Expressive Description Logics: Foundations for Practical Applications

Habilitationsschrift

Ralf Möller

University of Hamburg, Computer Science Department,
Vogt-Kölln-Straße 30, 22527 Hamburg

September 2000

Acknowledgments

First of all I would like to thank Prof. Dr. Bernd Neumann. His encouraging comments and enthusiasm over the years shaped the whole work discussed in this monograph. Always emphasizing the importance of applications to demonstrate the adequacy of theoretical work, he suggested to particularly consider real-world problems. Thus, it is also thanks to him that the description logic system RACE can deal with hard inference problems encountered in real-world applications. We had many interesting discussions during our research on deductive information systems. I also thank Bernd Neumann for valuable comments on a draft of this monograph. For additional reviews of this monograph I thank Prof. Dr. Franz Baader and Prof. Dr. Christopher Habel.

I am also particularly grateful to my colleague Dr. Volker Haarslev. Together we have been discussing the topics on description logic systems for the last three years. Many times we have been investigating new optimization strategies, which turned out to effective but were hard to implement. Many long discussions with Volker Haarslev lead to the development of formal proofs of the algorithms implemented in the description logic system RACE such that the application-oriented work of this thesis is complemented with theoretical results.

Furthermore, I would like to thank my colleague Michael Wessel for the years of exciting research we did together in the area of description logics, spatial reasoning and default reasoning. I also thank Michael Wessel for valuable comments on the final draft of this Habilitation Thesis. Anni-Yasmin Turhan contributed to the development of the RACE system by testing it and writing the documentation. Thanks also for careful comments on a draft version of this monograph.

For discussions and collaboration in the context of description logic reasoning and reasoning about topological relations I am indebted to Carsten Lutz (now at LuFG Theoretische Informatik at the University of Aachen). With his theoretical knowledge he immensely contributed to various decidability and undecidability results in the area of combined terminological and topological reasoning. Thanks also to other members of the LuFG Theoretische Informatik in Aachen for their detailed comments on papers covering particular theoretical topics summarized in this work.

All members of the Cognitive Systems Group of the University of Hamburg, in particular Carsten Schröder, and all members of the BAND project team from Lavielle GmbH and from the LKI, namely Rainer Joswig, Kay Hidde and Thomas Mantay, contributed to the results reported here. Many thanks to all of them.

Last but not least many thanks also to my diploma students Alissa Kaplunova, Irena Kortas, Glen Lehnert, Knud Möhle, Omar Nuri and Christian Zacharias. They all provided small pieces of the mosaic that now has been shaped into this monograph. Other students at the Computer Science Department have also contributed to the development of the RACE system.

Contents

Teil I

German Summary

Einleitung und Motivation

Die Probleme, die in Anwendungsprojekten der Informationstechnologie zu bearbeiten sind, werden zunehmend komplexer, da vielfach "tiefere" ontologische Modellierungen zur Problemlösung eingesetzt werden, Aspekte der Verteilung zu beachten sind, sowie auch Sicherheitsaspekte eine große Rolle spielen. So wundert es nicht, daß verschiedene Ansätze, Techniken und Methoden zur Bewältigung der Komplexität vorgeschlagen wurden. Zu nennen sind in diesem Zusammenhang zum Beispiel wissensbasierte Systeme, objektorientierte Rahmensysteme, vereinheitlichte Modellierungssprachen (z.B. UML, Unified Modeling Language), Spezifikationssprachen für das Verhalten von technischen Systemen, Modellierungssprachen für Arbeitsvorgänge (workflow modeling languages), Organisationsmodelle für Geschäftsprozesse usw. Das Ziel der verschiedenen Ansätze ist, die Erstellung benutzbarer, verläßliche und anpaßbare Softwaresysteme zu unterstützen. Allen Ansätzen gemeinsam ist der Bedarf nach automatischen Überprüfungstechniken für definierte Modelle. Es zeigt sich weiterhin in vielen Anwendungskontexten seit jeher ein hoher Bedarf für Systeme, mit denen komplexe Anwendungsprobleme bzw. deren Teilprobleme auf der Basis von deklarativen Modellen der Anwendungsdomäne automatisch gelöst werden können. Automatisches Lösen bedeutet in diesem Zusammenhang, daß die Problemlösung durch allgemeine Dienste bzw. vorgefertigte Bausteine und nicht durch Programmierung eines speziellen, anwendungsabhängigen Spezialverfahrens erbracht wird. Diese Habilitationsschrift faßt praktische und theoretische Forschungsergebnisse zur Entwicklung von Modellierungswerkzeugen zusammen, mit denen es möglich ist, deklarative Modelle aufzustellen, die einerseits auf Konsistenz geprüft werden können und andererseits als Basis für automatische Problemlösungsprozesse in verschiedenen Anwendungskontexten dienen können.

Um die automatische Verarbeitung von deklarativen Modellen zu ermöglichen, ist es notwendig, daß die Modellierungssprache auf einer formalen Semantik beruht. Auf der Basis der formalen Semantik einer Repräsentationsprache lassen sich Inferenzprobleme formal definieren. Das Lösen eines Inferenzproblems eines bestimmten Typs durch ein Softwaresystem wird häufig auch Inferenzdienst genannt. Algorithmen zur Realisierung von Inferenzdiensten können unter Bezugnahme auf die Semantik der Repräsentationssprache daraufhin bewertet werden, ob sie vollständig und

korrekt sind und auch terminieren. Je nach Ausdrucksmächtigkeit der Repräsentationssprache kann die Lösung von Inferenzproblemen durch unterschiedliche Verfahren erfolgen.

So kann zum Beispiel ein Inferenzproblem durch Suche über einem extensional gegebenen Datenbestand gelöst werden (vgl. z.B. klassische Datenbanksysteme). Bei komplexeren Anwendungen werden zunehmend ausdrucksstärkere Repräsentations- und Anfragesprachen untersucht, so daß die Beantwortung von Anfragen nicht mehr nur in einer reinen Wiedergabe der explizit gespeicherten Daten besteht. Die Beantwortung von Anfragen bedingt ein Schlußfolgern über implizite Informationen. Obwohl neuere Datenbanksysteme sich mehr und mehr in diese Richtung bewegen (vgl. [Abiteboul et al., 1995; Zaniold et al., 1997]), so läßt sich doch sagen, daß der Fokus von klassischen Datenbanksystemen eher auf Themen liegt wie z.B. Persistenz und Transaktionen sowie auch Effizienz in Bezug auf Speicherbedarf, Speichergeschwindigkeit und Suchgeschwindigkeit. Die Beantwortung von Anfragen, bei denen über implizite Informationen (bzw. Sachverhalte) geschlossen werden muß, fällt in den Bereich der sog. Inferenzsysteme (z.B. [Genesereth & Nilsson, 1987; Fitting, 1996; Poole et al., 1998]). Nichtsdestotrotz, da ausdrucksstärkere Repräsentationssprachen in vielen Anwendungen benötigt werden, spielen auch Inferenzen im Datenbankkontext eine immer größere Rolle [Chomicki & Saake, 1998; Kuper et al., 2000]. Es überrascht daher nicht, daß die Ergebnisse von Forschungsarbeiten zu formalen Inferenzsystemen und zur Wissensrepräsentation im Rahmen der konventionellen Informatik immer bedeutsamer werden (und umgekehrt).

Unter dem Begriff (formale) Inferenzsysteme seien Systeme zusammengefaßt, die Theoreme in einer entsprechend ausdrucksstarken Logik beweisen können. Wichtige Anwendungen von Inferenzsystemen sind sog. Informationssysteme, d.h. Anwendungssysteme mit denen ein Benutzer Auskünfte über einen bestimmten Informationsbestand erhalten kann. Ein Informationssystem, dessen Hauptleistungen durch formale Inferenzsysteme für ausdrucksstarke Beschreibungssprachen erbracht werden, soll hier als deduktives Informationssystem bezeichnet werden. Ein deduktives Informationssystem muß allerdings nicht auf einer monolithischen Architektur basieren, es kann durch mehrere kooperierende Teilkomponenten realisiert sein.

Die Entwicklung von formalen Inferenzsystemen bzw. Theorembeweisern hat eine lange Tradition. In der Literatur sind viele Beiträge aus unterschiedlicher Sicht und mit unterschiedlichem mathematischen Hintergrund zu finden. Für viele Anwendungsprobleme eignen sich zur Problemlösung *logikbasierte* Modellierungs- und Schlußsysteme. In dieser Arbeit wird in diesem Kontext ein anwendungsorientierter Ansatz verfolgt, in dem theoretische Resultate über die Entscheidbarkeit von verschiedenen Repräsentationssprachen durch praktische Arbeiten zu „effizienten“ Beweisprozeduren ergänzt werden. Effiziente Beweisprozeduren wiederum sind notwendig für die praktische Realisierung von Anfragebeantwortungssystemen mit

ausdrucksstarken Repräsentationssprachen. Die Entwicklung von Algorithmen, die vollständig und korrekt sind (und terminieren), aber im Durchschnittsfall nicht in eine kombinatorische Explosion hineinlaufen, ist ein vielversprechendes Forschungsfeld und stellt vielfältige Anforderungen auch aus einer theoretischen Sicht.

Wie schon erwähnt, können formale Beweissysteme in einem Informationsrecherche-Szenario zur Beantwortung von Anfragen eingesetzt werden. In diesem Kontext müssen Repräsentationssysteme, die in der Praxis angewendet werden sollen, eine adäquate Ausdruckskraft für eine breitgefächerte Art von Phänomen und Effekten aufweisen. So bildet zum Beispiel für geographische Informationssysteme (GIS) räumliches Hintergrundwissen über Artefakte und natürliche Objekte die Basis für Problemlösungsprozesse. Falls Anwendungsprobleme durch Inferenzdienste gelöst werden sollen, so ist die Festlegung einer formalen Semantik für die verwendeten Repräsentationskonstrukte unumgänglich, da nur so die Aufgabe von Inferenzdiensten definiert werden kann. Dabei ist das Zusammenspiel von räumlichem (bzw. zeitlichem) und terminologischem Wissen auf einer semantischen Ebene von besonderer Bedeutung. Die systematische Integration des konzeptuellen, begrifflichen Schließens mit Aspekten des räumlichen Schließens ist ein noch junges Forschungsgebiet zu dem in dieser Arbeit durch die Entwicklung von vollständigen und korrekten Inferenzalgorithmen wichtige Grundlagen gelegt werden. Um eine automatische Verarbeitung von deklarativen Modellen zu ermöglichen, sollten auch im Kontext von Informationsrecherchesystemen vorzugsweise Repräsentationssprachen untersucht werden, für die vollständige und korrekte (und terminierende) Inferenzalgorithmen existieren. Auf verschiedene Ansätze zur Logikprogrammierung mit Horn-Formeln wird daher nur am Rande eingegangen (vgl. auch die Argumentation in [Baader, 1999]).

Informationssysteme sind nicht die einzigen Anwendungsfelder, in denen formale Inferenzsysteme erfolgreich eingesetzt werden können. Durch das stets wachsende Interesse im Bereich des elektronischen Handels in verteilten Systemen wird das Problem der Verifikation vordringlich. Es geht dabei darum, durch Auswertung eines Modells zu prüfen, daß bestimmte, meist interagierende Dienste auch unter allen Umständen erbracht werden können. Als weiteren bedeutsamen Anwendungskontext für die formale Verifikation sind Telekommunikationssysteme zu nennen. In diesen Systemen wird eine Vielzahl von ebenfalls stark interagierenden Diensten angeboten. Die Kombinierbarkeit von Diensten stellt dabei ein großes Problem dar. Daher besteht vielfach der Wunsch, ausgehend von einem formalen Modell eines Systems und einer Menge von Einschränkungen (Invarianten) zu prüfen, ob ein fehlerfreier Betrieb des Systems gewährleistet werden kann. In diesem Kontext werden vielfach sogenannte Algorithmen zur Modellüberprüfung (model checking algorithms, vgl. [Clarke & Kurshan, 1996] für eine Übersicht) eingesetzt, um zu überprüfen, ob eine spezielle Realisierung eines Systems bestimmte Anforderungen erfüllt. Grundannahme bei der Modellierung eines Systems ist dabei, daß das Verhalten des Systems

als endlicher Zustandsraum repräsentierbar ist. Zustandsübergänge werden durch Aktionen bzw. Ereignisse ausgelöst. Anforderungen legen fest, daß in bestimmten Zuständen bestimmte Prädikate gelten müssen. Ziel der Modellüberprüfung ist es zu überprüfen, daß eine gegebene Systemrealisierung die Anforderungen erfüllt.

Ein Problem des Ansatzes der Modellüberprüfung ist, daß (i) nur endliche Zustandsräume behandelt werden können und (ii) nur zustandsorientierte Aspekte von Anwendungen betrachtet werden können. Daher werden in zunehmendem Maße auch Theorembeweiser zur Überprüfung von Spezifikationen eingesetzt. Theorembeweiser können mit unendlichen Zustandsräumen umgehen und unterstützen auch eine Überprüfung der Datenmodelle [Katoen, 1999].

Damit Deduktionstechniken erfolgreich in Anwendungen eingesetzt werden können, müssen ausdrucksstarke Modellierungs- und Repräsentationssprachen verwendet werden und effiziente Beweiserimplementierungen verfügbar sein. Wir werden in dieser Arbeit sehen, daß Theorembeweiser entwickelt werden können, die im Bereich der Verifikation von Spezifikationen für Telekommunikationssysteme erfolgreich eingesetzt werden können und somit Modellüberprüfungsansätze ergänzen können.

Eine Familie von logikbasierten Repräsentationsprachen, die für die obengenannten Aufgaben sehr gut geeignet ist, wird als Beschreibungslogiken bezeichnet (description logics, DLs) [Baader et al., 2001]. Mit Beschreibungslogiken wird eine rechnerinterne Modellierung von Objekten und Phänomenen der Realwelt vorrangig aus einer objektzentrierten Sicht vorgenommen [Woods & Schmolze, 1992; Donini et al., 1996]. Es werden hierzu Konstrukte zur Formulierung von sog. Konzepten und Rollen (binäre Relationen) sowie entsprechende, auf der formalen Semantik der Repräsentationssprache basierende Inferenzdienste bereitgestellt (z.B. Konsistenzprüfung, Klassifikation). In jüngster Zeit sind auch Beschreibungslogiken mit n-stelligen Relationen untersucht worden [Calvanese et al., 1998]. Lassen sich Teilprobleme einer Anwendung unter Rückgriff auf Beschreibungslogiken als entsprechende Inferenzprobleme formulieren, so stehen theoretisch abgesicherte und systematisch getestete Beweissysteme zur Lösung dieser Teilprobleme zur Verfügung.

Erfahrungen mit Anwendungen belegen, daß ausdrucksstarke Formalismen benötigt werden, damit nicht auf Ad-hoc-Lösungen für Teilprobleme zurückgegriffen werden muß. Die Entwicklung von Inferenzsystemen, die auf vollständigen und korrekten Algorithmen basieren und im mittleren Fall (average case) ein gutes Laufzeitverhalten zeigen, ist für ausdrucksstarke (aber entscheidbare) Beschreibungslogiken ein relativ neues Forschungsgebiet. Innerhalb dieser Habilitationsarbeit wurden zu den Problemstellungen der aufgezeigten Forschungsbereiche Lösungen erarbeitet. Es sind folgende Hauptpunkte hervorzuheben:

- Theoretische Arbeiten zur Entwicklung von Kalkülen für ausdrucksstarke Beschreibungslogiken.

- Entwicklung von optimierten aber vollständigen und korrekten Algorithmen für praxisgerechte Schlußverfahren (insbesondere für das Schließen mit Individuen).

- Implementierung eines Beschreibungslogik-Inferenzsystems namens RACE zur Durchführung von Untersuchungen zum durchschnittlichen (average case) Verhalten von Inferenzalgorithmen für Anwendungswissensbasen. Hierzu gehören empirische Analysen der Performanz des Inferenzsystems RACE für von anderen Forschergruppen entwickelten Formalisierungen von großen Ontologien (ontology engineering) sowie auch für die Verifikation von Soft- und Hardwaresystemen am Beispiel von Telekommunikationsanlagen.

- Erweiterung der Theorie zu Beschreibungslogiken zur Einbeziehung von Schlüssen bzgl. Einschränkungen über kontinuierlichen und diskreten Domänen (sog. Constraints) mit Anwendungen im Bereich der Konfigurierung.

- Theoretische Arbeiten zur semantikbasierten Integration von Techniken zum räumlichen (und zeitlichen) Schließen in das konzeptuelle Schließen mit Beschreibungslogiken.

- Aufzeigen der praktischen Verwendung der beschreibungslogischen Modellierungskonstrukte für Anwendungen aus verschiedenen Kontexten: Inferenzbasierte bzw. deduktive Informationssysteme, geographische Informationssysteme und Agenten-orientierte Informationssysteme.

In dieser Arbeit wird aufgezeigt, daß sich Beschreibungslogiken in vielfältiger Weise zur Modellierung und Problemlösung in Anwendungssystemen einsetzen lassen. Obwohl nicht bestritten wird, daß in bestimmten Kontexten auch andere Logikformalismen erfolgreich als Basis von Inferenzsystemen zum Einsatz kommen können (z.B. Datalog in sog. deduktiven Datenbanken), bildet gerade das breite Anwendungsspektrum für Beschreibungslogiken die Motivation für die aufgezeigten Forschungsarbeiten.

Stand der Kunst, Forschungsziele und -ergebnisse

Im folgenden Abschnitt wird der Stand der Forschung im Gebiet der beschreibungslogischen Repräsentations- und Inferenzsysteme näher beleuchtet. Nach einer Betrachtung der konzeptuellen Wissensrepräsentation werden Aspekte des Forschungsgebietes räumliches Schließen aufgeführt, die bzgl. der in dieser Arbeit verfolgten Kopplung von konzeptuellem und räumlichem Schließen besonders bedeutsam sind. Hierauf aufbauend werden die in dieser Arbeit erreichten Fortschritte zusammengefaßt.

Konzeptuelle Wissensrepräsentation

Logische Repräsentationssprachen haben sich bei der Erstellung von formalen Modellen als ein wichtiges Mittel zur Wissensrepräsentation erwiesen. Wir betrachten in dieser Arbeit mit Beschreibungslogiken primär sog. *konzeptuelle* Wissensrepräsentationsansätze [Brachman & Schmolze, 1985], da sie sich für viele Anwendungsbereiche als ein gutes Fundament erwiesen haben.

Frühe Ansätze zur konzeptuellen Wissensrepräsentation waren zunächst noch stark durch Ad-hoc-Betrachtungen der menschlichen Informationsverarbeitung geprägt. Inzwischen hat sich durch die Untersuchung der Semantik von Repräsentationskonstrukten eine starke Formalisierung durchgesetzt (siehe [Baader, 1999; Baader & Sattler, 2000] für einen Überblick über verschiedene Arbeiten). Die Ergebnisse der Forschungsarbeiten verdeutlichen, daß beschreibungslogische Repräsentationssprachen für die maschinelle Informationsverarbeitung besonders geeignet sind. Durch die formale Semantik von logischen Repräsentationssprachen ist es möglich, Inferenzdienste mit verifizierten Algorithmen zu realisieren. Dadurch werden Anwendungsarchitekturen möglich, die sich nicht auf anwendungsspezifische Verfahren sondern auf allgemeine, formal definierte Inferenzdienste stützen. Durch diese Art der Modellierung von zumeist komplexen Systemen kann eine erhebliche Einsparung bei den Entwicklungskosten bei gleichzeitiger Erhöhung der Sicherheit erzielt werden,

da für eine Einzelanwendung auf theoretisch untersuchte und praktisch ausgetestete Algorithmen und Problemlösungsverfahren zurückgegriffen werden kann.

Von besonderer Wichtigkeit für die maschinelle Verarbeitung der Modelle ist in diesem Zusammenhang die Entscheidbarkeit der logischen Repräsentationssprache. Der Einsatz von unentscheidbaren Sprachen und unvollständigen Inferenzverfahren ist zwar grundsätzlich möglich, birgt jedoch die Gefahr, daß die Unvollständigkeit im Anwendungskontext nicht besonders beachtet wird. Für viele Deduktionsprobleme sind dagegen Lösungen auf der Basis eines vollständigen und korrekten (und terminierenden) Kalküls erforderlich. Dabei wird die Tatsache, daß sich ein Teilproblem einer Anwendung bei entscheidbaren Sprachen direkt durch formale Deduktionstechniken lösen läßt (sofern das Problem nicht in einer zu hohen Komplexitätsklasse liegt) als besonderer Vorteil angesehen. Eine teure Entwicklung und Verifikation von Spezialalgorithmen ist nicht notwendig. Beschreibungslogiken sind ein prominenter Vertreter der Klasse der entscheidbaren Repräsentationssprachen und stehen im Mittelpunkt der hier zusammengefaßten Forschungsarbeiten.

Beschreibungslogiken: In Beschreibungslogiken werden sog. Konzeptterme zur Repräsentation von Klassen von (abstrakten) Objekten verwendet. Für eine spezielle Anwendung werden primitive Basiskonzepte festgelegt und mit Kombinationsoperatoren zu komplexeren Konzepttermen verknüpft. Binäre Beziehungen zwischen Individuen werden durch sog. Rollenterme beschrieben. Über Rollenterme können die Eigenschaften von Objekten festgelegt werden.

Konzeptterme beschreiben Modellstrukturen (im logischen Sinne) und drücken Einschränkungen über Modellstrukturen aus. Je nach Ausdrucksmächtigkeit der Sprachen werden Termbildungsoperatoren wie Konjunktion, Disjunktion und Negation zur Konzeptbildung, sowie Existenz- und Allquantoren zur Einschränkung der Strukturen von Rollenfüllern zugelassen (siehe z.B. die Referenzsprache $\mathcal{ALC}$ [Schmidt-Schauss & Smolka, 1991]). Ein wichtiges Inferenzproblem ist der Konsistenz- bzw. Erfüllbarkeitstest für Konzeptterme (siehe unten für weitere Inferenzdienste). Ein Konzept ist konsistent, wenn es eine Interpretationsfunktion gibt, die das Konzept in eine nicht-leere Menge abbildet, d.h. es gibt Individuen, die Instanz von dem Konzept sind.

Eine Menge von terminologischen Axiomen repräsentiert das terminologische Hintergrundwissen einer Anwendung und wird auch als TBox bezeichnet. Dabei wird zwischen einfachen (definitorischen) Axiomen und generellen Axiomen unterschieden. Bei einfachen Axiomen besteht deren linke Seite nur aus einem Konzeptnamen. Bei vielen Beschreibungslogiken wird vorausgesetzt, daß die Menge der terminologischen Axiome in der TBox zyklenfrei und eindeutig sind (siehe z.B. [Nebel, 1991] für eine Diskussion zur Semantik von Zyklen). Damit werden entweder Definitionen mit notwendigen und hinreichenden Bedingungen oder partielle Definitionen mit ausschließ-

lich notwendigen Bedingungen eingeführt. Bei generellen Axiomen zur Beschreibung von Konzeptinklusionen darf auf der linken Seite ein komplexer Konzeptterm stehen. Damit wird eine Erhöhung der Ausdruckskraft bei der Modellierung erreicht (siehe auch [Baader, 1990]).

Durch assertorische Axiome können auch Aussagen über einzelne Objekte getätigt werden, wobei eine Menge von assertorischen Axiomen als ABox bezeichnet wird. Da nicht nur Grundterme (Namen) in der ABox zugelassen sind, sondern allgemeine Konzeptterme (z.B. Disjunktionen) zur Beschreibung von Individuen verwendet werden können, läßt sich auch unvollständiges Wissen über Individuen repräsentieren. In Beschreibungslogiken wird eine Semantik der offenen Welt (open world semantics) realisiert.

