzerini und D. Nardi: Unifying Class-Based Representation Formalisms. *Journal of Artificial Intelligence Research*, 1999.

[CM92] M. Chein und M.L. Mugnier: Conceptual Graphs: Fundamental Notions. *Revue d'Intelligence Artificielle*, 6(4):365–406, 1992.

[CMS98] M. Chein, M. L. Mugnier und G. Simonet: Nested Graphs: A Graph-based Knowledge Representation Model with FOL Semantics. In: A.G. Cohn, L.Schubert und S.C. Shapiro (Herausgeber): *Principles of Knowledge Representation and Reasoning: Proceedings of the 6th International Conference (KR'98)*, Seiten 524–534. Morgan Kaufman, 1998.

[DG84] W.F. Dowling und J.H. Gallier.: Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *Logic Programming*, 3:267–284, 1984.

[DHL+92] F.M. Donini, B. Hollunder, M. Lenzerini, A. Marchetti, D. Nardi und W. Nutt: The complexity of existential quantification in concept languages. *Artificial Intelligence*, 53:309–327, 1992.

[DLN+92] F.M. Donini, M. Lenzerini, D. Nardi, B. Hollunder, W. Nutt und A.M. Spaccamela: The Complexity of Existential Quantification in Concept Languages. *Artificial Intelligence*, 53:309–327, 1992.

[DLNN91a] F. Donini, M. Lenzerini, D. Nardi und W. Nutt: The complexity of concept languages. In: *Principles of Knowledge Representation and Reasoning: Proceedings of the 2nd International Conference (KR'91)*, Seiten 151–162. Morgan Kaufmann, 1991.

[DLNN91b] F.M. Donini, M. Lenzerini, D. Nardi und W. Nutt: Tractable Concept Languages. In: J. Mylopoulos und R. Reiter (Herausgeber): *Proceedings of the 12th International Joint Conference on Aritficial Intelligence (IJCAI'91)*, Seiten 458–463. Morgan Kaufmann, 1991.

[Duq87] V. Duquenne: Contextual implications between attributes and some representational properties for finite lattices. In: B. Ganter, R. Wille und K.E. Wolf (Herausgeber): *Beiträge zur Begriffsanalyse*, Seiten 213–239. B.I. Wissenschaftsverlag, 1987.

[FP96] M. Frazier und L. Pitt: CLASSIC Learning. *Machine Learning*, 25:151–193, 1996.

[Gan91] B. Ganter: Finding all closed sets: A general approach. *Order*, 8:283–290, 1991.

[Gia95] G. de Giacomo: *Decidability of Class-Based Knowledge Representation Formalisms.* Doktorarbeit, Università degli Studi di Roma "La Sapienza", 1995.

[Gia96] G. de Giacomo: Eliminating "converse" from converse PDL. *Journal of Logic, Language, and Information*, 5:193–208, 1996.

[GJ79] M.R. Garey und D.S. Johnson: *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, 1979.

[GL94a] G. de Giacomo und M. Lenzerini: Boosting the correspondence between description logics and propositional dynamic logics. In: *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI-94)*, Seiten 205–212. AAAI-Press/the MIT-Press, 1994.

[GL94b] G. de Giacomo und M. Lenzerini: Concept languages with number restrictions and fixpoints, and its relationship with mu-calculus. In: *Proceedings of the 11th European Conference on Artificial Intelligence (ECAI-94)*, Seiten 411–415. John Wiley and Sons, 1994.

[GL96] G. de Giacomo und M. Lenzerini: TBox and ABox reasoning in expressive description logics. In: *Principles of Knowledge Representation and Reasoning: Proceedings of the 5th International Conference (KR'96)*, Seiten 316–327. Morgan Kaufmann, 1996.

[GR00a] F. Goasdoué und M.-C. Rousset: Rewriting Conjunctive Queries using Views in Description Logics with Existential Restrictions. In: F. Baader und U. Sattler (Herausgeber): *Proceedings of the International Workshop on Description Logics 2000 (DL2000)*, Nummer 33 in *CEUR-WS*. Verlag Mainz, 2000. Proceedings online available from http://SunSITE.Informatik.RWTH-Aachen.DE/Publications/CEUR-WS/Vol-33/.