Inferenzdienste: Ein beschreibungslogisches Repräsentationssystem stellt sowohl für TBoxen als auch für ABoxen eine Menge von Inferenzdiensten zur Verfügung, die hier ohne technische Details kurz aufgeführt werden. Wie schon erwähnt, ist der wichtigste Dienst der Erfüllbarkeitstest bzw. Konsistenztest für Konzeptterme. Bei der Erstellung eines terminologischen Modells einer Anwendung deutet die Inkonsistenz von bestimmten Konzeptnamen auf eventuelle Modellierungsfehler hin.

Ein weiter Dienst dient dazu, die Subsumptionsbeziehung zwischen Konzepttermen zu ermitteln. Die Subsumption von Konzepttermen ist für den Aufbau einer sog. Taxonomie (sog. Subsumptionshierarchie) bedeutsam. Für jeden Konzeptnamen (sog. atomares Konzept) werden automatisch die direkten Ober- und Unterkonzeptnamen bestimmt. Dies erfolgt auf der Basis der in der TBox aufgeführten Axiome zur Beschreibung des terminologischen Hintergrundwissens. Dieser Dienst wird auch TBox-Klassifikation genannt und kann in Anwendungen z.B. zur Strukturierung des Wissens in der Phase der Erstellung eines formalen Modells verwendet werden. Eine Verallgemeinerung des zuletzt angesprochenen Dienstes besteht darin, für jeden beliebigen Konzeptterm die direkten Ober- und Unterkonzeptnamen aus der TBox zu bestimmen (sog. Konzeptklassifikation). In Anwendungen kann dieser Dienst z.B. zur Abstraktionsbildung in Bezug auf das Hintergrundwissen eingesetzt werden.

Für viele Anwendungen ist es bedeutsam, Aussagen nicht nur über Konzepte (also über Mengen von Objekten) zu machen, sondern Aussagen über ganz bestimmte Individuen zu verwalten. Wie oben schon erwähnt, können Aussagen über Individuen durch assertorische Axiome formal deklariert werden. Der wichtigste Dienst ist die Überprüfung einer Menge von assertorischen Axiomen (also einer ABox) auf Konsistenz. Auch die Prüfung, ob ein Individuum eine Instanz eines gegebenen Konzeptes ist, kann für die Formalisierung von Anwendungsproblemen vielfach eingesetzt werden. Weiterhin kann die Menge der speziellsten atomaren Konzepte aus der TBox berechnet werden, von denen ein Individuum eine Instanz ist (auch Realisierung der ABox genannt). Dieser Inferenzdienst kann in z.B. in einem Anwendungkontext

zur bedingungsgesteuerten Auslösung von verschiedenen Aktionen verwendet werden. Grundlage für die Realisierung einer ABox sind die hinreichenden Bedingungen von Konzepten in der TBox. Sie bilden auch die Basis für den weiteren Dienst des Findens aller Individuen, die Instanz von einem gegebenen Konzept sind. Letzerer Dienst kann zur Recherche nach Individuen, die bestimmte Bedingungen erfüllen sollen, verwendet werden. Aus Sicht eines speziellen Individuums werden die über eine Rolle (Relation) zu diesem Objekt in Beziehung gesetzten Objekte als Füller dieser Rolle bezeichnet. Die Rollenfüller können ebenfalls über einen Inferenzdienst bestimmt werden.

Auf einige weitere Inferenzdienste (sog. Nicht-Standard-Inferenzdienste) soll kurz hingewiesen werden: Musteranpassung (matching) [Baader & Küsters, 2000], Termersetzung unter Berücksichtigung von Terminologien (rewriting using terminologies) [Baader et al., 2000] sowie Berechnung des kleinsten gemeinsamen Subsumierers (least common subsumer, LCS) [Cohen et al., 1992; Baader & Molitor, 2000]. Die Anwendungsgebiete dieser Inferenzdienste liegen z.B. in der Entdeckung von Redundanzen in Wissensbasen und in der Integration von verschiedenen Wissenbasen. Die LCS-Bestimmung kann beispielsweise zum induktiven Aufbau von Wissensbasen durch Beispiele verwendet werden. In dieser Arbeit wird die LCS-Bestimmung im Rahmen einer Anwendung zur beispielbasierten Informationsrecherche diskutiert.

Aus Sicht der Forschung gilt es, Inferenzdienste für möglichst ausdrucksstarke Beschreibungskonstrukte zu entwickeln, so daß für die Realisierung von Anwendungen Ad-hoc-Erweiterungen nicht notwendig sind.

Ausdrucksstarke Beschreibungslogiken: Die Beschreibungslogik $\mathcal{ALC}$ ist propositional vollständig, d.h. es wird die volle Negation unterstützt (daher der Name: Attributive Language with Complement). Es ist leicht zu sehen, daß in Sprachen, die bezüglich Komplementbildung abgeschlossen sind, die oben eingeführten Standard-Inferenzdienste auf ABox-Konsistenz zurückgeführt werden können. Das ABox-Konsistenzproblem für $\mathcal{ALC}$ ist (ohne TBoxen) PSPACE-vollständig [Schmidt-Schauss & Smolka, 1991]. Mit allgemeinen TBoxen liegt das Problem in EXPTIME.

Frühe Arbeiten zu Beschreibungslogiken [Donini et al., 1991b] untersuchten Teilsprachen von $\mathcal{ALC}$ mit polynomieller Zeitkomplexität für das Konzeptkonsistenzproblem (ohne allgemeine TBoxen). Teilsprachen von $\mathcal{ALC}$ werden auch unter dem Begriff *ausdrucksschwache* Beschreibungslogiken zusammengefaßt. Sprachen, die eine höhere Ausdrucksstärke als $\mathcal{ALC}$ besitzen, werden i.a. auch als *ausdrucksstark* bezeichnet. Die Begriffe dienen allerdings nur zur groben Charakterisierung der Logiken. Für ausdrucksstarke Beschreibungslogiken sind die Inferenzprobleme (im schlimmsten Fall) intraktabel.

Weitere, in der Literatur häufig diskutierte Operatoren zur Bildung von Konzept-

termen schränken die Anzahl von möglichen Rollenfüllern ein (z.B. $\mathcal{ALCN}$ [Donini et al., 1991a] mit einfachen Anzahlrestriktionen oder $\mathcal{ALCQ}$ [Hollunder & Baader, 1991] mit sog. qualifizierenden Anzahlrestriktionen). Einen Spezialfall der Anzahlrestriktion stellen sog. Attribute (features) dar. Attribute sind Rollen, die nur einen Füller bzgl. eines Individuums haben können. Zur Beschreibung von Rollen (bzw. Relationen) werden besondere Ausdrucksmittel bereitgestellt, z.B. Rollenhierachien, inverse Rollen, transitive Rollen. $\mathcal{ALC}$ mit transitiven Rollen wurde z.B. in [Sattler, 1996] untersucht. Algorithmen für den Konzeptkonsistenztest bzgl. Sprachen u.a. mit Attributen, Rollenhierarchien und transitiven Rollen wurden von [Horrocks, 1998] untersucht. Mit dem System FaCT wurde für diese Logik $\mathcal{ALCH}f_{R^+}$ eine optimierte Implementation entwickelt [Horrocks, 1997]. FaCT unterstützt allerdings keine ABox-Inferenzdienste.

ABox-Inferenzdienste mit vollständigen und korrekten Algorithmen sind ein zentraler Bestandteil dieser Habilitationsarbeit. Aufbauend auf dem Kalkül in [Buchheit et al., 1993b] für die Logik $\mathcal{ALCNR}$ wurde ein Kalkül für ABox-Konsistenz in der Logik $\mathcal{ALCNH}_{R^+}$, also $\mathcal{ALC}$ mit Anzahlrestriktionen, Rollenhierarchien und transitiven Rollen entwickelt [Haarslev & Möller, 1999c]. Dieser Kalkül dient als Basis für das hochoptimierte ABox-Inferenzsystem RACE. Wir kommen im nächsten Abschnitt darauf zurück.

Naturgemäß entwickelte sich während der Entwicklung von RACE der Stand der Kunst weiter. Ein kurzer Abriß soll hier die Darstellung abrunden. Die Interaktion von inversen Rollen mit qualifizierenden Anzahlrestriktionen und Rollenhierarchien (Logik $\mathcal{ALCQHI}_{R^+}$) bei der Bestimmung der Konzeptkonsistenz wird in [Horrocks & Sattler, 1999; Horrocks et al., 1999b] analysiert. Eine prototypische Implementierung (ohne ABox) wird in [Horrocks, 1999] aufgezeigt. Inzwischen wurde die Entscheidbarkeit des ABox-Konsistenzproblems sogar für die Logik $\mathcal{ALCQHI}_{R^+}$ gezeigt [Horrocks et al., 2000].[1] Eine Implementation liegt derzeit noch nicht vor. Weiterhin wurden im Kontext von ausdrucksstarken Beschreibungslogiken vollständige und korrekte Algorithmen für das Schließen mit Restriktionen für die Anzahl der Instanzen von Konzepten (und nicht nur Anzahlrestriktionen für Rollenfüller) sowie für Konzeptterme mit Individuen entwickelt [Tobies, 2000] (vgl. auch vorausgehende Arbeiten zu den beiden Themen [Baader et al., 1996] und [Schaerf, 1994]). Eine weitere Arbeit befaßt sich mit sog. symbolischen Anzahlrestriktionen bzw. arithmetischen Beziehungen zwischen Anzahlrestriktionen und stellt fest, daß alle Inferenzprobleme sogar schon unentscheidbar werden, wenn symbolische Anzahlrestriktionen mit ausdrucksschwachen Basislogiken kombiniert werden [Baader & Sattler, 1999]. Für eine detailliertere Übersicht über die Forschung im Bereich Beschreibungslogiken sei auf [Baader, 1999] und [Baader & Sattler, 2000] verwiesen.

[1] Die Autoren sprechen auch von einer sehr ausdrucksstarken Beschreibungslogik.

Optimierte Inferenzverfahren: In den vorherigen Abschnitt haben wir gesehen, daß viele Arbeiten sich auf die Erforschung von Repräsentationskonstrukten sowie auf die asymptotische Komplexität (worst case) der entsprechenden Inferenzverfahren konzentrieren (z.B. Verfahren für den Konsistenztest von Konzepttermen oder Verfahren für den Subsumptionstest zwischen Konzepttermen) [Donini et al., 1991a; Donini et al., 1997a]. Während die Forschungsarbeiten zu Beschreibungslogiken zunächst auf das Finden von traktablen (Teil-)sprachen (mit polynomieller Zeitkomplexität) fokussiert waren [Donini et al., 1991b; Cadoli, 1995], hat sich in jüngster Zeit eine neue Forschungsdisziplin etabliert, die sich damit beschäftigt, für im schlechtesten Fall exponentielle Inferenzprobleme geeignete domänenunabhängige Optimierungsverfahren und Heuristiken zu entwickeln, die es dennoch gestatten, die in realistischen Anwendungen verwendeten Konzeptterme „effizient“ zu verarbeiten [Baader et al., 1994; Ginsberg & McAllester, 1994; Freeman, 1995; Giunchiglia & Sebastiani, 1996; Horrocks, 1998; Horrocks & Patel-Schneider, 1999] (vgl. auch ähnlich motivierte Arbeiten im Kontext der Aussagenlogik, z.B. [Freeman, 1995; Truemper, 1998]). Der Begriff “effizient” soll nicht auf eine Optimierung auf der Bit-Ebene hindeuten, sondern bezieht sich auf die adaptive Auswahl von informierten Suchverfahren und die problembezogene Verwendung von zulässigen Heuristiken. In diesem neuen Forschungsgebiet sind in jüngster Zeit sehr motivierende Resultate im Bereich des Konsistenztests und der Klassifikation von TBoxen erzielt worden.

Aufbauend auf diesen Arbeiten (bzw. frühen Versionen hiervon) wurde für die ausdrucksstarke Beschreibungslogik $\mathcal{ALCNH}_{R^+}$ das Repräsentations- und Inferenzsystem RACE erstellt [Haarslev & Möller, 1999a; Haarslev & Möller, 1999d; Haarslev et al., 1999c]. RACE stellt zur Zeit eines der leistungsfähigsten implementierten Beschreibungslogiksysteme dar, das *ABox*-Schlußfolgerungen mit vollständigen, korrekten sowie terminierenden Algorithmen für eine Logik dieser Ausdruckskraft unterstützt [Haarslev & Möller, 2000b]. Auch bei den TBox-Inferenzdiensten gehört RACE zu den leistungsfähigsten Systemen, ist vielleicht sogar führend, da zur Zeit kein anderes System existiert, das auch für sehr große Wissensbasen (ca. 160.000 ggf. zyklische Axiome) eingesetzt werden kann und auf vollständigen und korrekten Algorithmen basiert [Haarslev & Möller, 2000c].

Einschränkungen über kontinuierlichen und diskreten Domänen: Für viele Anwendungen, insbesondere in technischen Bereichen, ist es nicht ausreichend, nur abstrakte Objekte (Symbole) zu betrachten. Vielmehr müssen beispielsweise auch reelle Zahlen in die konzeptuelle Modellierung einbezogen werden. Hierfür wurden im Bereich Beschreibungslogiken geeignete Konzeptbildungsoperatoren untersucht. Siehe hierzu insbesondere die Arbeiten zu $\mathcal{ALC}(\mathcal{D})$ [Baader & Hanschke, 1991a; Baader & Hanschke, 1992; Hanschke, 1993; Lutz, 1999b]. In $\mathcal{ALC}(\mathcal{D})$ können z.B. Prädikate über reelle Zahlen in das konzeptuelle Schließen integriert werden (eine Grundmenge

zusammen mit einer Menge von Prädikaten wird in $\mathcal{ALC}(\mathcal{D})$ als „konkrete Domäne“ $\mathcal{D}$ bezeichnet). In diesem Zusammenhang bedeutet „integriert“, daß sowohl beim Konsistenztest als auch beim Subsumptionstest von Konzepten die durch Prädikate der konkreten Domäne gegebenen Einschränkungen berücksichtigt werden.

Die Sprache $\mathcal{ALC}(\mathcal{D})$ enthält im Vergleich zu $\mathcal{ALC}$ zwei neue Elemente. $\mathcal{ALC}(\mathcal{D})$ gestattet die Deklaration von Attributen (s.o.) und erlaubt die Verwendung von Attributketten in einem neuen konzeptbildenden Operator (prädikatbasierte Existenzrestriktion). Mit diesem Operator kann die Existenz einer Sequenz von Attributketten gefordert werden, an deren Ende sich jeweils ein sog. konkretes Objekt befindet. Konkrete Objekte (auch Variablen genannt) stehen, je nach gewählter Grundmenge der konkreten Domäne, z.B. für eine reelle Zahl. Die Wertebereiche der Variablen, deren Existenz durch eine prädikatbasierte Existenzrestriktion gefordert werden kann, können durch Prädikate in Existenzrestriktionen eingeschränkt werden. Mittels einer ABox können Einschränkungssysteme „aufgespannt“ werden.

Obwohl mit $\mathcal{ALC}(\mathcal{D})$ aus Sicht der Anwendung im Gegensatz zu $\mathcal{ALC}$ eine erheblich erhöhte Ausdruckskraft bereitsteht und sich viele weitere praktische Probleme lösen lassen, sind schon frühzeitig mögliche Erweiterungen untersucht worden. Wie sich herausstellte [Baader & Hanschke, 1992], führt die Integration von weiteren beschreibungslogischen Sprachkonstrukten in $\mathcal{ALC}(\mathcal{D})$ ohne Beschränkungen der Kombinierbarkeit von Sprachkonstrukten schnell zur Unentscheidbarkeit der Repräsentationssprache (siehe auch [Baader & Sattler, 1998]). Die Erweiterung von $\mathcal{ALC}(\mathcal{D})$ um einen Operator für den transitiven Abschluß von Rollen ist beispielsweise unentscheidbar für allgemeine konkrete Domänen. Obwohl für praktische Anwendungen gegebenenfalls naheliegend, sind also Erweiterungen kritisch in Bezug auf Entscheidbarkeit.

In dieser Arbeit werden konkrete Domänen für die Sprache $\mathcal{ALCNH}_{R^+}$ (s.o.) untersucht. Durch syntaktische Einschränkung der prädikatbildenden Existenzrestriktion auf die Verwendung von ausschließlich Attributen (im Gegensatz zu Attributketten) wurde eine Sprache definiert ($\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$), für die durch Vorstellung eines vollständigen und korrekten (und terminierenden) Kalküls gezeigt werden konnte, daß die Standard-Inferenzprobleme entscheidbar sind. Die Anwendungen von $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$ sind vielfältig. In dieser Arbeit werden mit $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$ insbesondere Konstruktionsaufgaben gelöst.

Probabilistisches konzeptuelles Schließen

Arbeiten zur Integration von probabilistischen Informationen in beschreibungslogische Modelle werden in dieser Arbeit im Kontext von Anwendungen zur Informationsrecherche untersucht. Ein Ansatz zur probabilistischen Abstraktion von Konzepttermen wird in diesem Rahmen zur beispielbasierten Informationssuche vorgestellt

(siehe auch [Mantay & Möller, 1998; Mantay et al., 1999]).

Die Arbeiten basieren auf einer Erweiterung der Beschreibungslogik $\mathcal{ALN}$ zur Repräsentation von Vagheiten und Unsicherheiten in Beschreibungslogiken (sog. probabilistische Beschreibungslogiken) [Koller et al., 1997]. Auf andere, in diesem Kontext relevante und auf der Fuzzy-Set-Theorie basierende Beschreibungslogiken [Straccia, 1998; Tresp & Molitor, 1998] soll hier nicht weiter eingegangen werden. Eine Implementation der in [Koller et al., 1997] vorgestellten probabilistischen Logik wurde von [Kaplunova, 1999] erstellt und im Projekt „BAND“ des Labors für Künstliche Intelligenz evaluiert [Möller et al., 1998; Mantay & Möller, 1998].

Räumliches konzeptuelles Schließen

Nachdem in den vorangegangenen Abschnitten Beschreibungslogiken und ihre bisher erfolgten Erweiterungen dargestellt wurden, wenden wir uns jetzt speziell dem Forschungsstand auf dem Gebiet des räumliches Schließens zu. In dieser Arbeit wird aufgezeigt, daß das konzepuelle Schließen mit Beschreibungslogiken eng mit dem räumlichem Schließen gekoppelt werden kann (und muß). Die Kopplung erfolgt auf der Basis der Semantik der Repäsentationskonstrukte und stellt also nicht nur eine rein softwaretechnische Kopplung von Inferenzkomponenten dar.

Obwohl der Bereich des zeitlichen Schließens eine ebenso große Bedeutung hat und ebenfalls mit dem konzeptuellen Schließen in systematischer Weise verknüpft werden sollte, wird hier exemplarisch der Bereich des räumlichen Schließens, insbesondere des qualitativen räumlichen Schließens, vertiefend dargestellt.

Für Geo-Informationssysteme (vgl. [Bartelme, 1995]) wäre es beispielsweise interessant auszusagen, daß sich ein bestimmtes Objekt „innerhalb“ eines anderen Objektes befindet. Durch die qualitative Angabe des Ortes braucht die Position nicht in quantitativen Koordinaten angegeben zu werden. Diese sind eventuell nicht bekannt oder schwer zu beschaffen (siehe auch [Frank, 1997]). Weiterhin kann es bedeutsam sein, für alle Objekte „innerhalb“ eines bestimmten Objektes bestimmte Aussagen zu machen, ohne die enthaltenen Objekte explizit aufzuzählen. Zum Beispiel können alle „Regionen“ in einer „Stadt“ als „Bezirke“ ausgewiesen werden. Diese Beispiele verdeutlichen die Modellierungsmöglichkeiten mit beschreibungslogischen Konstrukten und lassen erkennen, daß der Kombination von räumlichen und beschreibungslogischen Schlüssen eine wichtige Bedeutung zukommt.

Qualitatives räumliches Schließen: Der Bereich des „qualitativen räumlichen Schließens“ ist ein Teilgebiet des „qualitativen Schließens“. Beim qualitativen Schließen wird versucht, das (qualitative) Verhalten von Systemen auf durch Verwendung von Begrifflichkeiten wie z.B. „zu hoch“, „zu niedrig“ und nicht durch absolute Wer-

tebereiche zu modellieren. Mit qualitativen Schlußverfahren werden dann aus gegebenen, expliziten Informationen neue, implizite Informationen gewonnen. In den letzten Jahren hat sich das „qualitative räumliche Schließen“ als ein eigenständiges Teilgebiet etabliert (siehe [Stock, 1997] für eine Übersicht). Da der Raum mehrdimensional und damit nicht adäquat durch einzelne symbolisch repräsentierte Größen modellierbar ist, können viele Erkenntnisse aus dem Bereich des zeitlichen qualitativen Schließens nur bedingt auf räumliche Problemstellungen übertragen werden. Eine wichtige Herausforderung für das qualitative räumliche Schließen besteht darin, Kalküle mit guten Repräsentationsmöglichkeiten und Inferenzleistungen für räumliche Größen zu ermöglichen (vgl. hierzu auch [Frank, 1997]). Vielfache Verwendung hat der RCC-8 Kalkül zur Repräsentation von topologischen Beziehungen gefunden [Randell et al., 1992].

Topologische Beziehungen zwischen räumlichen Entitäten sind für viele Anwendungen bedeutsam. Wir haben schon betont, daß in diesem Zusammenhang eine Integration des konzeptuellen, begrifflichen Schließens und des Schließens über räumliche Relationen notwendig ist. Dies sei an einem sehr einfachen Beispiel erläutert. Wir nehmen an, ein „Ferienhaus_mit_Angelmöglichkeit“ sei definiert als ein Ferienhaus, an das eine Wasserfläche direkt angrenzt. Weiterhin sei ein Konzept „Mückenfreier_Wald“ definiert als ein Wald, wobei zusätzlich gilt, daß alle Areale, die nicht separat zum Wald liegen, keine Wasserflächen sind. Durch Inferenztechniken sollte es möglich sein, automatisch zu erkennen, daß z.B. ein Wunschferienhaus „Ferienhaus_mit_Angelmöglichkeit“, das sich in einem „Mückenfreien_Wald“ befindet, nicht existieren kann. Ohne ein Schließen über topologische Relationen in Zusammenhang mit der Formulierung von Konzeptwissen ist dieses nicht möglich.

Die bisher entwickelten Inferenzverfahren für qualitative räumliche Relationennetze basieren überwiegend auf sog. Kompositionstabellen, die für Paare von Relationen R_i und R_j, wobei $R_i(a,b) \wedge R_j(b,c)$ gilt, alle möglichen Beziehungen zwischen a und c in Form einer Disjunktion angeben. Durch Rückführung des Konsistenzproblems für bestimmte Formeln einer intuitionistischen Logik auf das Konsistenzproblem für Relationennetze konnte in [Bennett, 1994] die Entscheidbarkeit des Konsistenzproblems von RCC-8-Netzen gezeigt werden. In [Bennett et al., 1997] wurde untersucht, wann das Pfadkonsistenzverfahren vollständig für eine gegebene Raumtheorie ist. Für Netze, an deren Kanten nur Basisrelationen aus RCC-8 stehen, wurde die Traktabilität des Konsistenzproblems bewiesen [Nebel, 1995]. Falls allerdings beliebige Disjunktionen von Relationen zugelassen werden, ist die Prüfung der Erfüllbarkeit von RCC-8-Netzen ein Inferenzproblem, daß nicht nicht mehr traktabel ist [Renz & Nebel, 1997]. Die Ermittlung einer global konsistenten Lösung kann durch Anwendung eines Backtracking-Verfahrens erfolgen. In [Renz & Nebel, 1997] werden maximal traktable Teilmengen von RCC-8-Relationen vorgestellt, die zur Optimierung eines vollständigen Konsistenztests ausgenutzt werden können (Partitionierung

der Relationennetze). Geometrische Realisierungen von Relationennetzen wurden in [Renz, 1998] untersucht. RCC-8 gehört zu den gut erforschten Gebieten des qualitativen räumlichen Schließens. In dieser Habilitationsarbeit wird daher vielfach auf RCC-8 zurückgegriffen, jedoch wird RCC-8 nur als ein Beispiel für einen Formalismus zum räumlichen Schließen verstanden. Die in dieser Arbeit erzielten Ergebnisse sind allerdings nicht an qualitative Formalismen gebunden, sondern lassen sich auch auf quantitative und andere qualitative Repräsentationsstrukturen verallgemeinern.

Mit den Relationennetzen können räumliche Beziehungen zwischen Entitäten gut repräsentiert werden. Ein Defizit besteht jedoch darin, daß bei den bisherigen Ansätzen kein konzeptuelles Wissen über die zueinander in Beziehung gesetzten Objekte in die Schlüsse einbezogen werden kann.

Beschreibungslogisches und räumliches Schließen: In [Haarslev et al., 1994] wurde erstmals ein Ansatz zum konzeptuellen Schließen über räumliche Objekte mithilfe von Beschreibungslogiken vorgestellt. Dieser Ansatz integriert quantitative, qualitative und konzeptuelle Information über Raumdomänen durch Benutzung „generativer qualitativer Relationen“, die die Integration spezieller Raumindizierungstechniken ermöglichen. Im logischen Sinne sind die räumlichen Relationen allerdings noch als primitiv anzusehen, denn topologische Schlüsse über räumliche Relationen werden bei der Terminologiebildung noch nicht adäquat berücksichtigt (vgl. den Abschnitt zu den beschreibungslogischen Inferenzdiensten). Obwohl die Modellierung mit objektzentrierten logischen Repräsentationsformalismen wie Beschreibungslogiken eine lange Tradition hat, wurde erst in jüngster Zeit die systematische Kombination von konzeptuellen und räumlichen bzw. raum-zeitlichen Repräsentationsformalismen näher untersucht (vgl. z.B. [Haarslev & Möller, 1997a]). Die Arbeiten sind durch aktuelle Anwendungsgebiete wie z.B. Geo-Informationssysteme motiviert. In diesen Systemen ist eine systematische Integration der konzeptuellen Wissensrepräsentation und des räumlichen bzw. raum-zeitlichen Schließens besonders wünschenswert.

In [Haarslev & Möller, 1997a] wird erstmals auf der Basis der Beschreibungslogik Classic [Brachman et al., 1991] eine Erweiterung zur Integration räumlicher Relationen unter Berücksichtigung ihrer Semantik diskutiert und ein entsprechend erweiterter Konzeptoperator vorgeschlagen. Schlüsse über räumliche Relationen werden mit dem Ansatz auch für die Klassifikation von TBoxen durchgeführt. Die theoretischen Ergebnisse werden in [Haarslev, 1998] auf allgemeine visuelle Notationen (z.B. ER-Diagramme, Petrinetze, etc.) angewendet.

Um eine adäquatere Behandlung von Beschreibungsformen für räumliches und raumzeitliches Wissen zu ermöglichen, wurde die Theorie zu $\mathcal{ALC}(\mathcal{D})$ erweitert [Lutz & Möller, 1997; Möller et al., 1997; Haarslev et al., 1998; Haarslev et al., 1999b]. Durch Prädikate über konkreten Domänen können räumliche (und auch zeitliche) Relationen als *definierte Rollen* eingeführt werden. Durch definierte Rollen wird

eine erhebliche Erhöhung der Ausdruckskraft gegenüber $\mathcal{ALC}(\mathcal{D})$ erreicht. Schlußfolgerungen über topologische Beziehungen zwischen räumlichen Objekten lassen sich in das konzeptuelle Schließen integrieren (siehe [Haarslev et al., 1999b]).

Für die Beschreibungslogik $\mathcal{ALCRP}(\mathcal{D})$ konnte die Entscheidbarkeit bewiesen werden, sofern syntaktische Einschränkungen in der Konzeptbeschreibungssprache hingenommen werden [Lutz et al., 1997; Lutz, 1998; Haarslev et al., 1998]. Ohne Syntaxeinschränkungen bleibt das Konsistenzprüfungsverfahren vollständig und korrekt, aber seine Terminierung kann nicht mehr garantiert werden [Lutz & Möller, 1997]. Aus Sicht des Stands der Forschung wird deutlich, daß $\mathcal{ALCRP}(\mathcal{D})$ – entsprechend instanziiert z.B. mit einer konkreten Domäne mit Prädikaten für RCC-8 – somit eine erste *entscheidbare* Raumlogik erster Ordnung ist. Das oben skizzierte Ferienhausbeispiel läßt sich mit $\mathcal{ALCRP}(\mathcal{D})$ auf einfache Weise formalisieren. Die angesprochene Inkonsistenz des Wunschferienhauses wird durch den Kalkül aus [Haarslev et al., 1998] entdeckt. Komplexere Anwendungsbeispiele zur Belegung der Ausdruckskraft von $\mathcal{ALCRP}(\mathcal{D})$ finden sich in [Haarslev et al., 1999b].

In dieser Arbeit wird die Anwendung von $\mathcal{ALCRP}(\mathcal{D})$ auch zur Modellierung von zeitlichen und raum-zeitlichen Phänomenen aufgezeigt (siehe auch [Lutz et al., 1997; Haarslev et al., 1999b]). Auf die Darstellung der umfangreichen Arbeiten zur Modellierung von zeitlichen Relationen z.B. mit Allens Zeitlogik soll hier allerdings verzichtet werden (siehe [Stock, 1997] und [Frank, 1998]).

Ermangelungsschließen in Beschreibungslogiken: Es wurden im Kontext von beschreibungslogischen Modellierungssystemen sowohl theoretische als auch praktische Arbeiten zur Repräsentation von Regeln für das Ermangelungsschließen (defaults) durchgeführt, um damit die Repräsentation von unvollständigem Wissen zu unterstützen. Da das Gebiet des Ermangelungsschließens selbst sehr groß ist (vgl. [Antoniou, 1997; Schaub, 1997] für zusammenfassende Arbeiten), wird hier nur auf Arbeiten eingegangen, die in direktem Zusammenhang mit Beschreibungslogiken stehen. Zu den Erweiterungen von beschreibungslogischen Systemen gehören Techniken des Ermangelungsschließens unter Verwendung von Ermangelungsregeln (Regeln mit zusätzlichen Konsistenzbedingungen) zur Repräsentation von Standardannahmen [Baader & Hollunder, 1992; Padgham & Zhang, 1993; Padgham & Nebel, 1993], Techniken zur Behandlung von Standardannahmen mit Prioritäten [Baader & Hollunder, 1993] sowie Techniken zum Schließen unter der Annahme der abgeschlossenen Welt (closed world assumption) [Weida, 1996]. Relevant sind in diesem Zusammenhang auch die Arbeiten zur Formalisierung von vorwärtsverkettenden Regelsystemen [Donini et al., 1992; Donini et al., 1994; Donini et al., 1998]. Im Zusammenhang mit konkreten Domänen und $\mathcal{ALC}(\mathcal{D})$ wurden Regelsysteme in [Hanschke, 1993] untersucht.

Ermangelungsschlüsse in Bezug auf konzeptuelle und räumliche Relationen werden

für zukünftige Anwendungen beispielsweise im Bereich der Geo-Informationssysteme als besonders wichtig eingestuft. Wenn z.B. nichts über die Position des Rathauses in einer Stadt bekannt ist, so sollte als Standardannahme gelten, daß sich das Rathaus im Zentrum der Stadt befindet, es sei denn, es spricht etwas dagegen. Die Behandlung von räumlichen und konzeptuellen Informationen durch Ermangelungsschlüsse in Beschreibungslogiken wurde zuerst in [Möller & Wessel, 1999; Möller et al., 1999b] untersucht.

Anwendungen

Anwendungen des räumlich-terminologischen Schließens mit Defaults im Bereich der Spezifikation von visuellen Notationen und Anfragesprachen sind in [Haarslev et al., 2000b] beschrieben. Beschreibungslogische Repräsentationssysteme können in der Praxis vielfältige weitere Anwendungen finden. Um das Potential des entwickelten Systems RACE einerseits sowie auch die umfangreichen Möglichkeiten der vorgeschlagenen theoretischen Erweiterungen andererseits zu verdeutlichen, werden in dieser Arbeit Anwendungen aus verschiedenen Kontexten vorgestellt. Wir betrachten insbesondere:

- Überprüfung von Spezifikationen von Telekommunikationsanlagen [Areces et al., 1999],
- Entwicklung von großen Ontologien in der Medizin- bzw. Bioinformatik [Schulz & Hahn, 2000],
- Konstruktion von technischen Geräten,
- Wissensbasierte Bildverarbeitung,
- Deduktive Informationssysteme mit Einsatzperspektiven im elektronischen Handel.

Die ersten beiden Anwendungen wurden von anderen Forschergruppen untersucht (op. cit.). Sie dienen zur Evaluierung des DL-Systems RACE. Die weiteren Anwendungen motivieren die Verwendung von formalen Inferenzsystemen im Kontext von Konstruktionsaufgaben, Interpretationsaufgaben und Rechercheaufgaben in kooperierenden Informationssystemen [Papazoglou & Schlageter, 1998].

Beschreibungslogiken haben sich in verschiedenen Arbeiten als ein vielversprechendes Werkzeug zur Informationsintegration herausgestellt (vgl. [Catarci & Lenzerini, 1993; Levy et al., 1996; Calvanese et al., 1998; Kashyap & Sheth, 1998; Klusch, 1998]). Die in dieser Habilitationsschrift zusammengefaßten Forschungsergebnisse zeigen neue

Möglichkeiten und Anwendungsgebiete auf. Als Beispiel betrachten wir die deduktive Zuordnung von Werbungen bei einem Fernsehprogramm-Auskunftssystem in einem verteilten Szenario [Möller et al., 2001].

Zusammenfassung

Zur Entwicklung eines formalen Repräsentations- und Inferenzsystems wurde in dieser Arbeit ein berechnungsorientierter Ansatz verfolgt. Es wurden Fortschritte sowohl im theoretischen als auch im praktischen Bereich erzielt. Die Zusammenfassung der Arbeit schließt mit einer Einschätzung des Erreichten und zeigt mit einem Ausblick einige der möglichen Erweiterungen zu den hier präsentierten Arbeiten auf.

Einschätzung

Mit dem System RACE wurde ein sehr mächtiges Repräsentations- und Inferenzsystem entwickelt. Die Architektur des RACE-Systems ist auf die Lösung von Anwendungsproblemen mit großen TBox und ABoxen zugeschnitten. RACE stellt neue Optimierungs- und Inferenztechniken bereit, so daß größere Anwendungsprojekte von beschreibungslogischen Schlüssen, die auf vollständigen und korrekten Algorithmen basieren, profitieren können. Aufgrund der engen Beziehungen zwischen Beschreibungs- und Modallogiken [Schild, 1991a] kann RACE auch als Modallogik-Theorembeweiser verwendet werden.

Das Ziel der Entwicklung von RACE war es, eine Architektur zu entwickeln, die gute Performanz im Bereich des modallogischen Schließens und gute Performanz im Bereich des TBox- und ABox-Schließens zeigt. Wie die empirischen Untersuchungen belegen, müssen in diesem Kontext Kompromisse gemacht werden. Für spezielle kombinatorische Probleme, die als Konsistenztest für bestimmte Typen von logischen Formeln kodiert werden (z.B. Formeln der Modallogik K_m), sind spezialisierte Beweiser entwickelt worden, die für diese Formelklasse etwa zweimal bis dreimal schneller sind als RACE. Allerdings können sie nur eingesetzt werden, wenn die Probleme mit den speziellen Logiken repräsentiert werden können. In vielen Anwendungen werden jedoch ausdruckstarke Konstrukte wie z.B. Anzahlrestriktionen zur Modellierung benötigt. Anzahlrestriktionen bedingen allerdings komplexere Algorithmen für den Konfliktest in Tableaubeweisern. Daher werden in RACE die Algorithmen adaptiv zur Eingabe automatisch ausgewählt, so daß für Sprachen ohne Anzahlrestriktionen keine Performanzeinbußen zu erwarten sind. RACE ist auch heute noch

eines der schnellsten Inferenzsysteme, die für Beschreibungslogiken verfügbar sind. Die Entwicklung von RACE ist noch nicht abgeschlossen, d.h. Optimierungtechniken für ausdruckstärke Konstrukte, wie z.B. inverse Rollen befinden sich derzeit in der Entwicklung und werden in RACE integriert. Man darf allerdings nicht außer Acht lassen, daß mit der Einführung von Optimierungstechniken die Integration neuer Sprachkonstrukte erheblich schwieriger wird. Obwohl also eventuell für die „neuen“ Sprachkonstrukte schon Entscheidungsverfahren vorliegen, kann eine Implementierung für größere Anwendungen nur sinnvoll vorgenommen werden, wenn bekannt ist, wie die neuen Konstrukte im Zusammenhang mit Optimierungs- und Cachingtechniken behandelt werden können.

Ein weiterer Teil dieser Arbeit betrachtete das Zusammenspiel zwischen räumlichem und konzeptuellem Schließen. Mit der Logik $\mathcal{ALCRP}(\mathcal{D})$ bzw. der speziellen Instanz $\mathcal{ALCRP}(\mathcal{RCC})$ wurde ein Vorschlag zur Integration des konzeptuellen und räumlichen Schließens am Beispiel von qualitativen topologischen räumlichen Beziehungen gemacht. Inzwischen wurde eine prototypische Implementierung erstellt [Turhan, 1998]. Es muß allerdings konstatiert werden, daß bislang noch keine optimierte Implementierung von $\mathcal{ALCRP}(\mathcal{D})$ (und $\mathcal{ALC}(\mathcal{D})$) existiert, die im Kontext von $\mathcal{ALC}$-Termen mit der Leistung von RACE vergleichbar wäre. Allerdings bieten die Resultate der Entwicklung von RACE eine gute Ausgangsbasis für die Entwicklung von neuen Optimierungstechniken für $\mathcal{ALC}(\mathcal{D})$ und $\mathcal{ALCRP}(\mathcal{D})$ (siehe [Turhan, 2000] für erste Ergebnisse).

Es sei allerdings betont, daß die mit $\mathcal{ALCRP}(\mathcal{D})$ und der konkreten Domäne RCC-8 möglichen Repräsentationen im Bereich des räumlich-konzeptuellen Schließens nur als ein erster Schritt aufzufassen sind. Einerseits lassen sich weitere Phänomene des räumlichen Schließen mit anderen konkreten Domänen modellieren. Andererseits wäre es auch wünschenswert, die mit $\mathcal{ALCRP}(\mathcal{D})$ eingeführten Rollenoperatoren im Kontext von $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$ zu untersuchen. $\mathcal{ALCRP}(\mathcal{D})$ bietet zwar definierte Rollen, jedoch werden Anzahlrestriktionen nicht unterstützt und Rollenhierachien bzw. transitive Rollen können nur über konkrete Domänen eingeführt werden.

Die im Kontext von Informationsrechercheanwendungen untersuchten Beispiele zeigen, daß die entwickelten Logiken eine gute theoretische Basis für den Aufbau von Informationssytemen mit Inferenzkomponente darstellen (sog. Inferenz-basierte Informationssysteme oder auch deduktive Informationssysteme). Mit Hilfe von Beschreibungslogik-Formalismen wurden Lösungen zur Realisierung einer Inferenzkomponente in einem verteilten System vorgeschlagen, in dem mehrere unterschiedliche Repräsentationssprachen und ihre Schlußkomponenten integriert werden können. Weiterhin werden im Kontext von Abstraktionsoperatoren zur Analyse von Anfragen auch probabilistische Inferenzkomponenten wirkungsvoll eingesetzt (siehe [Kaplunova, 1999] für Details der Implementation der Inferenzalgorithmen).

Der in dieser Arbeit verfolgte Ansatz kann als Ergänzung zu anderen Ansätzen ge-

sehen werden, welche die Beschreibungslogiklandschaft z.B. aus Sicht der Komplexitätstheorie betrachten [Donini et al., 1997a]. Eine Übersicht über viele andere Arbeiten mit wichtigen Resultaten, die auf vielfältige Weise diese Arbeit beeinflußt haben, bieten auch [Baader, 1999] und [Baader & Sattler, 2000]. Beschreibungslogiken bilden die Basis für eine Vielzahl von Projekten in einer sehr lebendigen Forschungslandschaft.

Ausblick

Es wurde betont, daß qualitative räumliche Relationen auf Basis des RCC-8-Kalküls nur als ein Beispiel für räumliches Schließen angesehen werden kann. Topologische Relationen eignen sich nur zur Repräsentation ganz bestimmter Beziehungen zwischen Objekten. Wie Schlußfolgerungen über Form, Position und Orientierung gewinnbringend in den Rahmen des konzeptuellen Schließens integriert werden, ist ein noch offenes Forschungsgebiet. Weiterhin ist die Integration von qualitativem und quantitativem Schließen in diesem Zusammenhang ein wichtiges Forschungsfeld. Die Interaktion zwischen qualitativen und metrischen Prädikaten wurde im räumlichen Bereich bisher kaum untersucht (siehe aber [Kautz & Ladkin, 1991; Meiri, 1996] für die Interaktion von qualitativen und metrischen Prädikaten über Zeitstrukturen). Obwohl bewiesen wurde, daß für die konkrete Domäne der reellen Zahlen ein Erfüllbarkeitsalgorithmus für Ungleichungen über ganzzahlige Polynome existiert [Tarski, 1951], konnte das Ergebnis bisher nicht praktisch umgesetzt werden. Für praktische Anwendungen ist ein effektives Verfahren notwendig, das bislang noch nicht in einem Beschreibungslogiksystem realisiert wurde. Erst in jüngster Zeit sind theoretische Resultate verfügbar geworden, die eine Implementierung eines Entscheidungsalgorithmus ermöglichen (siehe die Theorie der zylindrisch-algebraischen Dekomposition [Caviness & Johnson, 1998]), so daß räumlich-metrische Einschänkungen auf Erfüllbarkeit getestet werden können. Die Arbeiten hierzu sind allerdings noch in einen Beschreibungslogikkontext zu integrieren.

Ein weiteres noch offenes Problem ist die Erweiterung der bestehenden, hochoptimierten Inferenzalgorithmen in Bezug auf den Umgang mit konkreten Domänen im allgemeinen und räumlichen Informationen im speziellen. Relevant sind in diesem Kontext z.B. Arbeiten zur inkrementellen Erfüllung von Beschränkungen. Ebenfalls unklar ist bislang die Optimierung von Revisionsstrategien [Nebel, 1990a], insbesondere für räumliches Wissen. Bei inkrementeller Hinzufügung von assertorischen Axiomen läßt sich beispielsweise die Bestimmung der speziellsten atomaren Konzepte, von denen ein Individuum eine Instanz ist, eventuell mit deutlich weniger Aufwand inkrementell durchführen.

Eine Verallgemeinerung der Arbeiten zum Defaultschließen wurde durch die Einführung von autoepistemischen Operatoren in Beschreibungslogiken erreicht (siehe

[Donini et al., 1997b; Rosati, 1997]). Durch autoepistemische Operatoren können z.B. auch Integritätsbedingungen (Bedingungen an das Datenmodell und nicht an die repräsentierte Welt) ausgedrückt werden. Eine Untersuchung dieser Operatoren im Kontext von räumlich-terminologischen Schlüssen ermöglicht die Formalisierung von nicht-monotonen Schlüssen mit autoepistemischen Operatoren auch in diesem Bereich.

Mit der Verfügbarkeit von Inferenzsystemen für ausdrucksstarke Beschreibungslogiken wird es z.B. möglich, für wichtige Teile der Sprache UML (Unified Modeling Language, e.g. [Page-Jones, 2000]) eine formale Semantik zu definieren, so daß UML nicht nur zur Präsentation und Kommunikation von Objektmodellen innerhalb der Designphase einer Anwendung verwendet werden kann, sondern auch zur Lösung von Problemen eingesetzt werden kann, ohne daß Code für Spezialalgorithmen geschrieben werden muß. Arbeiten zur Verifikation von UML-Zustandsraum-Modellen mit Ansätzen der Modellüberprüfung finden sich etwa in [Lilius & Paltor, 1999c; Lilius & Paltor, 1999b; Lilius & Paltor, 1999a].

Probabilistische Erweiterungen von Beschreibungslogiken sind bisher noch nicht in Bezug auf die Modellierung von räumlichem Wissen untersucht worden. Wenn also im Falle unseres Rathausbeispiels wiederum keine Position bekannt ist, so könnte mit Techniken zur Behandlung von Unsicherheiten der angenommenen Position des Rathauses in der Mitte der Stadt eine höhere Wahrscheinlichkeit zugewiesen werden als etwa einer Position am Stadtrand. Einsatz und Weiterentwicklung dieser Techniken im Kontext der Repräsentation von räumlichem und zeitlichem Wissen in Beschreibungslogiken wurden bislang noch nicht eingehend untersucht, bieten aber beispielsweise für GIS-Modellierungen beträchtliches Potential.

Es wird argumentiert, daß der in dieser Arbeit verfolgte logische Ansatz eine Grundlage für Informationssysteme mit Inferenzdiensten bildet. ABoxen können im Prinzip zur Repräsentation von Informationen über bestimmte Objekte einer Domäne verwendet werden. Solange allerdings ABoxen im Hauptspeicher verwaltet werden, ist das Einsatzspektrum dieser Technologie auf bestimmte Anwendungen mit eingeschränktem Datenvolumen beschränkt. Es ist daher sehr vielversprechend, für ABox-Schlüsse Inferenzalgorithmen zu implementieren, die persistente Daten mit Transaktionen und Rücksetzmechanismen unterstützen (vgl. z.B. [Borgida & Brachman, 1993; Borgida, 1995] für erste Ansätze). Es sollte allerdings auch darauf hingewiesen werden, daß die Revision von logischen Assertionen, insbesondere im Kontext von räumlich-terminologischem Wissen, für sich genommen noch ein aktives Forschungsfeld darstellt.

Résumé

Mit der Arbeit konnten Grundlagen für ein Repräsentationsmedium entwickelt werden, so daß mit automatischen Inferenzdiensten Lösungen für (Teil-)Probleme einer Anwendung berechnet werden können. Dadurch ist eine kosteneffizientere Entwicklung von robusten und anpaßbaren Anwendungen mit erweiterter Funktionalität möglich. Die Einsichten, die die Entwicklung von optimierten Algorithmen vermitteln, verdeutlichen, daß die Dienste, die durch automatische Inferenzsysteme erbracht werden, nicht ohne weiteres durch Ad-hoc-Techniken der Softwareentwicklung realisiert werden können. Wissenschaftliche Arbeit ist notwendig, um Beweistechniken anzuwenden, so daß Korrektheit, Vollständigkeit und Terminierung auf der Ebene der Algorithmen gezeigt werden kann. Dedizierte Programmentwicklungs- und Testtechniken garantieren, daß die Implementation dieser Algorithmen möglichst fehlerfrei ist. Einmal entwickelt, können Inferenzsysteme in verschiedenen Anwendungskontexten wirkungsvoll zum Einsatz kommen.

Formale Inferenzsystemen und speziell auch Beschreibungslogiken können als sich wandelnde Disziplinen angesehen werden. Vor einigen Jahren noch bestand eine große Kluft zwischen Theorie und Praxis. Mit der Entwicklung von Optimierungstechniken jedoch zeigt sich, daß theoretische Ergebnisse bezüglich Entscheidbarkeit und Komplexität direkt bedeutsam sind für die Implementierung von praktischen Inferenzsystemen. Die Notwendigkeit von Optimierungstechniken in einer praktischen Implementierung stellt neue Anforderungen an theoretische Arbeiten bzgl. Beweistechniken, Analyse von Durchschnittsfällen usw. Neue Optimierungstechniken, die in RACE implementiert sind und auf der Basis von theoretischen Ergebnisse entwickelt wurden, ermöglichen die Realisierung von neuartigen Anwendungen.