[GR00b] F. Goasdoué und M.-C. Rousset: Rewriting Conjunctive Queries using Views in Description Logics with Existential Restrictions. Technischer Bericht, Université Paris Sud, Laboratire de Recherche ein Informatique, 2000.

[GW99] B. Ganter und R. Wille: *Formal Concept Analysis – Mathematical Foundations*. Springer-Verlag, 1999.

[Har93] P. Harmon: G2: Gensym's real-time expert system. *Intelligent Software Systems*, 9(3):1–16, 1993.

[HB91] B. Hollunder und F. Baader: Qualifying Number Restrictions in Concept Languages. In: *Principles of Knowledge Representation and Reasoning: Proceedings of the 2nd International Conference (KR'91)*, Seiten 335–346. Morgan Kaufmann, 1991.

[HNS90] B. Hollunder, W. Nutt und M. Schmidt-Schauss: Subsumption Algorithms for Concept Description Languages. In: *Proceedings of the 9th European Conference on Artificial Intelligence (ECAI-90)*, Pitman Publishing, 1990.

[Hor98] I. Horrocks: Using an Expressive Description Logic: FaCT or Fiction? In: A.G. Cohn, L. Schubert und S.C. Shapiro (Herausgeber): *Principles of Knowledge Representation and Reasoning: Proceedings of 6th International Conference (KR'98)*, Seiten 636–647. Morgan Kaufmann, 1998.

[HP98] I. Horrocks und P.F. Patel-Schneider: FaCT and DLP. In: H. de Swart (Herausgeber): *Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX-98)*, Band 1397 der Reihe *Lecture Notes in Artificial Intelligence*, Seiten 27–30. Springer-Verlag, 1998.

[HS73] H. Herrlich und G.E. Strecker: *Category Theory*. Allyn and Bacon, 1973.

[HS99] I. Horrocks und U. Sattler: A Description Logic with Transitive and Inverse Roles and Role Hierarchies. *Journal of Logic and Computation*, 9(3):385–410, 1999.

[HST99] I. Horrocks, U. Sattler und S. Tobies: Practical Reasoning for Expressive Description Logics. In: H. Ganzinger, D. McAllester und A. Voronkov (Herausgeber): *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, Nummer 1705 in *Lecture Notes in Artificial Intelligence*, Seiten 161–180. Springer-Verlag, 1999.

[HST00] I. Horrocks, U. Sattler und S. Tobies: Practical Reasoning for Very Expressive Description Logics. *Logic Journal of the IGPL*, 8(3):239–264, May 2000.

[HT00] I. Horrocks und S. Tobies: Reasoning with Axioms: Theory and Practice. In: A.G. Cohn, F. Giunchiglia und B. Selman (Herausgeber): *Principles of Knowledge Representation and Reasoning: Proceedings of the 7th International Conference (KR2000)*, Seiten 285–296. Morgan Kaufmann, 2000.

[JM95] M. Jarke und W. Marquardt: Design and Evaluation of Computer-Aided Process Modeling. *Intelligent Systems in Process Engineering*, 92(312):97–109, 1995.

[KB99] R. Küsters und A. Borgida: What's in an Attribute? Consequences for the Least Common Subsumer. Technischer Bericht DCS-TR-404, Rutgers University, USA, 1999.

[KBR86] T. Kaczmarek, R. Bates und G. Robins: Recent Developments in NIKL. In: *Proceedings of the 5th National Conference on Artificial Intelligence (AAAI'86)*, Band 2, Seiten 978–985. Morgan Kaufmann, 1986.

[KLM98] C. Krobb, B. Lohmann und W. Marquardt: The Chemical Engineering Data Model VeDa. Part 6: The Process of Model Development. Interner Bericht LPT-1998-06, Lehrstuhl für Prozesstechnik, RWTH Aachen, 1998. Siehe http://www.lfpt.rwth-aachen.de/Publication.

[Kob91] A. Kobsa: First Experiences with the SB-ONE Knowledge Representation Workbench in Natural Language Applications. *SIGART Bulletin*, 2(3):70–76, 1991.

[Kro97] C. Krobb: Entwicklung einer Spezialisierungshierarchie für Modellierungsschritte im objekt-orientierten Datenmodell VeDa. Diplomarbeit, RWTH Aachen, 1997. Siehe http://www-lti.informatik.rwth-aachen.de/Forschung/Papers.html.