Nicht zuletzt die aufgezeigten Anwendungsbeispiele belegen die Signifikanz der in dieser Arbeit erzielten Ergebnisse für die Wissensrepäsentation und auch die Informatik generell. Vieles wurde erreicht, doch vieles mußte offen bleiben. Ich hoffe, daß durch diese Arbeit neue Forschungsarbeiten mit neuen Resultaten angeregt werden.

Part II

Main Part

Chapter 1

Introduction

The problems which application developers are facing in projects involving information technology tend to become more and more complex. In many applications – in particular in the web context – models based on large ontologies are more and more important. Moreover, aspects of distribution as well as safety and verification concerns have to be considered. In order to reduce complexity, topics such as knowledge-based systems, object-oriented frameworks, unified modeling languages, workflow models, business (re)organization models etc. dominate the discussion about how to ensure that computer systems can be constructed which, on the one hand, match the requirements and, on the other hand, are reliable, robust and adaptable. In order to fulfill these requirements there is a huge demand for systems that automatically solve complex (sub)problems of applications based on declarative models of the domain. In addition, the verification of models used for computational purposes is often required in some application scenarios. The formalization of subtasks of applications as well-understood computational inference problems is a prerequisite to support the development of more powerful systems with better quality but less development costs. It should be emphasized, however, that a user of an application need not be aware of internal computational processes being based on formal models. For instance, users can successfully interact with web-based information retrieval systems based on database technology without being aware of the underlying relational calculus.

This Habilitation Thesis describes the results of practical and theoretical investigations for developing representation systems which support the construction of declarative models, which can be checked for consistency and which can be used as the basis for problem solving processes in different application contexts. In order to support automatic processing of models, the modeling language must be based on a clear semantics such that inference problems can be formally defined, inference algorithms can be systematically analyzed, and formal inference systems can be suc-

cessfully developed [Genesereth & Nilsson, 1987; Fitting, 1996; Poole et al., 1998]. Formal inference systems are of particular importance in safety-critical applications in order to check system specifications before systems are actually deployed in practice. Based on a formal model of the application domain, inference systems can also be used to directly implement subtasks of application systems. In so-called information systems, the subtasks of applications can be realized by specifying queries to be answered based on an explicitly given information model. Query answering is the basic operation found in information systems – be they web-based or not. Usually, this scenario is the realm of information retrieval and database systems. In both research areas more and more powerful representation and query languages are investigated. Rather than being a mere information lookup, answering queries in information systems for complex domains involves reasoning about implicit information, i.e. query answering systems require inference systems as subcomponents. Thus, a new generation of information systems, especially in a web-based scenario, can be called deductive information systems.

The focus of classical database systems is still mainly on other important aspects such as persistency, efficiency (storage requirements and speed of retrieval), transactions with rollback etc. However, as more powerful representation languages are required for building applications, inferences play a more and more important role in database systems as well (cf. [Abiteboul et al., 1995; Zaniold et al., 1997]). Not surprisingly, the results of research on formal inference systems and knowledge representation become directly relevant to "conventional computer science" (and vice versa) [Chomicki & Saake, 1998; Kuper et al., 2000]. An information system whose main services are based on formal inference processes will be called a deductive information system. A deductive information systems need not be implemented as a monolithic architecture, it could be comprised of many cooperating (deductive) subsystems.

The development of formal inference systems has a long tradition. The literature contains many contributions from different points of view and with different mathematical background. For many problems, it is appropriate that formal modeling and inference techniques are based on a logical approach, i.e. a logical characterization of various problem classes has already been developed in terms of deduction, induction, abduction, synthesis etc. In this work we pursue a computational approach combining theoretical results about the decidability of different logical representation languages with practical results concerning "efficient" proof procedures which are sound and complete (and terminating). Efficient proof techniques, in turn, are needed for implementing, for instance, query answering and specification checking systems based on expressive languages. The development of efficient algorithms that do not run into combinatorial explosion in the average case is a very active and exciting research field not only from a practical but also from a theoretical point of view.

In the context of deductive information system scenarios, formal representation systems applied in practice must support adequate representational expressivity to model different kinds of phenomena. For instance, in many geographic information systems (GIS), spatial knowledge about artifacts and natural objects should provide the basis for problem solving processes. If application problems are to be automatically solved from problem specifications, a sound semantics for the representation formalism is mandatory. In order to be able to give an abstract characterization about what is to be computed as a problem solution, it is very important to *integrate* conceptual and spatial (as well as temporal) representation structures on a semantical basis. There is no question that even in information retrieval contexts preferably representation languages for which sound and complete inference procedures exist should be considered. Therefore, logic programming approaches with Horn clauses are only briefly discussed (see also the analysis in [Baader, 1999]).

Information systems are not the only area where formal inference systems can be successfully applied. The ever-growing interest in electronic commerce in the context of distributed systems raises the problem of verifying that possibly interacting services can actually be provided in all circumstances. Furthermore, telecommunication systems also tend to supply a vast amount of different features that cannot necessarily be combined without any trouble. Thus, based on models of these systems and a set of constraints (or invariants), formal verification is a prevalent demand to ensure secure operation of these systems. In this context, model checking techniques (see e.g. [Clarke & Kurshan, 1996] for an overview) are successfully used to check whether a property stated for a system (based on a finite state model) is valid. A problem with model checking techniques is that (i) only finite state spaces can be handled and (ii) only control-oriented aspects of application can be verified. Therefore, model checking can be successfully complemented with theorem proving techniques which can handle infinite models and can also be used to verify data models [Katoen, 1999].

A family of logic-based representation languages that is well-suited for many of the above-mentioned tasks is known as *description logics* (DLs) [Baader et al., 2001]. The main representational means are concepts (unary predicates) and roles (binary predicates).[1] Description logics, which have a strong correspondence to modal logics, are interesting from a theoretical as well as from a computational point of view. With these modeling languages, application problems (or subproblems) can be reduced to formally well-defined inference services. Experiences with applications indicate that expressive description logics are required to solve practical modeling problems without resorting to ad hoc extensions. For expressive description logics the development of inference systems based on sound and complete calculi is a relatively new research area. The focus of the work presented in this Habilitation Thesis is on the development of decidable description logics with enough expressive power for solving

[1] In some approaches even n-ary roles have been investigated [Calvanese et al., 1998].

application problems. The work extensively investigates language features that can be used in practical applications. Thus, if a language feature – at the current state of the art – leads to combinatorial explosion even for toy problems, it is not considered for implementation. The significance of the new results presented in this work is shown by discussing problems that could not be solved before.

The formalization of application problems as deduction tasks and the employment of theoretically well-investigated proof systems as problem solvers results in a faster development of more robust applications with increased functionality. As many experiments indicate, in the average case solutions for many problems can indeed be computed in a reasonable amount of time on todays computers even though the worst-case complexity class of the problems might be exponential.

1.1 Applications, Research Problems and Research Methodology

The use of formal representation and inference systems is motivated with different kinds of applications:

- Checking of specifications for telecommunication systems [Areces et al., 1999].
- Development of large ontologies in bio-informatics [Schulz & Hahn, 2000].
- Construction of technical devices.
- Knowledge-based image interpretation.
- Deductive information systems with application perspective in ecommerce.

The first two application scenarios have been investigated by other groups (op. cit.). They serve as interesting testbeds for the practical evaluation of description logic inference algorithms. Besides configuration and image interpretation tasks, another application area considered in this work are deductive and cooperative information systems [Papazoglou & Schlageter, 1998]. In particular, we motivate the use of description logics in the context of spatial (geographical) as well as agent-based information systems. Description logics have been proven to be an important technology for information integration [Catarci & Lenzerini, 1993; Levy et al., 1996; Calvanese et al., 1998; Kashyap & Sheth, 1998; Klusch, 1998]).

The research problems being tackled in this Habilitation Thesis can be characterized as follows. The prevalent goal is to provide a language with adequate means for representation and inference. Although some aspects of nonmonotonic reasoning (default reasoning) are also investigated in this context, the focus of this work is on

conceptual representation languages for representing terminological and assertional knowledge with description logics.

First, sound and complete algorithms for expressive description logics are developed. Based on decidability proofs, sound and complete algorithms that allow for an *optimized* implementation are developed and evaluated using a practical implementation. Different subproblems have to be solved on the way to a practical inference system:

1. Development of an inference system based on a formal calculus for decidability proofs. The problem is to integrate and extend (i) techniques for informed search and (ii) techniques for caching intermediate results.

2. Application of benchmark generators to approach the point where combinatorial explosions occurs (transition phase) in order to identify combinatorial problems. Testing and verifying the implementation with benchmarks for which the results are known. Automatic testing with different optimization techniques dis/enabled.

3. Testing of application problems encoded as logical inference problems and testing of application knowledge bases (either manually composed or automatically generated) such that empirical experiments with "mass data" are possible.

Second, reasoning about predicates over discrete and continuous domains (e.g. about real numbers) is integrated with expressive conceptual reasoning using description logics. In the context of configuration problems a new DL representation language is used for describing the space of possible configurations. An inference algorithm for the representation language is given and its soundness, completeness (and termination) is shown.

Third, known theories for dealing with aspects of spatial reasoning are integrated into the framework for conceptual reasoning with description logics. The integrated treatment of conceptual and spatial reasoning on a semantical basis with a decidable description logic is a hard problem that, due to the best of our knowledge, is first considered in the work reported on in this thesis. For application problems, not only the standard inference services are important. Default reasoning about conceptual and spatial information is another problem for which solutions are provided.

As mentioned above, in order to solve an application problem with a logical approach, it is necessary to analyze how an application problem can be formalized and what kind of logic is required. However, not all application problems can be solved by resorting to one specific inference service offered by a logical inference system. Different representation problems might require different formalisms. For instance, as will be argued in this thesis, this is the case in an application scenario for building

a deductive information system in a distributed context. If query results obtained from inference systems for different formalisms are to be combined, for instance in an information system, a difficult problem is the translation of data and queries (i.e. abstraction and specialization) and the combination of the results. Thus, a computational architecture is developed in which certain subproblems can be solved with formal reasoning techniques.

Dealing with technical aspects of network protocols and "lower-level" communication techniques is an important problem. However, since much progress has been made in this area, the research methodology we pursue in this thesis is to focus on logical representations in order to define what is to be computed in order to organize a multi-agent inference scenario.

With this Habilitation Thesis it is shown that expressive description logics can be used in various ways for modeling and problem solving in application systems. Expressive description logics can complement other logic-based formalisms that might be used for subproblems as well (e.g. Datalog in deductive databases). However, as we will see, the broad spectrum of possible applications it the main motivation for the practical and theoretical investigations of description logics in this Habiliation Thesis.

1.2 Overview

Focusing on the description logic $\mathcal{ALCNH}_{R^+}$, Chapter 2 gives an introduction to description logics with a definition of the syntax and semantics of concept and role constructors.[2] Terminological and assertional axioms are introduced. Furthermore, inference problems are defined. For the description logic $\mathcal{ALCNH}_{R^+}$ the first sound and complete ABox calculus has been published in [Haarslev & Möller, 2000b]. $\mathcal{ALCNH}_{R^+}$ extends the basic logic $\mathcal{ALC}$ [Schmidt-Schauss & Smolka, 1991] with number restrictions, role hierarchies and transitive roles. In Chapter 2, additional concept and role constructors are introduced that are used in subsequent chapters. Thus, Chapter 2 provides the foundation for the other parts of this Habilitation Thesis.

In order to perform empirical tests of practical inference algorithms, the TBox and ABox description logic inference system RACE (Reasoner for ABoxes and Concept Expressions) for the DL $\mathcal{ALCNH}_{R^+}$ has been implemented. Chapter 3 gives an introduction to the design philosophy of description logic systems (see also [Möller & Haarslev, 2001]). With this chapter the motivation behind the design of DL representation and reasoning systems is given.

The DL system RACE is described in Chapter 4. RACE is the first implemented

[2] A convenient pronunciation of $\mathcal{ALCNH}_{R^+}$ is $\mathcal{ALC}$-nature.

DL system for ABox reasoning in the DL $\mathcal{ALCNH}_{R^+}$ which is based on sound and complete algorithms (see also [Haarslev et al., 1999c]). For the implementation of RACE, practical inference procedures known in the literature (see below for a discussion) have been adapted and extended. New algorithms and data structures have been developed and evaluated. Although, meanwhile, for some very expressive description logic constructs and their combination, sound and complete (and terminating) tableaux calculi have been presented in the theory literature, in many cases it is currently unknown how to optimize the average case. Rather than implementing an ad hoc version which is sound and complete but cannot compute solutions for even simple inference problems, the methodology of the RACE system is to support only those constructors for which it can be shown that larger problems can be solved. The algorithms and data structures used in the implementation of RACE are given in Chapter 4. Empirical evaluations of different algorithms are presented as well. The results have been published in [Haarslev & Möller, 1999a; Haarslev & Möller, 1999b; Haarslev & Möller, 2000a; Haarslev & Möller, 2000c; Haarslev & Möller, 2000d]. Furthermore, empirical investigations about the performance of RACE on the encoding of algorithmic problems in modal logics (e.g. for searching a maze) are described in [Haarslev & Möller, 2000a]. Although the focus of this work is on description logics, considering the direct correspondence between description logics and modal logics [Schild, 1991a] reveals that the results are of particular relevance to many other parts of computer science. Some of the results presented in Chapter 4 are published in this Habilitation Thesis for the first time.

Chapter 5 introduces the results about the combination of the DL $\mathcal{ALCNH}_{R^+}$ with constraint reasoning (involving so-called concrete domains). The decidability of the resulting language $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$ is shown (see also [Haarslev et al., 2000a]. Based on the language $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$ logical representations for application problems which can be represented as synthesis tasks (e.g. configuration problems) are developed.

In the sixth chapter, research results on the combination of terminological and spatial reasoning are discussed. A promising combination of spatial, temporal and conceptual reasoning on a semantical basis without resorting to ad hoc techniques has been presented with the DL $\mathcal{ALCRP}(\mathcal{D})$ [Haarslev et al., 1998; Haarslev et al., 1999b]. Today, these theoretical results are particularly important for knowledge representation but, since formal modeling is now more and more important also in the field of object-oriented modeling and database theory, the combination of conceptual and, for instance, spatial reasoning will possibly influence other areas of computer science in the near future. The work presented in Chapter 6 is originally based on earlier approaches concerning a less expressive variant presented in [Haarslev & Möller, 1997a]. Examples are based on work presented in [Haarslev & Möller, 1997b]. Initial results on the language $\mathcal{ALCRP}(\mathcal{D})$ and its calculus have been published in [Lutz,

1998] (see also [Lutz & Möller, 1997]). The reader who is mainly interested in spatial reasoning in combination with terminological reasoning can skip the chapters 3 and 4 and, after reading Chapter 5, may continue with Chapter 6. Chapter 6 also considers spatioterminological default reasoning [Möller & Wessel, 1999] with applications to image understanding [Möller et al., 1999a] and applications in visual query languages [Haarslev et al., 2000b].

Chapter 7 presents an extended example. For different application problems different representation languages are discussed. In order to combine specific representation formalisms for solving compound problems, an agent-based scenario is investigated. A broker agent is used to transform queries in such a way that specialist agents can be used to answer queries. For the languages used in this scenario, implementations are available that can be used in practical applications. Pointers to ongoing work and new results are provided where appropriate. The chapter is based on work published in [Möller et al., 1998; Möller et al., 2001]. Abstraction operators developed in the field of machine learning (least-common subsumer operation) are used to support example-based query answering. Parts of chapter 7 refer to notions defined in Chapter 5 (concrete domains) and Chapter 6 (complex roles). At the end of chapter 7, a combination of description logics and Bayesian networks is used in an information system. New results for defining abstraction operators are presented in this framework (see also [Mantay et al., 1999]). The development of abstraction operations in the context of conceptual and probabilistic inference systems demonstrates that both formalisms can be fruitfully combined.

Chapter 2

The Description Logic $\mathcal{ALCNH}_{R^+}$

In the field of formal inference systems, description logics have been proven to be a sound basis for solving application problems. Detailed introductions to description logics can be found in [Woods & Schmolze, 1992; Donini et al., 1996; Baader & Sattler, 2000]. The relation of description logics to logic programming techniques is described in [Baader, 1999].

In description logics, the main notions for domain modeling are concepts (unary predicates) and roles. In most DLs roles are binary predicates (but see [Calvanese et al., 1998] for a DL with n-ary roles). A set of axioms (TBox) is used for modeling the terminology of an application. Knowledge about specific individuals and their interrelationships is modeled with a set of additional axioms (so-called ABox).

The description logic $\mathcal{ALC}$ [Schmidt-Schauss & Smolka, 1991] is a propositionally complete representation language providing conjunction, universal quantification and full negation. As an introductory example, let us consider family relationships. For instance, a woman is a human whose gender is restricted to be female whereas a man is a human whose gender is restricted to be male. The concepts male and female are disjoint. A parent is a human which has at least one child which must be a human. Furthermore, a mother is a woman and a parent. This kind of terminological knowledge can easily be modeled with concepts and roles in the language $\mathcal{ALC}$.

$$\textbf{woman} \doteq \mathsf{human} \sqcap \forall\, \mathsf{has\_gender} \,.\, \mathsf{female}$$

$$\textbf{man} \doteq \mathsf{human} \sqcap \forall\, \mathsf{has\_gender} \,.\, \mathsf{male}$$

$$\textbf{parent} \doteq \mathsf{human} \sqcap \exists\, \mathsf{has\_child} \,.\, \mathsf{human}$$

$$\textbf{mother} \doteq \mathsf{woman} \sqcap \mathsf{parent}$$

$$\textbf{male} \sqsubseteq \neg\mathsf{female}$$

The first four declarations declare necessary and sufficient conditions for the concepts on the left-hand side. In the last declaration only necessary conditions are provided for male which is declared to be disjoint to female. From this it follows that the concepts man and woman are disjoint as well.

The description logic $\mathcal{ALCNH}_{R^+}$ extends $\mathcal{ALC}$ with number restrictions, role hierarchies, and transitively closed roles. For instance, in the family example, we assume a role has_descendant which is declared to be transitive. A direct descendant can be distinguished using role hierarchies, i.e. a non-transitive role has_child is declared with has_descendant as a "superrole".

$$\textbf{has\_child} \sqsubseteq \mathsf{has\_descendant}$$

Concept terms can be composed to describe complex conceptual notions. A "queen with a small family" can be defined as a mother who has at most two children and whose descendants are either princes or princesses. This type of queen can be described with the following axioms:

$$\begin{aligned}
&\textbf{queen_with_small_family} \sqsubseteq \\
&\quad \mathsf{mother} \sqcap \\
&\quad \exists_{\leq 2}\, \mathsf{has_child} \sqcap \\
&\quad \forall\, \mathsf{has_descendant} \,.\, (\mathsf{prince} \sqcup \mathsf{princess}) \\
&\textbf{prince} \sqsubseteq \mathsf{man} \\
&\textbf{princess} \sqsubseteq \mathsf{woman}
\end{aligned}$$

In this case only necessary conditions for the concepts queen_with_small_family, prince and princess are provided.

Now, let us turn to reasoning about individuals. If there exists an individual i1 which has a child i2 which, in turn has a child i3, then, due to has_descendent being transitive and a "superrole" of has_child, i3 is implicitly declared a descendant of i1.

$$\begin{aligned}
&(\mathsf{i1}, \mathsf{i2}) : \mathsf{has_child} \\
&(\mathsf{i2}, \mathsf{i3}) : \mathsf{has_child} \\
&\quad \mathsf{i1} : \mathsf{queen_with_small_family} \\
&\quad \mathsf{i1} : \forall\, \mathsf{has_descendant} \,.\, \forall\, \mathsf{has_gender} \,.\, \mathsf{female}
\end{aligned}$$

Moreover, since the individual i1 is known to be an instance of the above-mentioned concept queen_with_small_family, the individual i3 can be inferred to be an instance of (prince $\sqcup$ princess). With the assertional axiom $\mathsf{i1} : \forall\, \mathsf{has\_descendant} \,.\, \forall\, \mathsf{has\_gender} \,.\, \mathsf{female}$

it can be proven that i3 is an instance of princess because due to the disjointness of male and female the alternative prince of the disjunction prince $\sqcup$ princess is ruled out.

In the following sections the representation language $\mathcal{ALCNH}_{R^+}$ as well as basic inference problems are formally defined.

2.1 The Concept Language of $\mathcal{ALCNH}_{R^+}$

For presenting the syntax and semantics of the language $\mathcal{ALCNH}_{R^+}$ a few definitions are required.

Definition 1 (Roles, Role Axioms, Role Hierarchy) Let R be a set of *role names*. A role name is also called an atomic role or role for brevity. Furthermore, let $S \subseteq R$ be the set of *simple roles*. If R and S are role names, then $\mathsf{R} \sqsubseteq \mathsf{S}$ is a *role inclusion axiom*. If R is a role name, then $\mathsf{transitive}(\mathsf{R})$ is called a *role transitivity axiom*. Both kinds of axioms are called *role axioms*. A set of role inclusion axioms is also called a *role hierarchy*.

Additionally, we define the set of ancestors and descendants of a role as well as the set of transitive roles w.r.t. a set of role axioms.

Definition 2 (Role Descendants/Ancestors) Let $\mathcal{R}$ be a set of role axioms and $\sqsubseteq_{\mathcal{R}}$ be defined as $\{(\mathsf{R}, \mathsf{S}) \,|\, \mathsf{R} \sqsubseteq \mathsf{S} \in \mathcal{R}\}$ and let ${\sqsubseteq^*}_{\mathcal{R}}$ be the reflexive transitive closure of $\sqsubseteq_{\mathcal{R}}$ over R. Given a set of role axioms $\mathcal{R}$ the set $\mathsf{R}^{\uparrow}{}_{\mathcal{R}} := \{\mathsf{S} \in R \,|\, (\mathsf{R}, \mathsf{S}) \in {\sqsubseteq^*}_{\mathcal{R}}\}$ defines the *ancestors* and $\mathsf{R}^{\downarrow}{}_{\mathcal{R}} := \{\mathsf{S} \in R \,|\, (\mathsf{S}, \mathsf{R}) \in {\sqsubseteq^*}_{\mathcal{R}}\}$ the *descendants* of a role R w.r.t. a set of role axioms $\mathcal{R}$. The set of transitive roles $T_{\mathcal{R}}$ of a set of role axioms $\mathcal{R}$ is defined as $\{\mathsf{R} \,|\, \mathsf{transitive}(\mathsf{R}) \in \mathcal{R}\}$.

Definition 3 (Role Box) A finite set of role axioms is called a *role box* $\mathcal{R}$ if $\forall\, \mathsf{R} \in S : \mathsf{R}^{\downarrow}{}_{\mathcal{R}} \cap T_{\mathcal{R}} = \emptyset$. A role box is also called *RBox* for brevity.

In the following, the index ${}_{\mathcal{R}}$ is omitted if the role box $\mathcal{R}$ is clear from the context.

Using the definitions from above, the syntax of concept terms in $\mathcal{ALCNH}_{R^+}$ is defined as follows.

Definition 4 (Concept Terms) Let C be a set of concept names which is disjoint from R. Any element of C is a *concept term*. If C and D are concept terms, $\mathsf{R} \in R$ is an arbitrary role, $\mathsf{S} \in S$ is a simple role, $n, m \in \mathbb{N}, n > 1$, and $m > 0$, then the following expressions are also concept terms:

- $\mathsf{C} \sqcap \mathsf{D}$ *(conjunction)*
- $\mathsf{C} \sqcup \mathsf{D}$ *(disjunction)*

- $\neg \mathsf{C}$ *(negation)*
- $\forall \mathsf{R} \,.\, \mathsf{C}$ *(value restriction)*
- $\exists \mathsf{R} \,.\, \mathsf{C}$ *(exists restriction)*
- $\exists_{\leq m} \mathsf{S}$ *(at most number restriction)*
- $\exists_{\geq n} \mathsf{S}$ *(at least number restriction).*

A concept term may be put in parentheses. Concept names are also called atomic concepts. For brevity, concept terms are also called concepts. $\top$ ($\bot$) is considered as an abbreviation for $\mathsf{C} \sqcup \neg \mathsf{C}$ ($\mathsf{C} \sqcap \neg \mathsf{C}$) for some $\mathsf{C} \in C$. For an arbitrary role R, the term $\exists_{\geq 1} \mathsf{R}$ can be rewritten as $\exists \mathsf{R} \,.\, \top$, $\exists_{\geq 0} \mathsf{R}$ as $\top$, and $\exists_{\leq 0} \mathsf{R}$ as $\forall \mathsf{R} \,.\, \bot$. Thus, we do not consider these terms as number restrictions in our language.

Definition 5 (Terminological Axiom, TBox) If C and D are concept terms, then $\mathsf{C} \sqsubseteq \mathsf{D}$ is a terminological axiom. A terminological axiom is also called *generalized concept inclusion* or *GCI*. A finite set of terminological axioms $\mathcal{T}$ is called a *terminology* or *TBox*. For a pair of axioms $\mathsf{C} \sqsubseteq \mathsf{D}$, $\mathsf{D} \sqsubseteq \mathsf{C}$ the abbreviation $\mathsf{C} \doteq \mathsf{D}$ is used.

The next definition gives a model-theoretic semantics to the language introduced above.

Definition 6 (Semantics) An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a set $\Delta^{\mathcal{I}}$ (the abstract domain) and an interpretation function $\cdot^{\mathcal{I}}$. The interpretation function $\cdot^{\mathcal{I}}$maps each concept name C to a subset $\mathsf{C}^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, each role name R from R to a subset $\mathsf{R}^{\mathcal{I}}$ of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. Let the symbols C, D be concept expressions and let R, S be role names. Then, the interpretation function is extended to arbitrary concept and role terms as follows ($\| \cdot \|$ denotes the cardinality of a set):

$$
\begin{aligned}
(\mathsf{C} \sqcap \mathsf{D})^{\mathcal{I}} &:= \mathsf{C}^{\mathcal{I}} \cap \mathsf{D}^{\mathcal{I}} \\
(\mathsf{C} \sqcup \mathsf{D})^{\mathcal{I}} &:= \mathsf{C}^{\mathcal{I}} \cup \mathsf{D}^{\mathcal{I}} \\
(\neg \mathsf{C})^{\mathcal{I}} &:= \Delta^{\mathcal{I}} \setminus \mathsf{C}^{\mathcal{I}} \\
(\exists \mathsf{R} \,.\, \mathsf{C})^{\mathcal{I}} &:= \{a \in \Delta^{\mathcal{I}} \mid \exists\, b : (a, b) \in \mathsf{R}^{\mathcal{I}}, b \in \mathsf{C}^{\mathcal{I}}\} \\
(\forall \mathsf{R} \,.\, \mathsf{C})^{\mathcal{I}} &:= \{a \in \Delta^{\mathcal{I}} \mid \forall\, b : (a, b) \in \mathsf{R}^{\mathcal{I}} \Rightarrow b \in \mathsf{C}^{\mathcal{I}}\} \\
(\exists_{\geq n} \mathsf{R})^{\mathcal{I}} &:= \{a \in \Delta^{\mathcal{I}} \mid \|\{b \mid (a, b) \in \mathsf{R}^{\mathcal{I}}\}\| \geq n\} \\
(\exists_{\leq m} \mathsf{R})^{\mathcal{I}} &:= \{a \in \Delta^{\mathcal{I}} \mid \|\{b \mid (a, b) \in \mathsf{R}^{\mathcal{I}}\}\| \leq m\}
\end{aligned}
$$

An interpretation $\mathcal{I}$ is a *model of a concept* C iff $\mathsf{C}^{\mathcal{I}} \neq \emptyset$. An interpretation $\mathcal{I}$ satisfies a terminological axiom $\mathsf{C} \sqsubseteq \mathsf{D}$ iff $\mathsf{C}^{\mathcal{I}} \subseteq \mathsf{D}^{\mathcal{I}}$. $\mathcal{I}$ is a *model of a TBox* $\mathcal{T}$ iff it satisfies

all terminological axioms $\mathsf{C} \sqsubseteq \mathsf{D}$ in $\mathcal{T}$. An interpretation $\mathcal{I}$ is a *model of an RBox* $\mathcal{R}$ iff $\mathsf{R}^{\mathcal{I}} \subseteq \mathsf{S}^{\mathcal{I}}$ for all role inclusions $\mathsf{R} \sqsubseteq \mathsf{S}$ in $\mathcal{R}$ and, in addition, $\forall\, \mathsf{transitive}(\mathsf{R}) \in \mathcal{R} : \mathsf{R}^{\mathcal{I}} = (\mathsf{R}^{\mathcal{I}})^{+}$.

2.2 The Assertional Language of $\mathcal{ALCNH}_{R^+}$

In the following, the language for representing knowledge about specific individuals is introduced.

Definition 7 (Assertional Axioms, ABox) Let $O = O_O \cup O_N$ be a set of individual names (or individuals), where the set O_O of "old" individuals is disjoint with the set O_N of "new" individuals. Old individuals are those names that explicitly appear in an ABox given as input to an algorithm for solving an inference problem (see below), i.e. the initially mentioned individuals must not be in O_N. Elements of O_N will be generated internally. If C is a concept term, $\mathsf{R} \in R$ a role name and $\mathsf{a}, \mathsf{b} \in O_O$ are individual names, then the following expressions are *assertional axioms* or *ABox assertions*:

- $\mathsf{a} : \mathsf{C}$ *(concept assertion)*,
- $(\mathsf{a}, \mathsf{b}) : \mathsf{R}$ *(role assertion)*,

An *ABox* $\mathcal{A}$ is a finite set of assertional axioms.

We need a few additional terms: An individual b is called a *direct successor* of an individual a in an ABox $\mathcal{A}$ iff $\mathcal{A}$ contains the assertional axiom $(\mathsf{a}, \mathsf{b}) : \mathsf{R}$. An individual b is called a *successor* of a if it is either a direct successor of a or there exists in $\mathcal{A}$ a chain of assertions $(\mathsf{a}, \mathsf{b}_1) : \mathsf{R}_1, (\mathsf{b}_1, \mathsf{b}_2) : \mathsf{R}_2, \ldots, (\mathsf{b}_\mathsf{n}, \mathsf{b}) : \mathsf{R}_{\mathsf{n}+1}$. In case that $\mathsf{R}_\mathsf{i} = \mathsf{R}_\mathsf{j}$ or $\mathsf{R}_\mathsf{i} \in \mathsf{R}^{\downarrow}$ for all $\mathsf{i}, \mathsf{j} \in 1..\mathsf{n}+1$ we call b the (direct) R-*successor* of a. A (direct) *predecessor* is defined analogously.

The interpretation function $\cdot^{\mathcal{I}}$ of the interpretation $\mathcal{I}$ can be extended to the assertional language. Every individual name from O is mapped to a single element $\Delta^{\mathcal{I}}$ in such a way that for $\mathsf{a}, \mathsf{b} \in O_O$, $\mathsf{a}^{\mathcal{I}} \neq \mathsf{b}^{\mathcal{I}}$ if $\mathsf{a} \neq \mathsf{b}$ (*unique name assumption*). This ensures that different individuals in O_O are interpreted as different objects. The unique name assumption does not hold for elements of O_N, i.e. for $\mathsf{a}, \mathsf{b} \in O_N$, $\mathsf{a}^{\mathcal{I}} = \mathsf{b}^{\mathcal{I}}$ may hold even if $\mathsf{a} \neq \mathsf{b}$.

An interpretation satisfies an assertional axiom $\mathsf{a} : \mathsf{C}$ iff $\mathsf{a}^{\mathcal{I}} \in \mathsf{C}^{\mathcal{I}}$ and $(\mathsf{a}, \mathsf{b}) : \mathsf{R}$ iff $(\mathsf{a}^{\mathcal{I}}, \mathsf{b}^{\mathcal{I}}) \in \mathsf{R}^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is a *model of an ABox* $\mathcal{A}$ iff it satisfies all assertional axioms in $\mathcal{A}$.

If an interpretation $\mathcal{I}$ is a model of a TBox $\mathcal{T}$, an RBox $\mathcal{R}$ or an ABox $\mathcal{A}$, then $\mathcal{I}$ is also said to satisfy $\mathcal{T}$, $\mathcal{R}$ or $\mathcal{A}$, respectively.

Definition 8 (Knowledge Base) A knowledge base is a triple $(\mathcal{T}, \mathcal{R}, \mathcal{A})$ where $\mathcal{T}$ is a TBox, $\mathcal{R}$ is an RBox and $\mathcal{A}$ is an ABox. An interpretation $\mathcal{I}$ is a *model of a knowledge base* $(\mathcal{T}, \mathcal{R}, \mathcal{A})$ iff $\mathcal{I}$ is a model of $\mathcal{T}$, $\mathcal{R}$ and $\mathcal{A}$.

2.3 Inference Problems

In the following we define several inference problems. The inference problems are not only relevant for $\mathcal{ALCN}\mathcal{H}_{R^+}$ but also for other description logics discussed below.

Definition 9 (Consistency Problem) A *concept* is called *consistent* (w.r.t. a TBox $\mathcal{T}$ and an RBox $\mathcal{R}$) iff there exists a model of C (that is also a model of $\mathcal{T}$ and $\mathcal{R}$). An *ABox* $\mathcal{A}$ is *consistent* (w.r.t. a TBox $\mathcal{T}$ and an RBox $\mathcal{R}$) iff $\mathcal{A}$ has model $\mathcal{I}$ (which is also a model of $\mathcal{T}$ and $\mathcal{R}$). A *knowledge base* is called *consistent* iff there exists a model. A concept, ABox or knowledge base which is not consistent is called inconsistent.

Definition 10 (Subsumption Problem) A concept term D *subsumes* a concept term C (w.r.t. a TBox $\mathcal{T}$ and an RBox $\mathcal{R}$) iff $\mathsf{C}^{\mathcal{I}} \subseteq \mathsf{D}^{\mathcal{I}}$ for all interpretations $\mathcal{I}$ (that are models of $\mathcal{T}$ and $\mathcal{R}$). If D subsumes C, then C is said to be *subsumed by* D.

Besides these basic problems, some additional inference services are provided by description logic systems. A basic reasoning service is to compute the subsumption relationship between atomic concepts (i.e. elements from C). This inference is needed to build a hierarchy of concept names w.r.t. specificity. The problem of computing the most-specific atomic concepts mentioned in $\mathcal{T}$ which subsume a certain concept is known as computing the *parents* of a concept. The *children* are the most-general atomic concepts mentioned in $\mathcal{T}$ which are subsumed by a certain concept. We use the name *concept ancestors* (*concept descendants*) for the transitive closure of the parents (children) relation. The computation of the parents and children of every concept name is also called *classification* of the TBox. Another important inference service for practical knowledge representation is to check whether a certain atomic concept is inconsistent. Usually, incoherent atomic concept are the consequence of modeling errors. Checking the consistency of all atomic concepts mentioned in a TBox without computing the parents and children is called a TBox *coherence check*.

If the description logic supports full negation, consistency and subsumption can be mutually reduced to each other since D subsumes C (w.r.t. a TBox $\mathcal{T}$ and an RBox $\mathcal{R}$) iff $\mathsf{C} \sqcap \neg\mathsf{D}$ is inconsistent (w.r.t. $\mathcal{T}$ and $\mathcal{R}$) and C is inconsistent (w.r.t. $\mathcal{T}$ and $\mathcal{R}$) iff C is subsumed by $\bot$ (w.r.t. $\mathcal{T}$ and $\mathcal{R}$). Consistency of concept terms can be reduced to ABox consistency as follows: A concept term C is consistent (w.r.t. a TBox $\mathcal{T}$ and an RBox $\mathcal{R}$) iff the ABox $\{\mathsf{a}:\mathsf{C}\}$ is consistent (w.r.t. $\mathcal{T}$ and $\mathcal{R}$).

Definition 11 (Instance Problem) An individual i is an *instance* of a concept C (w.r.t. a TBox $\mathcal{T}$ and RBox $\mathcal{R}$ and an ABox $\mathcal{A}$) iff $\mathsf{i}^{\mathcal{I}} \in \mathsf{C}^{\mathcal{I}}$ for all models $\mathcal{I}$ (of $\mathcal{T}$, $\mathcal{R}$ and $\mathcal{A}$).

Again, for description logics which support full negation for concept terms, the instance problem can be reduced to the problem of deciding if the ABox $\mathcal{A} \cup \{\mathsf{a}:\neg\mathsf{C}\}$ is inconsistent (w.r.t. $\mathcal{T}$ and $\mathcal{R}$). The test is also called *instance checking*.

Definition 12 (Direct Types) The most-specific atomic concepts mentioned in a TBox $\mathcal{T}$ of which an individual is an instance are called the *direct types* of the individual w.r.t. a knowledge base $(\mathcal{T}, \mathcal{R}, \mathcal{A})$.

The direct types inference problems can be reduced to subsequent instance problems (see [Nebel, 1990a] for details).

Definition 13 (Instance Retrieval) The *retrieval* inference problem is to find all individuals mentioned in an ABox that are an instance of a certain concept term C.

The retrieval inference problem can also be reduced to subsequent instance problems.

Definition 14 (Filler Retrieval) The set of *fillers* of a role R w.r.t. an individual i in a knowledge base $(\mathcal{T}, \mathcal{R}, \mathcal{A})$ is defined as $\{\mathsf{x} \mid (\mathcal{T}, \mathcal{R}, \mathcal{A}) \models (\mathsf{i}, \mathsf{x}):\mathsf{R}\}$ where $(\mathcal{T}, \mathcal{R}, \mathcal{A}) \models ax$ means that all models of $\mathcal{T}$ and $\mathcal{A}$ w.r.t. $\mathcal{R}$ are also models of ax.

Given the ABox consistency problem is decidable, the filler retrieval inference problem can be implemented as follows. Let Test be an atomic concept not mentioned in $\mathcal{A}$ and the TBox $\mathcal{T}$. It is easy to verify that determining the filler of a role w.r.t. an individual can be reduced to checking for all individuals x mentioned in an ABox $\mathcal{A}$ whether the ABox $\mathcal{A}_{\mathcal{T}} \cup \{\mathsf{i}:\forall\,\mathsf{R}\,.\,\mathsf{Test}, \mathsf{x}:\neg\mathsf{Test}\}$ is inconsistent w.r.t. $\mathcal{T}$ and $\mathcal{R}$.

Recently, additional inference problems (so-called non-standard inference problems) have been discussed in the literature. For the sake of completeness we just mention some of them: matching [Baader & Küsters, 2000], rewriting concept terms using terminologies [Baader et al., 2000], least common subsumer (LCS) [Cohen et al., 1992; Baader & Molitor, 2000], computation of maximal consistent (minimal inconsistent) ABox subsets [Baader & Hollunder, 1995a]. For a detailed discussion of applications for these inference problems see the cited literature.

In [Haarslev & Möller, 2000b] it is shown that the ABox consistency problem for the language $\mathcal{ALCNH}_{R^+}$ is decidable. Decidability is shown by giving a tableaux calculus that is sound and complete and terminates (for the proofs see [Haarslev & Möller, 2000b]).

In Chapter 4 the description logic inference system RACE is described. RACE is based on the results of [Haarslev & Möller, 2000b] and implements a DL reasoner offering inference services for all standard inference problems described above.

2.4 Using DLs in Applications: An Example

The inference services offered by a DL representation and inference system can be used to build application domain models and to also implement specific (sub)problems of applications. Examples are given in this section. Note that, for conciseness, the examples are somewhat small-grained, but they should convey the main ideas. We begin by specifying some terminological axioms for modeling concepts in a merchant navy domain.

$$\textbf{captain} \sqsubseteq \mathsf{person}$$

$$\textbf{ship} \sqsubseteq \forall\, \mathsf{has\_home\_port} \,.\, \mathsf{port} \sqcap \forall\, \mathsf{has\_captain} \,.\, \mathsf{captain}$$

$$\textbf{cargo\_ship} \doteq \mathsf{ship} \sqcap \exists_{\geq 1}\, \mathsf{has\_captain} \sqcap \exists_{\leq 1}\, \mathsf{has\_captain} \sqcap \exists_{\geq 1}\, \mathsf{has\_cargo\_storage}$$

$$\textbf{shipyard} \sqsubseteq \forall\, \mathsf{has\_ship\_in\_repair\_dock} \,.\, \mathsf{ship\_in\_shipyard}$$

The first terminological axiom declares a subsumption relationship between atomic concepts. All captains are persons. The axiom for ship specifies that each filler of the attribute has_home_port must be a port, i.e. in DL terminology, each filler must be an instance of the concept port. Furthermore, each filler of the attribute has_captain must be a captain.

Note that it is not always necessary to introduce a name for concepts used in a application. For instance, it might be useful to retrieve all ship individuals which have at least one captain as a role filler for the role has_captain. This can be implemented by invoking the retrieval inference service on the concept $\mathsf{ship} \sqcap \exists_{\geq 1}\, \mathsf{has\_captain}$. For instance, in a deductive information system, a certain concept term might be automatically computed, for instance, during web crawling activities etc.

For real-world problems, large TBoxes are usually required. If in the ontology engineering phase of a large project a new concept name is introduced, it might not be obvious to which concept names already in the terminology it is actually related. Let us add an an example axiom for ship_with_captain to our example TBox.

$$\textbf{ship\_with\_captain} \doteq \mathsf{ship} \sqcap \exists_{\geq 1}\, \mathsf{has\_captain} \sqcap \exists_{\leq 1}\, \mathsf{has\_captain}$$

Though not explicitly stated it is evident that, due to the semantics given above, ship_with_captain subsumes cargo_ship. That is to say, cargo_ship is a subconcept

of ship_with_captain. Invoking the inference service for computing the parents and children of the "new" concept name ship_with_captain w.r.t. the example TBox will lead to the answer {ship} for the parents and {cargo_ship} for the children. A DL reasoning and inference system automatically issues a warning if, probably by a modeling mistake, a concept name is inconsistent. Furthermore, if concept names are found to subsume each other (i.e. they are equivalent) a warning is generated as well.

Not all subsumption relationships are as obvious as the subsumption between the concepts ship_with_captain and cargo_ship, as the following example axioms indicate. First, a few role axioms are defined.

$$\textbf{has\_container\_storage} \sqsubseteq \text{has\_cargo\_storage}$$
$$\textbf{has\_cooling\_storage} \sqsubseteq \text{has\_container\_storage}$$
$$\textbf{has\_gas\_storage} \sqsubseteq \text{has\_container\_storage}$$

The role has_container_storage is a subrole of has_cargo_storage. Afterwards, two subroles for has_container_storage are defined: has_cooling_storage and has_gas_storage. Then, the following concept axioms are added to the TBox.

$$\textbf{container\_ship} \sqsubseteq \text{cargo\_ship}$$
$$\exists\, \text{has\_container\_storage} \,.\, \top \sqsubseteq \text{container\_ship}$$
$$\top \sqsubseteq \forall\, \text{has\_container\_storage} \,.\, \text{container}$$
$$\top \sqsubseteq \forall\, \text{has\_cooling\_storage} \,.\, \text{cooled\_container}$$
$$\top \sqsubseteq \forall\, \text{has\_gas\_storage} \,.\, \text{gas\_container}$$
$$\textbf{type\_1\_ship} \sqsubseteq$$
$$\exists_{\leq 1000}\, \text{has\_container\_storage} \sqcap$$
$$\exists_{\geq 600}\, \text{has\_cooling\_storage} \sqcap$$
$$\exists_{\geq 600}\, \text{has\_gas\_storage} \sqcap$$

A container_ship is a ship. Moreover, a so-called domain restriction for the role has_container_storage is declared with the second axiom. If there are any individuals set into relation to an individual i via the role has_container_storage (i.e. i is on the left-hand side), then the individual i must be a container_ship. Afterwards, three range restrictions are defined. For any individual it holds that the fillers of the role has_container_storage must be instances of container, the fillers of has_cooling_storage must be instances of the concept cooled_container, and the fillers of has_gas_storage must be instances of gas_container. The last axiom specifies a

type_1_ship as an object with up to 1000 fillers for has_container_storage, at least 600 fillers for has_cooling_storage and at least 600 fillers for has_gas_storage.

Now let us assume that due to safety problems, container ships with storage capacity for containers which can be cooled and which may contain gas are classified as a dangerous ships. For instance, the following axiom is added to the TBox in order to define a concept dangerous_ship with necessary and sufficient conditions.

$$\begin{aligned}&\textbf{dangerous\_ship} \doteq \\ &\quad (\exists\, \mathsf{has\_container\_storage}\,.\,(\mathsf{cooled\_container} \sqcap \mathsf{gas\_container})) \sqcup \\ &\quad (\exists\, \mathsf{has\_cargo\_storage}\,.\,\mathsf{toxic\_waste})\end{aligned}$$

After TBox classification, non-obvious subsumption relationships become apparent. Due to the domain restriction for has_container_storage, the concept dangerous_ship is inferred to be an instance of container_ship. Furthermore, and more interesting, it can be deduced that the concept type_1_ship is subsumed by dangerous_ship. Since there can be at most 1000 fillers for has_container_storage, there must exist at least 200 individuals which are fillers of both roles has_cooling_storage and gas_container. Hence, there must be at least 200 fillers of has_container_storage which are instances of the concept cooled_container ⊓ gas_container. Classifying the TBox and asking for the concept descendants of dangerous_ship yields type_1_ship, possibly among others.

A container itself could include dangerous things. For instance, a certain class of containers with toxic waste (but with an innocent name) is defined.

$$\begin{aligned}&\textbf{type\_47\_container} \sqsubseteq \mathsf{container} \sqcap \exists\, \mathsf{has\_cargo\_storage}\,.\,\mathsf{toxic\_waste} \\ &\textbf{type\_2\_ship} \sqsubseteq \exists\, \mathsf{has\_container\_storage}\,.\,\mathsf{type\_47\_container}\end{aligned}$$

Unfortunately, with the axioms given above type_2_ship would not be classified as a dangerous_ship. This would be the case, however, if has_cargo_storage were declared as a transitive role. As indicate above, this facility is also supported by $\mathcal{ALCN}\mathcal{H}_{R^+}$.

Now let us turn to an ABox example and introduce some assertional axioms for individuals. In some circumstances an individual should be an instance of a concept when the instance is set into relation to another instance. For example, a ship individual should be an instance of the concept ship_in_shipyard when it is set into relation to a shipyard. A similar situation occurs when persons become customers if they are set into relation to a bank. This dynamic classification is no problem when ABoxes are used for object representation. The ship example is continued with the following assertions.

$$\mathsf{s1} : \mathsf{ship},\ \ \mathsf{yard1} : \mathsf{shipyard},\ \ (\mathsf{yard1}, \mathsf{s1}) : \mathsf{has\_ship\_in\_repair\_dock}$$

Even though s1 is "created" as a ship, asking for the direct types of s1 reveals that s1 is also an instance of ship_in_shipyard because s1 is set into relation to yard1 via has_ship_in_repair_dock and, due to the axiom for shipyard in the TBox, for each shipyard *all* fillers of has_ship_in_repair_dock are instances of ship_in_shipyard. This instance classification is "dynamic" because s1 will no longer be a ship_in_shipyard when the related statement is retracted. In applications, this kind of "dynamic typing" can be used as a "trigger" to invoke certain actions. For instance, in a listing of a set of ships a ship_in_shipyard might be marked accordingly etc.