[Küs98] R. Küsters: Characterizing the Semantics of Terminological Cycles in $\mathcal{ALN}$ using Finite Automata. In: A.G. Cohn, L. Schubert und S.C. Shapiro (Herausgeber): *Principles of Knowledge Representation and Reasoning: Proceedings of 6th International Conference (KR'98)*, Seiten 499–510. Morgan Kaufmann, 1998.

[Küs00] R. Küsters: *Non-Standard Inference Services in Description Logics*. Doktorarbeit, RWTH Aachen, 2000.

[LB87] H. Levesque und R.J. Brachman: Expressiveness and Tractability in Knowledge Representation and Reasoning. *Computational Intelligence*, 3:78–93, 1987.

[Lut99] C. Lutz: Complexity of Terminological Reasoning Revisited. In: *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, Band 1705 der Reihe *Lecture Notes in Artificial Intelligence*, Seiten 181–200. Springer-Verlag, 1999.

[Mac91] R. MacGregor: Inside the LOOM classifier. *SIGART Bulletin*, 2(3):88–92, 1991.

[Mai83] D. Maier: *The Theory of Relational Databases*. Computer Science Press, Rockville, 1983.

[Man99] T. Mantay: Computing Least Common Subsumers in Expressive Description Logics. In: P. Lambrix, A. Borgida, M. Lenzerini, R. Möller und P. Patel-Schneider (Herausgeber): *Proceedings of the International Workshop on Description Logics 1999 (DL'99)*, Nummer 22 in *CEUR-WS*, Seiten 86–90, 1999. Proceedings online available from http://SunSITE.Informatik.RWTH-Aachen.DE/Publications/CEUR-WS/Vol-22/.

[Mar92] W. Marquardt: An Object-Oriented Representation of Structured Process Models. In: R. Gani (Herausgeber): *Proceedings of the European Symposium on Computer Aided Process Engineering*, Seiten 329–336, 1992.

[Mar96a] W. Marquardt: Towards a Process Modeling Methodology. In: R. Berber (Herausgeber): *Model-Based Process Control*, NATO-ASI series, Seiten 3–40. Kluwer Press, 1996.

[Mar96b] W. Marquardt: Trends in Computer-Aided Process Modeling. *Computers and Chemical Engineering*, 20(6/7):591–609, 1996.

[MC92] M.L. Mugnier und M. Chein: Polynomial Algorithms for Projection and Matching. In: H.D. Pfeiffer und T.E. Nagle (Herausgeber): *Proceedings of the 7th Annual Workshop on Conceptual Structures: Theory and Implementation*, Band 754 der Reihe *Lecture Notes in Artificial Intelligence*, Seiten 239–251. Springer-Verlag, 1992.

[MDW91] E. Mays, R. Dionne und R. Weida: K-Rep system overview. *SIGART Bulletin*, 3(2):93–97, 1991.

[MGG93] W. Marquardt, A. Gerstlauer und E.D. Gilles: Modeling and Representation of Complex Objects: A Chemical Engineering Perspective. In: *Proceddings of the 6th International Conference on Intelligent Engineering Applications/Artificial Intelligence in Engineering*, Seiten 219–228, 1993.

[Min75] M. Minsky: A framework for representing knowledge. In: P. Winston (Herausgeber): *The Psychology of Computer Vision*, McGraw-Hill, New York, 1975.

[MIP+98] D. McGuinness, C. Isbell, M. Parker, P. Patel-Schneider, L. Resnick und C. Welty: A Description Logic-based configurator on the Web. *ACM Sigart Bulletin*, 9:20–22, 1998.

[MPS98] D.L. McGuinness und P.F. Patel-Schneider: Usability Issues in Knowledge Representation Systems. In: *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98) and of the 10th Conference on Innovative Applications of Artificial Intelligence (IAAI-98)*, Seiten 608–614. AAAI Press, 1998.

[MRI95] D.L. McGuinness, L. Alperin Resnick und C. Isbell: Description Logic in practice: A CLASSIC application (Video Presentation). In: *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJ-CAI'95)*, Seiten 2045–2046. Morgan Kaufmann, 1995.