Note that in some sense an ABox is more expressive than a relational database. With assertional axioms individuals can be declared as being instances of arbitrary concepts of the DL. In particular, it is possible do declare that an individual is an instance of, for example, motorship $\sqcup$ sailingship. In standard databases this is usually not supported with the same generality.

Another difference between description logic and databases is that in DLs open-world reasoning is the standard inference mode, i.e. even if $(\mathsf{s1}, \mathsf{c1}) : \mathsf{has\_captain}$ was added to our example ABox, asking whether s1 is an instance of ship_with_captain would yield the answer 'no'. Number restrictions can be used to "simulate" some kind of closed world reasoning. For example, after asserting the ABox statements and $\mathsf{s1} : \exists_{\leq 1}\, \mathsf{has\_captain}$, the ship s1 is automatically classified as a ship_with_captain because the sufficient conditions for ship_with_captain are fulfilled (see also the axiom for ship presented above).

Advanced reasoning examples for description logics in general, and for $\mathcal{ALCNH}_{R^+}$ in particular, are presented in subsequent chapters.

2.5 Related Work on DL Theory

In the literature different kinds of additional concept and role constructors are discussed and it is hard to characterize different languages using a common naming scheme. As can be seen by the example of $\mathcal{ALCNH}_{R^+}$, different letters are used to indicate the operators supported by a specific description logic: $\mathcal{N}$ is used for (simple) number restrictions, an $\mathcal{H}$ indicates that role hierarchies are supported. An $_{R+}$ stands for transitive roles.

$\mathcal{ALCNH}_{R^+}$ is an extension of $\mathcal{ALCNH}$ that itself can be polynomially reduced to $\mathcal{ALCNR}$ [Buchheit et al., 1993a] and vice versa. The letter $\mathcal{R}$ indicates role conjunctions. It is possible to rephrase every hierarchy of role names with a set of role conjunctions and vice versa [Buchheit et al., 1993a]. The language $\mathcal{ALCNH}_{R^+}$ extends $\mathcal{ALCNR}$ by additionally providing transitively closed roles. $\mathcal{ALCNH}_{R^+}$ also extends other related description logics such as $\mathcal{ALC}_{R^+}$ [Sattler, 1996] and $\mathcal{ALCH}f_{R^+}$ [Horrocks, 1998]. In contrast to $\mathcal{ALCNH}_{R^+}$ the logic $\mathcal{ALCH}f_{R^+}$ supports only so-

called features (hence the letter f is used). In principle, if a role f is declared to a be feature, an implicit GCI of the form $\top \sqsubseteq \exists_{\leq 1} \mathsf{f}$ is assumed. Thus, features are functional roles.

If also the (in)equality of features fillers can be enforced, the letter $\mathcal{F}$ is used. Some description logics do not provide full negation. In this case, the letter $\mathcal{E}$ indicates that exists restrictions of the form $\exists \mathsf{R} \,.\, \mathsf{C}$ are provided (e.g. the language $\mathcal{ALE}$ provides existential restrictions but no disjunction and negation only for concept names). In subsequent chapters the letter h will be used to indicate role hierarchies with single inheritance only. Furthermore, the suffix $_{trans}$ indicates the availability of an operator for the transitive closure of roles [Baader, 1991].

A sound and complete tableaux calculus for concept reasoning with $\mathcal{ALC}$ and qualified number restrictions is presented in [Hollunder & Baader, 1991]. Qualified number restrictions are indicated with the letter $\mathcal{Q}$. Compared to simple number restrictions, in qualified number restrictions restrict the number of fillers which are instances of a given concept (rather than all fillers). The work on these logics has been extended and a tableaux calculus for deciding *concept consistency* for the language $\mathcal{SHIQ}$ has been presented in [Horrocks et al., 1999b]. In addition to $\mathcal{ALCH}f_{R^+}$, $\mathcal{SHIQ}$ also supports qualified number restrictions and inverse roles (indicated with the letter $\mathcal{I}$). According to the naming scheme used in this Habilitation Thesis the name of $\mathcal{SHIQ}$ would be $\mathcal{ALCQHI}_{R^+}$.[1] First tests with a prototype implementation presented in [Horrocks, 1999] reveal that optimizations for expressive languages with the addition of inverse roles still have to be devised.

Another approach is presented in [De Giacomo & Lenzerini, 1996] where the logic $\mathcal{CIQ}$ for reasoning with TBoxes and ABoxes is introduced. The reasoning procedures developed for $\mathcal{CIQ}$ are based on a polynomial encoding of $\mathcal{CIQ}$ TBoxes into sublanguages of $\mathcal{CIQ}$ [De Giacomo & Lenzerini, 1996]. A similar approach is taken for ABoxes of the languages $\mathcal{CI}$ and $\mathcal{CQ}$. In comparison to $\mathcal{ALCNH}_{R^+}$ and the other approaches mentioned above, $\mathcal{CIQ}$ offers more operators (e.g. inverse roles and transitive closure of roles) but supports role conjunction instead of role hierarchies and allows number restrictions only for primitive roles. No implementation of $\mathcal{CIQ}$ has been developed.

Some description logics provide support for reasoning with individuals in concept terms (see e.g. [Schaerf, 1994]). For describing a concept extensionally as a set of individuals the one-of construct $\{\mathsf{i}_1, \ldots, \mathsf{i}_n\}$ is used (indicated with $\mathcal{O}$). For logics without full existential restrictions of the form $\exists \mathsf{R} \,.\, \mathsf{C}$ sometimes a so-called fills construct $\mathsf{R} : \mathsf{i}$ is introduced (indicated with the letter $\mathcal{B}$). The construct indicates that a certain individual i is a filler of a role R for all instances of a concept. However, this operator is syntactic convenience if existential restrictions are provided since it

[1] The pronunciation of $\mathcal{ALCQHI}_{R^+}$ is assumed to be $\mathcal{ALC}$-choir.

could be written as $\exists \mathsf{R} . \{\mathsf{i}\}$ (see also [Schaerf, 1994, p. 25]). In description logics supporting the one-of operator, for instance, the concept consistency problem can only be defined w.r.t. a knowledge base (i.e. an ABox must always be considered for TBox classification). In practice, this means that it is not possible to classify a TBox in advance and use the results for many ABoxes that refer to the TBox. It is still an open problem whether the concept constructor for sets of individuals $\mathcal{O}$ can be integrated into expressive description logics with inverse roles such as $\mathcal{ALCQHI}_{R^+}$. For detailed overviews on additional description logic operators see for instance [Baader & Sattler, 2000].

The tableaux calculus for deciding the ABox consistency problem [Haarslev & Möller, 2000b] also provides the basis for the $\mathcal{ALCNH}_{R^+}$ description logic system RACE that implements sound and complete (and terminating) inference algorithms for all standard inference problems mentioned above. The RACE system and the optimization techniques used in the implementation are introduced in detail in Chapter 4. Recently, an extension of $\mathcal{SHIQ}$ for dealing with ABoxes has been presented in [Horrocks et al., 2000]. At the time of this writing a DL inference system for this logic supporting reasoning about ABox inference problems comparable to RACE is not available.

As we will see in the next two chapters, a direct implementation of a tableaux calculus with chronological backtracking is hardly appropriate for practical purposes. Dedicated search strategies are necessary in order to avoid combinatorial explosion to occur even for simple inference problems. Before well-established as well as new optimization techniques are introduced for an implementation of $\mathcal{ALCNH}_{R^+}$, we will give an overview of related work on the development of system architecture for representing and reasoning with terminological and assertional knowledge.

Chapter 3

Related Work on DL Systems

In this chapter a description of the goals behind the development of different DL systems is given from a historical perspective. The description of DL systems allows important insights into the development of the knowledge representation research field as a whole. The design decisions behind the well-known systems which we discuss in this chapter do not only reflect the trends in different knowledge representation research areas but also characterize the point of view on knowledge representation that different researchers advocate. The chapter discusses general capabilities of the systems and gives an analysis of the main language features and design decisions behind system architectures. The analysis of current systems in the light of a historical perspective might lead to new ideas for the development of even more powerful description logic systems of the future. References to previous descriptions of DL systems (e.g., [MacGregor, 1991a; Woods & Schmolze, 1992; Horrocks, 1997]) or publications on DL theory that also contain discussions about description logic systems (e.g. [Patel-Schneider, 1987a; Nebel, 1990a; Schmidt, 1991]) are included where appropriate. For references to other systems not mentioned here see also [Woods & Schmolze, 1992] and [Nebel, 1990b, p. 46f., p. 63f.].

3.1 The First Generation

Inspired by research on human cognitive behavior, proposals for knowledge representation languages were first discussed in the late sixties. E.g. [Quillian, 1967] is one of the first publications of these languages called "semantic networks" (see also [Quillian, 1968]). Originally, *semantic network formalisms* were seen as alternatives to first-order logic. In a similar spirit, [Minsky, 1975] has introduced the initial notion of a *frame system*. The motivation of these representation formalisms was to mimic human reasoning in the sense of achieving "cognitive adequacy". Thus, the idea was to support problem solving with appropriate representation structures

that somehow "resemble" representation structures assumed in human information processing. The exploitation of inheritance was a predominant idea in frame systems. The specification of knowledge bases should be simple and the use of the representation structures should be intuitive ("epistemological adequacy"). However, as pointed out by [Woods, 1975], it was not at all simple to specify what an inference system was supposed to actually compute. The late seventies saw initial research on the relation of frame systems and first-order logic [Hayes, 1977; Hayes, 1979] which revealed that some aspects of frame-based systems can be considered as special "instantiations" of first-order reasoning. Hayes argued that frame-based reasoning was not an entirely new way of knowledge representation that had particular advantages over first-order reasoning. Specific features of frame systems beyond first-order reasoning, e.g. defaults, were not very well understood at that time. The consequence of these publications was that many researchers did not consider frame systems and semantic network systems as possible alternatives to logic-based approaches any more.

The criticisms of early frame and semantic network representation formalisms stimulated research on the development of mathematical structures and techniques for defining the semantics of the constructs supported by different representation languages. For instance, in early frame systems there was no clear distinction between constructs for representing "generic" knowledge about sets of individuals and knowledge about "specific" individuals. Furthermore, frames were often used as data structures in procedural programs. For these programs a formal specification of what they were expected to compute was rarely provided. Rather than interpreting frame structures as data structures, [Woods, 1975] suggested to use a formal semantics to clearly specify what is to be computed by inference algorithms.

KL-ONE

Inspired by critics such as [Woods, 1975], Brachman started to develop a new representation system (called KL-ONE) that inherently included the notion of inferring implicit knowledge from given declarations [Brachman, 1977; Brachman, 1979]. Although the initial approach was not logic-based, KL-ONE started the era of representation systems which can be used to formalize application problems as inference problems over the constructs supported by the representation language. One of the prevailing inference patterns is centered around inheritance [Brachman, 1983]. The final report on the KL-ONE language is published in [Brachman & Schmolze, 1985].

One of the core ideas behind KL-ONE as a representation language for the "epistemological level" resulted from problems with languages offering built-in primitives for general representation purposes (e.g. the CD theory [Schank, 1975]). Rather than providing general built-in primitives, in KL-ONE, for a specific representation prob-

lem a set of adequate primitives is defined by the user. The primitives are denoted by so-called *concept names*. The next idea was to use *concept-forming operators* to build new concepts from basic concepts. These compound concepts are also referred to as "concepts", "concept terms" or "concept descriptions". *Generic concepts* were intended to denote classes of individuals and *individual concepts* were intended to denote individuals (see also [Nebel, 1990a, p. 42]). Individuals are related by so-called *roles* which, in turn, can be primitive roles (role names) or roles described with role constructors [Brachman & Schmolze, 1985].

Concepts and roles are the building blocks for representational purposes. The main idea behind concepts and concept constructors in KL-ONE is that the *meaning of a concept* is derived only from the meaning of its superconcepts and other restrictions associated with a concept [Brachman & Schmolze, 1985]. A KL-ONE generic concept consists of a set of superconcept names, a set of role descriptions, and a set of structural descriptions [Patel-Schneider, 1987a, p. 58f.].[1] Roles can be viewed as potential relationships between individuals of the class denoted by the concept and other individuals in the world [Nebel, 1990a, p. 42]. Role descriptions are restrictions and differentiations. The former restricts the class of permitted fillers (value restrictions) or the number of fillers (number restrictions). Differentiations are used to describe a subrole with possible value or number restrictions. Structural descriptions were used to state relationships between the fillers of roles (see also [Patel-Schneider, 1987a, p. 58f.]). Individual concepts consist simply of a set of values for roles plus a set of generic concepts which they are an instance of. Thus to be a subconcept of a generic concept, an individual concept has to satisfy all the restrictions inherited by the generic concept. However, the semantics of individuals was never completely worked out (see [Schmolze & Brachman, 1982, p. 23–31] cited after [Nebel, 1990a, p. 64]).

The representation structures offered by KL-ONE are similar to semantic networks or frames. Although, initially, the structures offered by KL-ONE are called "structural inheritance networks" [Brachman, 1977; Brachman, 1979], in [Brachman & Levesque, 1984] the authors talk again of frame structures.[2] In accordance with [Nebel, 1990a, p. 45] we argue that in contrast to e.g. the CD theory [Schank, 1975], providing a (large) set of primitive representation structures (names) for all kinds of representation purposes was not the development goal of KL-ONE. As Nebel points out [Nebel, 1990a, p. 45], more important and unique for KL-ONE is the core idea

[1] Note that, in KL-ONE-like languages, there are specific syntactic constructs for specifying superconcepts. These specific constructs are no longer present in logic-based concept languages of the nineties.

[2] There are big differences between frame systems and description logic systems: if for i the restriction $\forall R.C$ holds, and we set i into relation to j via the role r, then every KL-ONE-based system concludes that j is an instance of C. In standard frame-based systems, j can only be set into relation to i via R if it is already known that j is an instance of C. Otherwise, in frame systems at least a warning is issued or even an error is signalled.

of proving ways to specify *concept definitions*, i.e. the possibility to let a knowledge engineer declare the relation of "high-level concepts" to "lower-level primitives".

A concept definition is an assignment of a (unique) name to a concept term. In KL-ONE the well known distinction between the two kinds of concept definitions, definitions with necessary and sufficient conditions and definitions with only necessary conditions (so-called primitive definitions), has been investigated for knowledge representation purposes for the first time.[3] In the original approach no cycles are allowed in the set of concept definitions.[4]

The most important consequence of the introduction of concept definitions with necessary and sufficient conditions was that reasoning about the relationships between concepts became important. The superconcept-subconcept relationships are dependent on the concept terms used in the definitions. Especially the notion of defined concepts (with necessary and sufficient conditions) led to the idea of *concept subsumption*. Rather than with frame systems where the superconcepts are always given explicitly, in KL-ONE the set of direct superconcepts (i.e. concept names) can be computed automatically given a set of concept definitions. Thus, the idea was that the inheritance hierarchy (or subsumption hierarchy) can be computed automatically by inferring implicit information. Inferences in KL-ONE are based on the open-world assumption. Considering the inheritance structure and the lattice of direct superconcepts, concepts can be "inserted" between other concepts automatically. In KL-ONE there is still the notion of a "told" subsumer mentioned explicitly in a list of so-called superconcepts but, according to the semantics, there are also *computed most-specific subsumers*, i.e. those concept names appearing in the TBox which are the most-specific concept names subsuming a certain concept term. The development of an algorithm for computing the direct subsumers (the "classifier") is described in [Schmolze & Lipkis, 1983]. Another inference component called "realizer" computes the most specific atomic concepts of which an individual is an instance [Mark, 1981]. Initial KL-ONE systems were implemented in INTERLISP [Lipkis, 1981] and Smalltalk [Fikes, 1981]. The inference services offered by the classifier and realizer are first exploited in the Consul system [Kaczmarek et al., 1986].

First investigation about defaults, exceptions and concept definitions have been published in [Brachman, 1984]. Nowadays, the semantical theory of defaults and description logic is much clearer, see [Baader & Hollunder, 1992; Baader & Hollunder,

[3] In the literature, some authors use the word "definition" as a synonym for concept terms themselves (e.g. [Schmidt, 1991], see also [Woods, 1991, p. 65]). In this case, "primitive" concepts with only necessary conditions are introduced with a specific marker to be used in concept terms.

[4] The semantics of cycles has been analyzed in [Baader, 1990; Baader, 1991; Nebel, 1990a; Nebel, 1991]. The so-called *descriptive semantics* provides many advantages compared to so-called *fixed point semantics*. For details see [Nebel, 1990a]. One of the first publications of an expressive description logic supporting cyclic axioms with a descriptive semantics and a sound and complete calculus is [Buchheit et al., 1993a].

1993; Baader & Schlechta, 1993; Padgham & Zhang, 1993; Padgham & Nebel, 1993; Baader & Hollunder, 1995a; Baader & Hollunder, 1995b; Donini et al., 1997b].

At the first KL-ONE workshop [Schmolze & Brachman, 1982] it became clear that the informal specification of the semantics of KL-ONE concept and role constructors led to serious problems. The development of the classifier was based on the intuitive meaning of the KL-ONE formalism [Nebel, 1990a, p. 46]. Attempts to logically reconstruct the representation constructs, e.g. [Schmolze & Israel, 1983; Israel & Brachman, 1984] resulted in a deeper understanding of the formalism. Given the formal semantics, implemented algorithms for classification and realization have been shown to be incomplete. Later, investigations showed that KL-ONE (with the formal semantics given in the logical reconstruction approaches) is undecidable (e.g. this holds for the combination of conjunction, value restrictions and role-value maps [Schmidt-Schauss, 1989]). In [Brachman & Levesque, 1984] first thoughts about the tractability of subsumption for sublanguages are discussed. Terminological reasoning with concept definitions even for sublanguages with low expressiveness has been shown to be inherently intractable [Nebel, 1990b, p. 28 and p. 71f.]. Proposals for a semantics based on many-valued logics (e.g. [Patel-Schneider, 1986; Patel-Schneider, 1987a; Patel-Schneider, 1987b; Patel-Schneider, 1989a]) ensure tractable algorithms concerning concept consistency reasoning but also result in a weak expressiveness: many intuitive inferences are not sanctioned by this semantics (see also [Nebel, 1990a]).

Another result of [Schmolze & Brachman, 1982] was that the semantics of individual concepts was not quite clear (e.g. concerning coreference and unique name assumption, see above). Thus, at the first KL-ONE workshop [Schmolze & Brachman, 1982], the notions of a *hybrid* reasoning system consisting of a TBox (a set of concept definitions) and an ABox (a set of assertions concerning individuals) were made more precise. The change of the view on KL-ONE spelled out in [Schmolze & Brachman, 1982, pp. 8–17] (see also [Nebel, 1990a, p. 46]) can be summarized as follows: Not the names of the representation structures are important but the functionality, i.e. the inference services the system provides. The inferences have to be formally defined based on the semantics of the representation formalism. This view led to the development of the functional view of knowledge representation as pursued with the development of the system KRYPTON.

KRYPTON

The knowledge representation system KRYPTON [Brachman et al., 1983a; Brachman et al., 1983b; Brachman et al., 1985] can be seen as the first approach to define a new language of the KL-ONE family with a formal, Tarskian semantics. The distinction between TBox and ABox was first mentioned by the KRYPTON approach (see also

[MacGregor, 1991a, p. 391]). Similar to KL-ONE the distinction between primitive and defined concepts and the computation of the most-specific atomic concepts which instantiate individuals is one of the core ideas of KRYPTON. Furthermore, the goal was to overcome the problems with individuals in KL-ONE [Nebel, 1990a, p. 63].

KRYPTON offers a concept language with low expressiveness. While the initial approach [Brachman et al., 1983b] was too expressive to be tractable (see also [MacGregor, 1991a, p. 390]), in a revised version [Brachman et al., 1985] the concept constructors of KRYPTON are defined as conjunction, value restrictions and role chains. Thus, subsumption checking is polynomial [Patel-Schneider, 1987a, p. 75]. For the ABox a full-fledged resolution-based FOPL theorem prover [Stickel, 1982] is used, i.e. the ABox reasoner of KRYPTON is incomplete. Another perspective is that KRYPTON starts with a first-order logic theorem prover and augments it with a special-purpose inference system for terminological reasoning to cut out some of the combinatorial search [Vilain, 1985]. KRYPTON can be regarded as one of the first efforts in combining knowledge representation and theorem proving techniques but was not used for applications [Nebel, 1990a, p. 63f.].

Rather than dealing with specific representation structures and operations on them, KRYPTON offers a so-called "functional approach." Using interface functions tell and ask, the system can define the knowledge base and ask queries about it, respectively. In this sense, a "functional approach" means that KRYPTON does not necessarily maintain, for instance, the inheritance hierarchy or even an ABox as a graph structure. If, for internal implementation purposes of the inference engine, graph structures are used, they are nevertheless hidden from the user in order to avoid "procedural" operations to be carried out with knowledge representation structures. Arbitrary procedural operations are usually not related to the semantics of the representation formalism such that it is hard to characterize what is actually represented and computed as solutions to inference problems. Thus, the focus of KRYPTON is on the structures to be maintained by the system but is centered around the question about what should the system do for the user, i.e. what services should be made available. In other publications this idea was described as the "knowledge level" [Newell, 1982]: inference services for concept terms are checks for consistency, disjointness and subsumption, for the TBox the most-specific subsumers and for the ABox consistency, instance checking, realization and instance retrieval are offered as inference services. The user should know, at some level not dependent on implementation details, what questions the system is capable of answering and what operations are permitted that allow new information to be provided to it and to be derived. For instance, it is not important how the association between an individual and a certain role filler is actually represented in terms of memory arrangements (called the symbol level). What counts for the underlying implementation is what operations must be supported in order to answer queries at the semantical level. This view

about KL-ONE-based representation systems was one of the major achievements of the KRYPTON project.

NIKL, PENNI, KL-TWO

At the same time as KRYPTON, the knowledge representation system NIKL was developed as a successor of KL-ONE. NIKL was a New Implementation of KL-ONE [Schmolze & Israel, 1983; Schmolze, 1985; Schmolze & Mark, 1991]. As discussed in [Kaczmarek et al., 1986] in NIKL roles are also ordered with respect to subsumption (see also [Schmidt, 1991, p. 13]).

The assertional components of KL-ONE have been initially discarded in the NIKL system (see the NIKL user guide [Robins, 1986]). Compared to the initial KL-ONE implementation, the algorithms in the NIKL classifier are faster in the average case because "obvious" information is exploited to a larger degree (see [MacGregor, 1988, p. 405] or [MacGregor, 1991a, p. 392]). However, the subsumption algorithm of NIKL is incomplete and it is hard to characterize which inferences are omitted [Schmolze & Israel, 1983] (see also [Patel-Schneider, 1987a, p. 74]).

Later, an assertional reasoning component was added with the system PENNI which is based on RUP [McAllester, 1982]. The resulting system was called KL-TWO [Vilain, 1985] (see also [Schmidt, 1991, p. 15]). In KL-TWO a propositional reasoner with equality (the PENNI subsystem) is augmented with a so-called quantificational reasoning component (the NIKL subsystem). For the propositional part in the PENNI component, incremental additions and retractions are supported due to the facilities provided by RUP. However, as shown in [Patel-Schneider, 1989b] the concept language of NIKL contains concept and role constructs that render the satisfiability problem for NIKL concept terms undecidable (see also [Schmidt-Schauss, 1989]).

Concerning hybrid reasoning, i.e. the systematic integration of TBox and ABox reasoning, there are shortcomings as well. Because in RUP different constants do not necessarily denote different objects, the unique name assumption is not built into the assertional component PENNI. Thus, number restrictions imposed by NIKL concepts often do not have the intended effects concerning hybrid reasoning. Other sources of incompleteness have been pointed out (see also the analysis of "inferential gaps" in [Nebel, 1990a, p. 63f.]). The research on the KL-TWO system demonstrated that hybrid reasoning is not just a matter of integrating reasoning subsystems at the software level. Hybrid reasoning requires a dedicated architecture implementing a sound and complete calculus which, in turn, can be developed only after a deep analysis of the semantics of the representation constructs. Nevertheless, the principle idea of exploiting subsumption information for resolution-based first-order reasoning has been integrated in many theorem proving systems.

KANDOR

Research on KANDOR [Patel-Schneider, 1984] was influenced by the KRYPTON architecture and the performance problems of the NIKL approach. The goal of KANDOR was to increase the expressive power of the terminological representation component in such a way that an efficient subsumption algorithm could be developed. Basically, KANDOR supported conjunction, value restriction and number restrictions as concept-forming operators. In minimum number restrictions, range restricted roles could be used (qualifying minimum number restriction, see also [Patel-Schneider, 1987a, p. 76]). In order to provide effective inference algorithms (e.g. for an information retrieval scenario) in the KANDOR approach the expressiveness of the assertional component was cut down to a representation system comparable to a database (without revision mechanisms). Subsumption in KANDOR was shown to be co-NP-complete (see [Nebel, 1988], and [Nebel, 1990a, p. 90] for details). The initially proposed subsumption algorithm with polynomial runtime must have been incomplete.

KANDOR was called a frame-based system (which might be reasonable because of the expressiveness offered by the ABox language). A frame in KANDOR was essentially a specification of conditions for describing how an individual can be an instance of it (in terms of superframes and restrictions). KANDOR supported defined frames and primitive frames. The system adopts the "small interfaces" approach of KRYPTON, i.e. models were built using the declaration interface (tell interface), and application services were realized with the query interface (ask interface). Although called a frame system, frames were not treated as record structures to be manipulated by procedural programs. The authors of KANDOR argued for a small knowledge representation system that could be used as part of larger systems with different subcomponents. The main achievement of KANDOR was the introduction of a small-can-be-beautiful approach which, finally, led to the design of the system CLASSIC which will be discussed in detail in the next section.

3.2 Second Generation DL Systems

Whereas the prototypical implementations of first generation systems have been used to study knowledge representation problems, second generation DL systems have been more extensively used in serious applications. The implementations discussed in this section are not only prototypes but are much more stable. In addition, at the beginning of the nineties, the systems have been called description logic systems. We first discuss systems for (almost) tractable languages based on (almost) complete algorithms.

Classic

The basic Classic system supported the logic $\mathcal{ALNFI}$h with TBoxes and ABoxes plus facilities for dealing with numbers [Borgida et al., 1989]. Classic is still available for research purposes. Implementation languages are Common Lisp and C. The interfaces are described in [Resnick et al., 1995]. Full Classic also contained the concept constructors $\mathcal{O}$ and $\mathcal{B}$ for referring to individuals in concept terms.

Subsumption in full Classic was initially assumed to be polynomial [Borgida et al., 1989]. Problems with individuals in full Classic was recognized in [Patel-Schneider et al., 1991]. At the same time Classic was shown to be co-NP complete [Lenzerini & Schaerf, 1991]. In the modified semantics for the concept constructors $\mathcal{O}$ and $\mathcal{B}$ (see [Borgida & Patel-Schneider, 1994]) the interpretation function maps individuals in concept terms to disjoint sets of domain objects. With this semantics concerning individuals the inference algorithms of the Classic system could be shown to be complete [Borgida & Patel-Schneider, 1994]. However, given the non-standard semantics for the concept constructors $\mathcal{O}$ and $\mathcal{B}$, the same effect can be achieved with exists restrictions for atomic concepts and disjunction of atomic concepts:[5] For each individual I a new atomic concept A_I can be introduced. Atomic concepts are also mapped to sets of individuals. Additionally, a set of axioms ensures that the new atomic concepts are disjoint. Now every term of the form $r : I$ can be replaced by $\exists r.A_I$. Terms of the form $\{I_1, \ldots, I_n\}$ can be replaced by $A_{I_1} \sqcup \ldots \sqcup A_{I_n}$. In an ABox, for each individual I a concept assertion is added to ensure that the individual is an instance of the associated atomic concept A_I. Thus, only in an ABox, a real coreference between roles can be enforced. On the one hand, we can call the Classic system "almost" complete. "Almost" refers to non-standard semantics w.r.t. individuals being supported by the current system implementations. On the other hand, the transformation makes clear that in Classic nevertheless a limited kind of disjunction (with concept names for which no definitions exist) can be expressed while retaining polynomial inference algorithms.

The recommended techniques for knowledge-based system development with Classic are outlined in [Brachman et al., 1991]. As Brachman [Brachman, 1992, p. 256] points out, a tractable description logic does not guarantee that a system is useful in practice. Therefore, the Classic system was also carefully designed to meet practical requirements and to guarantee predictable system behavior. The context in which the system was expected to be used requires that many queries are given to knowledge bases which rarely change. The architectural design of Classic supports a precomputation of index structures such that queries can be answered quickly (mostly by simple storage retrieval). The architecture is made possible by a careful selection of the concept and role constructors of the description logic language. In-

[5] Note that these concept constructors are not directly provided by Classic.

ference services for the description logic supported by CLASSIC can be implemented by transforming concept expressions into a normal form ("structural subsumption"). Once the normal form is computed, queries can be answered by inspecting the data structures used to encode the normal form. It should be noted that in CLASSIC retraction of told information is possible but not optimized.

Another facility offered by CLASSIC is a rule system. Rules are applied to individuals explicitly named in the ABox. Furthermore, rules are applied in a forward-chaining way. Basically, a rule has a precondition (a concept) and a conclusion (also a concept). If it can be shown that an individual mentioned in the ABox is an instance of the precondition concept, a concept assertion for stating the membership of the individual in the conclusion concept is added to the ABox. In order to provide support for modeling, the rule base is statically checked for inconsistencies. For instance, if there are two rules whose preconditions subsume each other, the conclusions must not be disjoint.

Furthermore, CLASSIC provides simple support for closed-world reasoning ([Resnick et al., 1995], see also [Weida, 1996]). Closing a role for an individual means adding an appropriate maximum number restriction for the role. The maximum number of fillers is restricted to the largest integer such that the minimum number restriction with this integer (and the corresponding role) is entailed by the knowledge base. The problem is that in combination with rules, the exact sequence of several closing operations determines what actually holds in the resulting ABox. These and other problems concerning different closing operations have to be considered with default reasoning as theoretical background [Baader & Hollunder, 1995a; Baader & Hollunder, 1995b; Donini et al., 1997b; Rosati, 1997]. For an integration of defaults into the CLASSIC system see [Wahlöf, 1996; Lambrix et al., 1998].

CLASSIC is one of the first systems that provides support for incorporating inferences over other domains. Consistency and subsumption checking for expressions of another domain (e.g. the reals) can be integrated into the CLASSIC system via an extension interface [Borgida et al., 1996]. CLASSIC was one of the first description logic systems being designed with respect to users which are non-experts in description logic theory. An important lesson learned by the CLASSIC approach was the importance of explanation and output pruning facilities [McGuinness & Borgida, 1995; McGuinness, 1996; Borgida & McGuinness, 1996]. Moreover, CLASSIC was the first system capable of supporting some reasonable form of error reporting [Brachman, 1992]. However, at the current state of the art there is hardly an adequate measure for the quality of these indispensible services [Brachman, 1992, p. 253].

Although CLASSIC is a very successful description logic modeling environment, the low expressiveness of the CLASSIC description logic made it hard to use the system in many kinds of applications. In many cases, users wanted more expressiveness [Patel-Schneider et al., 1990]. In the following sections we discuss systems for (more)

expressive description logics. As can be expected, increases in expressiveness came at a certain price. The predictability of the behavior of CLASSIC in terms of performance could not be reached by systems implementing complete algorithms. On the other hand, incomplete algorithms have the problem that results computed a system cannot be trusted in general. Thus, the complete-incomplete debate for expressive description logic systems started at the end of the eighties and the beginning of the nineties. First, we describe the systems LOOM and BACK, which are based on incomplete algorithms. Afterwards, research on description logic systems based on complete algorithms is summarized with a discussion of the systems $\mathcal{KRIS}$ and CRACK.

LOOM

The LOOM architecture [MacGregor & Bates, 1987; MacGregor, 1991b] offers TBox and ABox reasoning facilities for a description logic that can be characterized by the name $\mathcal{ALCQRIFO}$ plus additional constructs for dealing with numbers (see also [Brill, 1994] or [Horrocks, 1997, p. 43]). LOOM is based on KL-ONE, i.e. concept definitions with necessary or with necessary and sufficient conditions play an important role in domain modeling with LOOM. It should be emphasized that truth maintenance facilities for revision have been built into the LOOM architecture right from the beginning and have influenced the design of the whole system [MacGregor, 1988; MacGregor & Brill, 1992]. While first LOOM versions have been based on description logics [MacGregor & Brill, 1992] in later versions an attempt was made to develop a "description classifier for the Predicate Calculus" [MacGregor, 1994]. For instance, facilities for dealing with definitions for relations were added. However, the inference algorithms used in the LOOM system are known to be incomplete. One of the design goals of LOOM was to support rule-based programming [Yen et al., 1991b; Yen et al., 1991a; MacGregor & Burstein, 1991]. Based on the rule system, it is possible to specify additional necessary conditions for individuals which (i) are explicitly mentioned in the ABox and (ii) are derived to be instances of a certain defined concept. The additional necessary conditions are called "implications" in LOOM [MacGregor, 1988]. Note however, that these additional necessary conditions are not exploited for TBox reasoning, i.e. these "implications" are not to be confused with generalized concept inclusions which are standard in newer systems.

In order to meet the performance requirements of the applications for which LOOM has been developed (e.g. natural language and image interpretation), incomplete algorithms for concept consistency and subsumption are implemented. Concerning ABox reasoning, the LOOM applications require specific strategies to avoid the computation of unused results. Rather than employing the usual forward-chaining strategy of computing the most-specific atomic concepts of which the ABox individuals are instances, LOOM uses a scheme that considers the queries being posed

to the system. Thus, backward-chaining strategies for query answering are used in the implementation [MacGregor & Brill, 1992]. However, for the rule system, it is important to detect whether an individual is an instance of a concept that is used as a precondition of a rule. In this case, forward-chaining techniques are exploited [MacGregor, 1991b; MacGregor & Brill, 1992]. The combination of forward-chaining and backward-chaining inferences can be specified for a certain application problem by "marking" concepts accordingly. The user can control the inference process by these means but he is also responsible for estimating the effects of these declarations.

The arguments for the LOOM approach can be summarized as follows: The intractability of the representation language can hardly be avoided to fulfill the requirements of users. Therefore, the idea is to support the features in one system rather than as a set of application-specific ad hoc supplements ("Where resides the scruffiness?" [MacGregor, 1991a, p. 396]). Obviously, incompleteness is no problem as long as the answers of the inference system are interpreted in the right way (i.e. "no" answers should not be trusted). Several researchers have argued that there is always the inherent danger that non-expert users either do not know this or might not recognize this as a potential danger (cf. the work on complete systems [Baader & Hollunder, 1991a; Baader & Hollunder, 1991b] discussed below). However, if a combinatorial explosion occurs in a complete algorithm, in practice, no result is available as well. Concerning incomplete algorithms for decidable description logics, similar arguments as for other modeling environments based on first-order logic can be mentioned: If, in a certain application, concept terms are checked for consistency and a combinatorial explosion occurs in complete algorithms, incomplete algorithms at least might provide some support e.g. for building a TBox. Just signalling a timeout during the execution of a complete algorithm that runs into a combinatorial explosion might result in less information. In this case, an incomplete algorithm might succeed in finding at least some inconsistencies. Note however, that in modern inference system technologies that support complete reasoning, incomplete reasoners are used as "preprocessors" in order to speed up the inference system (see the next chapter).

LOOM supports different kinds of individuals (classified instances, light instances, CLOS instances). For different kinds of instances different levels of inference services are supported. E.g., for classified instances, the set of most specific atomic concepts of which the classified individual is an instance is computed once new assertions are specified. Thus, for these instances, the rule-based forward chaining engine is triggered and possibly new assertions are automatically added to an ABox (for details see [MacGregor & Brill, 1992]).

A problem with the LOOM approach is that from a user perspective it is hard to characterize the source of the incompleteness of the LOOM reasoning algorithms (see the discussion in [Horrocks, 1997, p. 42]). Although the inference techniques

used in LOOM are characterized in [MacGregor, 1991b, p. 90], once a system is incomplete, there is no adequate measure for the "quality of service" in terms of an implementation-independent characterization. For instance, in CLASSIC the characterization of the incompleteness of the inference system concerning individual reasoning has been given in terms of a weak semantics for the offered representation constructs (see above). It should be noted that specifying the incompleteness on the semantical level is no trivial task. Not only incompleteness issues are important in this context. For instance, the theoretical background for giving a semantics for rule-based computations has only been analyzed recently [Donini et al., 1992; Donini et al., 1994; Donini et al., 1998].

Incomplete reasoning facilities might not only be a problem that has to be considered at the tell and ask interface. With an example we demonstrate that incomplete inference algorithms can have effects in situations a user might not be aware of: LOOM also supports closed-world reasoning. The strategy for closing a role for an individual is to count the number of known role fillers. However, in addition to the individuals explicitly mentioned in the ABox, existential restrictions and minimum number restrictions have to be considered. Assuming too few of these individuals might result in an inconsistency. This is demonstrated with a simple knowledge base example with the following ABox $\{i : \exists r.A \sqcap \exists r.B \sqcap \exists r.C, (i, j) : r\}$. Let us assume, in the TBox there exist axioms such that A is *implicitly* declared as disjoint from both concepts, B and C. In the LOOM system, specific reasoning techniques (e.g. "conditioning") are implemented to compute the number of necessary fillers. Closing the role r for i by adding $i : (\leq 1\, r)$ makes the ABox inconsistent. However, since LOOM is incomplete, it might be the case that the disjointness of A and B as well as A and C is not detected and, therefore, too few fillers are assumed to exist in the closing process. Thus, the added maximum number restriction might be too restrictive, i.e. the system is unsound if closed-world reasoning is employed. Note that the semantic basis of automatic closing of roles as offered by LOOM is hard to characterize for expressive representation languages. Obviously, closing the role r for i with $i : (\leq 2\, r)$ might be a candidate. However, closing the role r for i with $i : (\leq 3\, r)$ might also be possible. In this case we have more individuals but with less specific constraints.

The current version of LOOM is implemented in Common Lisp and is available for research purposes. A new system (called PowerLOOM) for Common Lisp as well as C and Java-based platforms is under way.

BACK and FLEX

Research on BACK (Berlin Advanced Computational Knowledge representation system) started in 1985 approximately at the same time as work on the LOOM system

was initiated. BACK can also be called a knowledge representation environment [von Luck et al., 1987; Peltason et al., 1989; Kindermann & Quantz, 1990; Peltason, 1991].

The description logic of the initial BACK system can be called $\mathcal{ALQRI}$. There was support for reasoning with numbers and attribute sets. Research on the inference algorithms for the basic BACK language has stimulated the development of theoretical results on the complexity of concept consistency reasoning (e.g., [Nebel, 1988; Nebel, 1990a]) as well as the semantics of cycles [Nebel, 1991]. Additionally, not only terminological reasoning was considered but an investigation was made on the development of a hybrid architecture consisting of a TBox and an ABox. Issues of integration and balancing in hybrid knowledge representation systems, namely balanced expressiveness and tight coupling in hybrid systems, are analyzed in [Nebel & von Luck, 1987; Nebel & von Luck, 1988]. Research on the BACK system helped to shape the current view on balanced representation schemes with TBox and ABox. TBox concept terms are also used in the ABox to assert information about individuals. In addition, distinct individuals are assumed to denote distinct objects. Hence, the number of role fillers can be counted and compared against number restrictions (this was also done in KRYPTON as pointed out by [Woods & Schmolze, 1992, p. 165]). The algorithms used in BACK for instance checking and instance retrieval are described in [Nebel & von Luck, 1987; Nebel & von Luck, 1988; Kindermann & Randi, 1990]. In general, the discussion of the problems of incomplete algorithms that has been sketched in the previous section also applies to the BACK system because the inference algorithms used in BACK are also known to be incomplete.

In order to provide a knowledge representation environment, the BACK architecture was designed to support incremental additions to the ABox. BACK was one of the first attempts to implement algorithms for reasoning about retractions of ABox assertions. BACK supports retraction of told information, also called literal retraction [Nebel, 1990a; Kindermann, 1992]. This is also supported in the LOOM system. ABox assertions can be retrieved from a database by automatically computing SQL queries [Schmiedel, 1993]. For the applications considered in the BACK project, reasoning about time was important. Therefore, an integration of temporal reasoning and terminological reasoning was investigated by several project members. Investigations about how to incorporate temporal reasoning into terminological reasoning are reported in [Schmiedel, 1988; Schmiedel, 1990; Schild, 1991b; Fischer, 1992; Neuwirth, 1993].

In the successor system FLEX [Quantz et al., 1995], incomplete algorithms have been implemented for the description logic $\mathcal{ALCQRIFO}$. Additionally, reasoning about equations and inequations concerning integers is supported. Furthermore, the FLEX system served as a testbed for investigating weighted defaults [Quantz & Royer, 1992]. The initial implementation of FLEX has been developed in Prolog. FLEX ++

is a reimplementation in C++. The implementation was faster, but for application knowledge bases the performance was not sufficient. Appropriate optimization techniques (see the next chapter) had not been investigated in the context of description logics at the time of the development of the FLEX implementation.

In general, it is quite difficult to compare different systems and knowledge representation environments because the services being offered and the representation languages are not standardized (see [Patel-Schneider & Swartout, 1993] for a proposal on standardizing representation languages and inference services). Experiences with system implementations indicated that either limited expressiveness or incompleteness of reasoning can lead to problems in some applications. Therefore, other researchers investigated the implementation of systems based on sound and complete algorithms (published at the end of the eighties and beginning of the nineties). One can consider [Schmidt-Schauss & Smolka, 1991] as a starting point of this development (see also [Donini et al., 1997a]). Based on tableaux calculi, practical description logic implementations have been developed. We discuss the architectures of the systems $\mathcal{KRIS}$ and CRACK.

$\mathcal{KRIS}$

The development of sound and complete reasoning systems for more expressive description logics started at the end of the eighties. One of the main developments in this direction is the system $\mathcal{KRIS}$. The approach of $\mathcal{KRIS}$ was to implement sound and complete algorithms for an expressive description logic but develop optimization techniques for TBox reasoning such that, in practice, reasonable performance could be expected. The description logic of $\mathcal{KRIS}$ is $\mathcal{ALCNF}$ [Baader & Hollunder, 1991a; Baader & Hollunder, 1991b]. As an addition, $\mathcal{KRIS}$ provides enumerated types ($\mathcal{O}$ operator) and an interface for reasoning about so-called concrete domains [Baader & Hanschke, 1991a; Baader & Hanschke, 1992] (e.g. linear inequations over the reals). Role conjunctions were supported with a prototype implementation. The focus of the work in the $\mathcal{KRIS}$ project was on TBox-Classification. Nevertheless, $\mathcal{KRIS}$ is one of the first systems also supporting sound and complete ABox reasoning in expressive description logics. Even multiple ABoxes can be handled. The implementation language of $\mathcal{KRIS}$ is Common Lisp (see [Hollunder et al., 1991] for a User's Guide and [Achilles et al., 1991] for a description of the graphical user interface).

The idea of optimizing TBox classification was to exploit "obvious" information concerning "told" superconcepts and primitive concepts. In many concept definitions of application knowledge bases the right-hand side is a conjunction with concept names and concept terms. The conjuncts which are concept names on the right-hand side are defined as the "told" subsumers. Another important point was to avoid recomputation of subsumption relations found in preceding computation steps. Thus,

caching and propagation techniques were implemented. Information is propagated in the subsumption lattice such that expensive subsumption tests are avoided where possible. $\mathcal{KRIS}$ is the first system for which systematic empirical tests have been carried out. The algorithms evaluated in [Baader et al., 1992; Baader et al., 1994] are still in use in modern description logic systems (see below). Extensions such as defaults have been investigated (see also [Baader & Hollunder, 1992; Baader & Hollunder, 1993; Hollunder, 1994]) but have not been implemented in $\mathcal{KRIS}$.

Although the benchmarks considered in [Baader et al., 1994] revealed that the performance of $\mathcal{KRIS}$ for TBox reasoning was comparable to that of other systems of that time, the more or less direct implementation of *nondeterministic* tableaux algorithms that were developed for proving the decidability of problems in the field of theoretical computer science with chronological backtracking as in $\mathcal{KRIS}$ leads to performance problems for many applications. One of the main results of the $\mathcal{KRIS}$ project was that sound and complete inference algorithms are an important starting point for research on optimized sound and complete algorithms for practical system development.

Crack

One of the main research goals of the system Crack was to implement sound and complete algorithms for dealing with inferences about individuals in concept terms. Rather than providing a non-standard semantics as in Classic (individuals are mapped onto sets of domain objects), in Crack, individuals are mapped to elements of the domain. Thus, coreferences also have to be considered in concept terms. Crack supports the description logic $\mathcal{ALCRIFO}$ [Bresciani et al., 1995]. The implementation of Crack is based on Common Lisp. Crack provided a web interface.

In a similar way as in $\mathcal{KRIS}$, obvious information is exploited in the architecture to some extent but, nevertheless, Crack is a direct implementation of the tableaux rules of the underlying calculus. In the middle of the nineties it became clear that sound and complete reasoning is needed for many applications but the employed inference techniques which have been developed for (manually) deriving decidability results, e.g. with tableaux algorithms, were not suited for direct implementation. Thus, it became clear that there is a long way to go from a decidability proof to a working system, which has good performance in the average case. In the next section, we describe research on new system architectures where this problem has been treated by different optimization techniques concerning informed search while still retaining soundness, completeness and termination.

Before we discuss research on next generation description logics that provide optimization techniques suitable for large-scale practical applications, we present an

overview of some additional systems with interesting features developed at the beginning of the nineties.

Other systems

The list of systems we have discussed in this chapter is certainly incomplete. At the beginning of the nineties, the large number of projects involved in the development of knowledge representation systems shows the importance of this area. Usually description logic systems are built around a core engine which is a consistency checker. However, there are other services to be supplied which are also important to make the systems usable in larger application projects.

Among other points, the graphical manipulation of representations has been investigated in the SB-ONE project [Allgayer, 1990; Kobsa, 1991b; Kobsa, 1991a]. The implementation language was Common Lisp. Techniques for graphical interfaces to support knowledge base development with SB-ONE are described in [Kalmes, 1988; Kalmes, 1990] (see also [Abrett & Burstein, 1987] for a description of the KREME system). Furthermore, in SB-ONE the use of contexts (also called partitions) was explored for user modeling applications in natural language generation.

Another important point for DL inference systems is persistency and transaction management. We have already discussed the BACK approach [Schmiedel, 1993] (see also [Borgida, 1995]). Additional investigations have been made in the K-REP system [Mays et al., 1991b; Mays et al., 1991a].