[MvWB99] W. Marquardt, L. von Wedel und B. Bayer: Perspectives on Lifecycle Process Modeling. In: *Proceedings of 5th International Conference on Foundations of Computer-Aided (FOCAPD'99)*, 1999.

[MW98a] D.L. McGuinness und J.R. Wright: Conceptual modeling for configuration: A description logic-based approach. *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing Journal*, 12:333–334, 1998.

[MW98b] D.L. McGuinness und J.R. Wright: An industrial strength Description Logic-based configurator platform. *IEEE Intelligent Systems*, 13(4):66–77, 1998.

[Nas97] M. Nassiri: Berechnung einer erweiterten Subsumtionshierarchie. Diplomarbeit, RWTH Aachen, 1997. Siehe http://www-lti.informatik.rwth-aachen.de/Forschung/Papers.html.

[Nav96] NaviCon GmbH: *Anaconda: Aufbau begrifflicher Informationssysteme*, 1996. Siehe http://www.mathematik.tu-darmstadt.de/ags/esz/.

[Neb90a] B. Nebel: *Reasoning and Revision in Hybrid Representation Systems*, Band 422 der Reihe *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 1990.

[Neb90b] B. Nebel: Terminological Reasoning is Inherently Intractable. *Artificial Intelligence*, 43(2):235–249, 1990.

[Neb91] B. Nebel: Terminological cycles: Semantics and computational properties. In: J. Sowa (Herausgeber): *Formal Aspects of Semantic Networks*, Seiten 331–361. Morgan Kaufmann, 1991.

[Pel91] C. Peltason: The BACK System - An Overview. *SIGART Bulletin*, 2(3):114–119, 1991.

[Pie91] B.C. Pierce: *Basic Category Theory for Computer Scientists.* Foundations of Computing Series. MIT Press, 1991.

[Poh96] K. Pohl: *Process Centered Requirements Engineering.* Wiley, New York, 1996.

[Qui68] M. Quillian: Semantic memory. In: M. Minsky (Herausgeber): *Semantic Information Processing*, Seiten 216–270. MIT Press, 1968.

[Rey77] S.W. Reyner: An Analysis of a good Algorithm for the Subtree Problem. *SIAM Journal of Computing*, 6(4):730–732, December 1977.

[Sat95] U. Sattler: A Concept Language for an Engeneering Application with Part-Whole Relations. In: A. Borgida, M. Lenzerini, D. Nardi und B. Nebel (Herausgeber): *Proceedings of the 1995 International Workshop on Description Logics (DL'95)*, Seiten 119–123, 1995.

[Sat96] U. Sattler: A Concept Language Extended with Different Kinds of Transitive Roles. In: G. Görz und S. Hölldobler (Herausgeber): *20. Deutsche Jahrestagung für Künstliche Intelligenz*, Nummer 1137 in *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 1996.

[Sat98] U. Sattler: *Terminological knowledge representation systems in a process engineering application.* Doktorarbeit, RWTH Aachen, 1998. Siehe http://www-lti.informatik.rwth-aachen.de/Forschung/Papers.html.

[Sat00] U. Sattler: Description Logics for the Representation of Aggregated Objects. In: W. Horn (Herausgeber): *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI2000)*. IOS Press, 2000.

[Sch89] M. Schmidt-Schauss: Subsumption in KL-ONE is Undecidable. In: *Principles of Knowledge Representaion and Reasoning: Proceedings of the 1st International Conference (KR'89)*, Seiten 421–431, 1989.

[Sch91] K. Schild: A correspondence theory for terminological logics: Preliminary report. In: J. Mylopoulos und R. Reiter (Herausgeber): *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI'91)*, Seiten 466–471. Morgan Kaufmann, 1991.

[Sch93] A. Schaerf: On the Complexity of the Instance Checking Problem in Concept Languages with Existential Quantification. *Journal of Intelligent Information Systems*, 2:265–278, 1993.

[SM98a] D. Souza und W. Marquardt: The Chemical Engineering Data Model VeDa. Part 2: Structural Modeling Objects. Interner Bericht LPT-1998-02, Lehrstuhl für Prozesstechnik, RWTH Aachen, 1998. Siehe http://www.lfpt.rwth-aachen.de/Publication.

[SM98b] D. Souza und W. Marquardt: The Chemical Engineering Data Model VeDa. Part 3: Geometrical Modeling Objects. Interner Bericht LPT-1998-03, Lehrstuhl für Prozesstechnik, RWTH Aachen, 1998. Siehe http://www.lfpt.rwth-aachen.de/Publication.

[Sow84] J.F. Sowa: *Conceptual Structures – Information, Processing in Mind and Machine.* Addison Wesley, 1984.

[SS91] M. Schmidt-Schauss und G. Smolka: Attributive Concept Descriptions with Complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[Stu96a] G. Stumme: The concept classification of a terminology extended by conjunction and disjunction. In: N. Foo und R. Goebel (Herausgeber): *PRICAI'96: Topics in artificial intelligence*, Band 1114 der Reihe *Lecture Notes in Artificial Intelligence*, Seiten 121–131. Springer-Verlag, 1996.

[Stu96b] G. Stumme: Exploration Tools in Formal Concept Analysis. In: *Proceedings of Ordinal and Symbolic Data Analysis (OSDA'95)*, Band 8 der Reihe *Studies in Classification, Data Analysis, and Knowledge Organization*, Seiten 31–44. Springer-Verlag, 1996.

[SW00] G. Stumme und R. Wille: *Begriffliche Wissensverarbeitung – Methoden und Anwendungen*. Springer-Verlag, 2000.

[Tob99] S. Tobies: A PSpace Algorithm for Graded Modal Logic. In: H. Ganzinger (Herausgeber): *Proceedings of the 16th International Conference on Automated Deduction (CADE-16)*, Nummer 1632 in *Lecture Notes in Artificial Intelligence*, Seiten 52–66. Springer-Verlag, 1999.

[Val84] L.G. Valiant: A Theory of the Learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.

[vWM98] L. von Wedel und W. Marquardt: The Chemical Engineering Data Model VeDa. Part 5: Material Modeling Objects. Interner Bericht LPT-1998-05, Lehrstuhl für Prozesstechnik, RWTH Aachen, 1998. Siehe http://www.lfpt.rwth-aachen.de/Publication.

[vWM00] L. von Wedel und W. Marquardt: ROME: A Repository to Support the Integration of Models over the Lifecycle of Model-based Engineering Processes. In: *Proceedings of ESCAPE-10*, 2000.

[Woo75] W.A. Woods: What's in a link: Foundations for semantic networks. In: D.G. Bobrow und A.M. Collins (Herausgeber): *Representation and Understanding: Studies in Cognitive Science*, Seiten 35–82. Academic Press, 1975.

[WWV+93] J.R. Wright, E.S. Weixelbaum, G.T. Vesonder, K. Brown, S.R. Palmer, J.I. Berman und H.H. Moore: A knowledge-based configurator that supports sales, engineering, and manufacturing at AT&T network systems. *AI Magazine*, 14(3):69–80, 1993.

Lebenslauf des Autors

Persönliche Daten

Name:	Ralf Molitor
Geburtsdatum:	24. Juni 1973
Geburtsort:	Rheydt, Deutschland
Nationalität:	deutsch

Schulausbildung

- *August 1979 – Juli 1983:* Katholische Grundschule Liedberg
- *August 1983 – Juni 1992:* Gymnasium Korschenbroich. Abschluss Abitur.

Studium

- *Oktober 1992 – September 1994:* Grundstudium der Informatik (Nebenfach Mathematik) an der RWTH Aachen. Abschluss Vordiplom.
- *Oktober 1994 – Juni 1997:* Hauptstudium der Informatik (Nebenfach Mathematik) an der RWTH Aachen. Abschluss Diplom.

Promotionsstudium

- *Oktober 1997 bis Dezember 2000:* Promotionsstudium der Informatik an der RWTH Aachen.

Stipendium

- *Oktober 1997 bis September 2000:* Stipendiat im Graduiertenkolleg „Informatik & Technik“ an der RWTH Aachen.

Berufliche Tätigkeit

- *Oktober 2000 bis Dezember 2000:* Wissenschaftlicher Angestellter am Lehr- und Forschungsgebiet Theoretische Informatik der RWTH Aachen