3.3 The Next Generation

The declarative nature of description logic modeling is even more important when problems are treated for which languages are required that are no longer tractable. Inspired by theoretical advances, e.g. for handling number restrictions, role conjunctions, generalized concept inclusions as well as cyclic axioms with descriptive semantics ($\mathcal{ALCNR}$ [Buchheit et al., 1993a]), transitive roles ($\mathcal{ALC}_{R^+}$ [Sattler, 1996]), role hierarchies and features ($\mathcal{ALCH}f_{R^+}$ [Horrocks, 1998]) at the end of the nineties another generation of sound and complete description logic systems was developed. The logic $\mathcal{ALCNH}_{R^+}$ introduced in the previous chapter extended this line of work. While for $\mathcal{ALC}_{R^+}$ and $\mathcal{ALCH}f_{R^+}$ only a calculus for concept consistency was developed, for $\mathcal{ALCNH}_{R^+}$ ABox consistency was considered right from the beginning in a similar way as for $\mathcal{ALCNR}$. However, with the addition of transitive roles $\mathcal{ALCNH}_{R^+}$ is more expressive than $\mathcal{ALCNR}$. Meanwhile, as mentioned before, the development of sound and complete calculi for even more expressive description logics than $\mathcal{ALCNH}_{R^+}$ is in progress. However, as we will see, optimization techniques for more expressive logics than $\mathcal{ALCNH}_{R^+}$ still have to be developed.

The new generation systems available today are FaCT, DLP and RACE. All systems can be called fast in the average case. Systematic benchmarking and performance competitions provided for stable implementations. However, rather than directly implementing the tableaux calculus used for the theoretical decidability proofs and complexity analyses, a rigorous investigation into methods for informed search was made for developing the next generation of description logic systems. The pioneering system was FaCT. Since details about optimization techniques are discussed in the next chapter in order to present the novelties of the RACE architecture, we just mention some features of the DL systems FaCT and DLP for the sake of completeness.

The system FaCT supports TBox reasoning for the description logic $\mathcal{ALCH}f_{R^+}$ [Horrocks, 1997; Horrocks, 1998]. Furthermore, an experimental version of FaCT also supports concept reasoning with inverse roles and qualified number restrictions (iFaCT, [Horrocks, 1999]). However, most of the optimization techniques used in FaCT had to be disabled in this version. FaCT has been used for applications in the medical domain. Recent work on applications concerning query containment [Horrocks et al., 1999a] has demonstrated the necessity for also supporting ABox reasoning.

Based on similar techniques as FaCT, the system DLP utilizes extended techniques for optimizations [Horrocks & Patel-Schneider, 1998b; Horrocks & Patel-Schneider, 1998c; Horrocks & Patel-Schneider, 1999; Patel-Schneider, 1999]. DLP supports TBox reasoning for the description logic $\mathcal{ALCN}_{trans}$. From a modal logic perspective, $\mathcal{ALCN}_{trans}$ can also be called propositional dynamic logic (PDL) with a restricted form of graded modalities, i.e. simple number restrictions. However, in the current version no generalized concept inclusions and no TBoxes with forward references are supported (e.g. more optimizations are possible, certain algorithms for dealing with GCIs are currently not implemented). ABoxes are not supported either.

3.4 Lessons Learned

Considering the evolving technology of description logic systems it becomes clear that at the end of the nineties there is an enormous interest in description logic reasoning systems. This is demonstrated by the quite large number of system implementations. Currently, all modern DL systems are based on sound and complete algorithms. Thus, system developers can really rely on all answers computed by a DL system. This positive trend has been initiated by the development of optimization techniques that ensure stable runtimes for average case inputs for real-world problems even if the worst-case complexity is exponential (see also below). The trend has been initiated by the landmark system FaCT.

The original idea of the tell and ask interface of Krypton is still realized in modern

systems. However, currently, the systems support only some kind of batch-oriented behavior. A knowledge base (TBox and ABox) is passed to the systems (tell interface). Afterwards, queries can be answered (ask interface). But, no incremental additions to the knowledge base are possible after the first query is answered. The difficulty is that complex transformations on the knowledge bases are necessary in order to compute an internal representation that can be used for relatively fast query answering (see the discussion on optimization techniques in subsequent chapters). The price to pay is that algorithms for appropriately handling incremental additions to a knowledge base are not yet known. Other features, e.g. explanation facilities, retraction etc. still have to be developed for expressive DLs as well.

As a second and quite important lesson one can see that description logics with more expressiveness and sound and complete algorithms impose a different view in modeling. Concept definitions as known from, for instance, CLASSIC are no longer the central modeling device if generalized concept inclusions (representing cyclic implications or equalities) are available.[6]

A third lesson we can learn from considering description logic systems and their development is that the implementation language is hardly important for the magnitude of speed (compared to the expressiveness of the description logic). What really counts is the set of optimization strategies, the implementation of index data structures and the selection of clever heuristics. There are first attempts to provide a distributed implementation of a description logic system. However, performance problems in network communication lead to server-based solutions, i.e. a knowledge base is being processed at a single workstation computer (but may be accessed from different clients). Benchmark generators and standardized application knowledge bases are used for metering system performance. Thus, different system implementations can be compared.

Another lesson is that the development of techniques for incorporating space and time into description logics is still an open issue. The necessity of a semantics-based integration of temporal and terminological reasoning has been emphasized in first investigations in the BACK project. However, early approaches (e.g. [Schmiedel, 1990]) have been shown to be undecidable [Halpern & Shoham, 1991; Schild, 1993]. In the context of planning, the opportunities of an integrated environment combining temporal and terminological reasoning were clearly demonstrated with the RHET system [Allen, 1991]. It has been shown that spatial reasoning (e.g. about topological relations) induces non-obvious subsumption relations between concept terms [Haarslev & Möller, 1997a]. We will return to this topic in Chapter 6.

[6]Nevertheless, description logics can still be called object-based representation formalisms, although there are some approaches to deal with n-ary relations [Schmolze, 1989; Calvanese et al., 1998] as well.

Chapter 4

RACE: From a Tableaux Calculus to an Inference System

This chapter investigates optimization techniques for practical reasoning with expressive ABox description logic systems. We discuss optimization techniques well known in the field and introduce new techniques for TBox and ABox reasoning. The analysis is based on the description logic $\mathcal{ALCNH}_{R^+}$ described in Chapter 2. In order to empirically evaluate optimization techniques for the $\mathcal{ALCNH}_{R^+}$ tableaux calculus, the DL system RACE has been developed [Haarslev et al., 1999c]. RACE implements an $\mathcal{ALCNH}_{R^+}$ reasoner for answering queries concerning ABoxes and TBoxes (with generalized concept inclusions, GCIs) and can handle knowledge bases that cannot be processed with FaCT or DLP.[1]

It is demonstrated that optimization techniques developed for testing concept consistency and concept subsumption [Horrocks & Patel-Schneider, 1999] scale up for testing ABox consistency and can be directly integrated into an ABox reasoning architecture. We introduce and analyze several optimization techniques which are either novel or significantly extend the techniques presented in [Horrocks & Patel-Schneider, 1999]. RACE supports the logic $\mathcal{ALCNH}_{R^+}$ because (i) $\mathcal{ALCNH}_{R^+}$ is an expressive DL which can be used for solving many application problems (see below), and (2) it was the design philosophy of RACE is to support only those representation constructs for which optimization techniques were available or could be developed.

As we have seen in Section 2.3 all standard inference problems in $\mathcal{ALCNH}_{R^+}$ can be reduced to knowledge base consistency. In the following a calculus to decide the consistency of an $\mathcal{ALCNH}_{R^+}$ knowledge base $(\mathcal{T}, \mathcal{R}, \mathcal{A})$ is devised. The first publication with a calculus for this language is [Haarslev & Möller, 2000b]. Basically,

[1] RACE is freely available for research purposes [Haarslev & Möller, 1999e].

the calculus is given as set of non-deterministic rules. Thus, in an implementation of the non-deterministic algorithm, dedicated search techniques are required (see below).

4.1 A Tableaux Calculus for $\mathcal{ALCNH}_{R^+}$

As a first step the original ABox $\mathcal{A}$ of the knowledge base is transformed w.r.t. the TBox $\mathcal{T}$. The idea is to derive an ABox $\mathcal{A}_\mathcal{T}$ that is consistent w.r.t. an RBox $\mathcal{R}$ (and an empty TBox) iff $(\mathcal{T}, \mathcal{R}, \mathcal{A})$ is consistent. The calculus introduced below is applied to $\mathcal{A}_\mathcal{T}$ and the role box $\mathcal{R}$.

In order to define the transformation steps for deriving $\mathcal{A}_\mathcal{T}$, we have to introduce a few technical terms. First, for any concept term we define its negation normal form.

Definition 15 (Negation Normal Form) A concept is in negation normal form iff negation signs occur only in front of concept names.

Proposition 16 Every $\mathcal{ALCNH}_{R^+}$ concept term C can be transformed into an equisatisfiable concept $nnf(\mathsf{C})$ in negation normal form by recursively applying the following transformation rules to subconcepts from left to right. If no rule is applicable, the resulting concept is in negation normal form and all models of C are also models of $nnf(\mathsf{C})$ and vice versa. The transformation is possible in linear time.

- $\neg\neg\mathsf{C} \rightarrow \mathsf{C}$
- $\neg(\mathsf{C} \sqcap \mathsf{D}) \rightarrow \neg\mathsf{C} \sqcup \neg\mathsf{D}$
- $\neg(\mathsf{C} \sqcup \mathsf{D}) \rightarrow \neg\mathsf{C} \sqcap \neg\mathsf{D}$
- $\neg\forall\,\mathsf{R}\,.\,\mathsf{C} \rightarrow \exists\,\mathsf{R}\,.\,\neg\mathsf{C}$
- $\neg\exists\,\mathsf{R}\,.\,\mathsf{C} \rightarrow \forall\,\mathsf{R}\,.\,\neg\mathsf{C}$
- $\neg\exists_{\leq m}\,\mathsf{S} \rightarrow \exists_{\geq m+1}\,\mathsf{S}$
- $\neg\exists_{\geq m}\,\mathsf{S} \rightarrow \exists_{\leq m-1}\,\mathsf{S}$

In order to deal with GCI and number restrictions, some additional ABox assertions are used internally.

Definition 17 (Additional ABox Assertions) Let C be a concept term, $\mathsf{a}, \mathsf{b} \in O$ be individual names, then the following expressions are also assertional axioms:

- $\forall\, x\,.\, x\!:\!\mathsf{C}$ *(universal concept assertion)*,[2]
- $\mathsf{a} \neq \mathsf{b}$ *(inequality assertion)*.

[2] $\forall\, x\,.\, x\!:\!\mathsf{C}$ is to be read as $\forall\, x\,.\,(x\!:\!\mathsf{C})$.

An interpretation $\mathcal{I}$ satisfies an assertional axiom $\forall\, x\,.\, x\!:\!\mathsf{C}$ iff $\mathsf{C}^{\mathcal{I}} = \Delta^{\mathcal{I}}$ and $\mathsf{a} \neq \mathsf{b}$ iff $\mathsf{a}^{\mathcal{I}} \neq \mathsf{b}^{\mathcal{I}}$.

We are now ready to define an augmented ABox as input to the tableaux rules.

Definition 18 (Augmented ABox) For an initial ABox $\mathcal{A}$ we define its *augmented* ABox $\mathcal{A}_{\mathcal{T}}$ w.r.t a TBox $\mathcal{T}$ by applying the following transformation rules to $\mathcal{A}$. Then, for every GCI $\mathsf{C} \sqsubseteq \mathsf{D}$ in $\mathcal{T}$ the assertion $\forall\, x\,.\, x\!:\!(\neg\mathsf{C} \sqcup \mathsf{D})$ is added to $\mathcal{A}$. Every concept term occurring in $\mathcal{A}$ is transformed into its negation normal form. Let $O_{\mathcal{A}} = \{\mathsf{a}_1, \ldots, \mathsf{a}_\mathsf{n}\}$ be the set of individuals mentioned in $\mathcal{A}$, then the set of inequality assertions $\{\mathsf{a}_\mathsf{i} \neq \mathsf{a}_\mathsf{j} \mid \mathsf{a}_\mathsf{i}, \mathsf{a}_\mathsf{j} \in O_{\mathcal{A}},\, \mathsf{i}, \mathsf{j} \in 1..\mathsf{n}, \mathsf{i} \neq \mathsf{j}\}$ is added to $\mathcal{A}$.

In order to check the consistency of an $\mathcal{ALCNH}_{R^+}$ knowledge base $(\mathcal{T}, \mathcal{R}, \mathcal{A})$ the augmented ABox $\mathcal{A}_{\mathcal{T}}$ is computed. Then, the tableaux rules are applied to the augmented ABox $\mathcal{A}_{\mathcal{T}}$ and a role box $\mathcal{R}$. The rules are applied in accordance with a completion strategy (see below).

Lemma 19 A knowledge base $(\mathcal{T}, \mathcal{R}, \mathcal{A})$ is consistent if and only if $\mathcal{A}_{\mathcal{T}}$ is consistent w.r.t. the role box $\mathcal{R}$ (and an empty TBox).

Proof. "$\Rightarrow$" Since $(\mathcal{T}, \mathcal{R}, \mathcal{A})$ is consistent there exists a model $\mathcal{I}_{\mathcal{D}} = (\Delta^{\mathcal{I}}, \Delta^{\mathcal{D}}, \cdot^{\mathcal{I}})$ such that $\forall\, \mathsf{C} \sqsubseteq \mathsf{D} \in \mathcal{T} : \mathsf{C}^{\mathcal{I}} \subseteq \mathsf{D}^{\mathcal{I}}$. This is equivalent to $\forall\, \mathsf{a} \in \Delta^{\mathcal{I}} : \mathsf{a} \in \mathsf{C}^{\mathcal{I}} \implies \mathsf{a} \in \mathsf{D}^{\mathcal{I}}$. Hence, $\forall\, \mathsf{a} \in \Delta^{\mathcal{I}} : \mathsf{a} \in (\neg\mathsf{C})^{\mathcal{I}} \vee \mathsf{a} \in \mathsf{D}^{\mathcal{I}}$ or $\forall\, \mathsf{a} \in \Delta^{\mathcal{I}} : \mathsf{a} \in (\neg\mathsf{C} \sqcup \mathsf{D})^{\mathcal{I}}$. In other words: $(\neg\mathsf{C} \sqcup \mathsf{D})^{\mathcal{I}} = \Delta^{\mathcal{I}}$. Thus, due to the semantics given above $\forall\, x\,.\, x\!:\!\neg\mathsf{C} \sqcup \mathsf{D}$ is also satisfied.

"$\Leftarrow$" This can be shown by applying the arguments in the other direction.

Since all terminological axioms in $\mathcal{T}$ are appropriately represented in $\mathcal{A}_{\mathcal{T}}$, a model for both $\mathcal{A}_{\mathcal{T}}$ and $\mathcal{R}$ is also a model for $\mathcal{A}$, $\mathcal{T}$ and $\mathcal{R}$ and vice versa. □

The tableaux rules require the notion of blocking their applicability. This is based on so-called concept sets, an ordering for new individuals and the notion of a blocking individual.

Definition 20 (Concept Set) Given an ABox $\mathcal{A}$ and an individual a occurring in $\mathcal{A}$, we define the *concept set* of a as $\sigma(\mathcal{A}, \mathsf{a}) := \{\mathsf{C} \mid \mathsf{a}\!:\!\mathsf{C} \in \mathcal{A}\}$.

Definition 21 (Ordering) We define an *individual ordering* '$\prec$' for new individuals (elements of O_N) occurring in an ABox $\mathcal{A}$. If $\mathsf{b} \in O_N$ is introduced in $\mathcal{A}$, then $\mathsf{a} \prec \mathsf{b}$ for all new individuals a already present in $\mathcal{A}$.

Definition 22 (Blocking Individual, blocked) Let $\mathcal{A}$ be an ABox and $\mathsf{a}, \mathsf{b} \in O_N$ be individuals in $\mathcal{A}$. We call a the *blocking individual* of b if the following conditions hold:

1. $\sigma(\mathcal{A}, \mathsf{a}) \supseteq \sigma(\mathcal{A}, \mathsf{b})$
2. $\mathsf{a} \prec \mathsf{b}$

If a is a blocking individual for b, then b is said to be *blocked* by a.

We are now ready to define the *completion rules* that are intended to generate a so-called completion (see also below) of an ABox $\mathcal{A}_\mathcal{T}$ w.r.t. an RBox $\mathcal{R}$. From this point on, if we refer to an ABox $\mathcal{A}$, we always consider ABoxes derived from $\mathcal{A}_\mathcal{T}$.

Definition 23 (Completion Rules)

R⊓ The conjunction rule.
if 1. $\mathsf{a}\!:\!\mathsf{C} \sqcap \mathsf{D} \in \mathcal{A}$, and
2. $\{\mathsf{a}\!:\!\mathsf{C},\ \mathsf{a}\!:\!\mathsf{D}\} \not\subseteq \mathcal{A}$
then $\mathcal{A}' = \mathcal{A} \cup \{\mathsf{a}\!:\!\mathsf{C},\ \mathsf{a}\!:\!\mathsf{D}\}$

R⊔ The disjunction rule (nondeterministic).
if 1. $\mathsf{a}\!:\!\mathsf{C} \sqcup \mathsf{D} \in \mathcal{A}$, and
2. $\{\mathsf{a}\!:\!\mathsf{C},\ \mathsf{a}\!:\!\mathsf{D}\} \cap \mathcal{A} = \emptyset$
then $\mathcal{A}' = \mathcal{A} \cup \{\mathsf{a}\!:\!\mathsf{C}\}$ **or** $\mathcal{A}' = \mathcal{A} \cup \{\mathsf{a}\!:\!\mathsf{D}\}$

R∀C The role value restriction rule.
if 1. $\mathsf{a}\!:\!\forall\,\mathsf{R}\,.\,\mathsf{C} \in \mathcal{A}$, and
2. $\exists\,\mathsf{b} \in O, \mathsf{S} \in \mathsf{R}^{\downarrow} : (\mathsf{a}, \mathsf{b})\!:\!\mathsf{S} \in \mathcal{A}$, and
3. $\mathsf{b}\!:\!\mathsf{C} \notin \mathcal{A}$
then $\mathcal{A}' = \mathcal{A} \cup \{\mathsf{b}\!:\!\mathsf{C}\}$

R∀$_+$C The transitive role value restriction rule.
if 1. $\mathsf{a}\!:\!\forall\,\mathsf{R}\,.\,\mathsf{C} \in \mathcal{A}$, and
2. $\exists\,\mathsf{b} \in O, \mathsf{T} \in \mathsf{R}^{\downarrow}, \mathsf{T} \in T, \mathsf{S} \in \mathsf{T}^{\downarrow} : (\mathsf{a}, \mathsf{b})\!:\!\mathsf{S} \in \mathcal{A}$, and
3. $\mathsf{b}\!:\!\forall\,\mathsf{T}\,.\,\mathsf{C} \notin \mathcal{A}$
then $\mathcal{A}' = \mathcal{A} \cup \{\mathsf{b}\!:\!\forall\,\mathsf{T}\,.\,\mathsf{C}\}$

R∀$_x$ The universal concept restriction rule.
if 1. $\forall\,x\,.\,x\!:\!\mathsf{C} \in \mathcal{A}$, and
2. $\exists\,\mathsf{a} \in O$: a mentioned in $\mathcal{A}$, and
3. $\mathsf{a}\!:\!\mathsf{C} \notin \mathcal{A}$
then $\mathcal{A}' = \mathcal{A} \cup \{\mathsf{a}\!:\!\mathsf{C}\}$

R∃C The role exists restriction rule (generating).
if 1. $\mathsf{a}\!:\!\exists\,\mathsf{R}\,.\,\mathsf{C} \in \mathcal{A}$, and
2. $\mathsf{a} \in O_N \Rightarrow (\neg\exists\,\mathsf{c}$ in $\mathcal{A}$: $\mathsf{c} \in O_N, \mathsf{c}$ is a blocking individual for $\mathsf{a})$, and
3. $\neg\exists\,\mathsf{b} \in O, \mathsf{S} \in \mathsf{R}^{\downarrow} : \{(\mathsf{a}, \mathsf{b})\!:\!\mathsf{S},\ \mathsf{b}\!:\!\mathsf{C}\} \subseteq \mathcal{A}$
then $\mathcal{A}' = \mathcal{A} \cup \{(\mathsf{a}, \mathsf{b})\!:\!\mathsf{R},\ \mathsf{b}\!:\!\mathsf{C}\}$ where $\mathsf{b} \in O_N$ is not used in $\mathcal{A}$

$\mathbf{R}\exists_{\geq n}$ The number restriction exists rule (generating).
if 1. $\mathsf{a}\colon\exists_{\geq n}\,\mathsf{R} \in \mathcal{A}$, and
2. $\mathsf{a} \in O_N \Rightarrow (\neg\exists\,\mathsf{c}$ in $\mathcal{A}\colon \mathsf{c} \in O_N, \mathsf{c}$ is a blocking individual for $\mathsf{a})$, and
3. $\neg\exists\,\mathsf{b_1}, \ldots, \mathsf{b_n} \in O, \mathsf{S_1}, \ldots, \mathsf{S_n} \in \mathsf{R}^{\downarrow}$:
$\{(\mathsf{a}, \mathsf{b_k})\colon \mathsf{S_k} \mid \mathsf{k} \in 1..\mathsf{n}\} \cup \{\mathsf{b_i} \neq \mathsf{b_j} \mid \mathsf{i}, \mathsf{j} \in 1..\mathsf{n}, \mathsf{i} \neq \mathsf{j}\} \subseteq \mathcal{A}$

then $\mathcal{A}' = \mathcal{A} \cup \{(\mathsf{a}, \mathsf{b_k})\colon \mathsf{R} \mid \mathsf{k} \in 1..\mathsf{n}\} \cup \{\mathsf{b_i} \neq \mathsf{b_j} \mid \mathsf{i}, \mathsf{j} \in 1..\mathsf{n}, \mathsf{i} \neq \mathsf{j}\}$
where $\mathsf{b_1}, \ldots, \mathsf{b_n} \in O_N$ are not used in $\mathcal{A}$

$\mathbf{R}\exists_{\leq n}$ The number restriction merge rule (nondeterministic).
if 1. $\mathsf{a}\colon\exists_{\leq n}\,\mathsf{R} \in \mathcal{A}$, and
2. $\exists\,\mathsf{b_1}, \ldots, \mathsf{b_m} \in O, \mathsf{S_1}, \ldots, \mathsf{S_m} \in \mathsf{R}^{\downarrow}\colon \{(\mathsf{a}, \mathsf{b_1})\colon \mathsf{S_1}, \ldots, (\mathsf{a}, \mathsf{b_m})\colon \mathsf{S_m}\} \subseteq \mathcal{A}$
with $\mathsf{m} > \mathsf{n}$, and
3. $\exists\,\mathsf{b_i}, \mathsf{b_j} \in \{\mathsf{b_1}, \ldots, \mathsf{b_m}\} : \mathsf{i} \neq \mathsf{j}$, $\mathsf{b_i} \neq \mathsf{b_j} \notin \mathcal{A}$

then $\mathcal{A}' = \mathcal{A}[\mathsf{b_i}/\mathsf{b_j}]$, i.e. replace every occurrence of $\mathsf{b_i}$ in $\mathcal{A}$ by $\mathsf{b_j}$

We call the rules R$\sqcup$ and R$\exists_{\leq n}$ *nondeterministic* rules since they can be applied in different ways to the same ABox. The remaining rules are called *deterministic* rules. Moreover, we call the rules R$\exists$C and R$\exists_{\geq n}$ *generating* rules since they are rules that can introduce new individuals.

Given an ABox $\mathcal{A}$, more than one rule might be applicable to $\mathcal{A}$. Rule application is controlled by a completion strategy in accordance to the ordering for new individuals (see Definition 21).

Definition 24 (Completion Strategy) We define a *completion strategy* that must observe the following restrictions.

- Meta rules:
  - Apply a rule to an individual $\mathsf{b} \in O_N$ only if no rule is applicable to an individual $\mathsf{a} \in O_O$.
  - Apply a rule to an individual $\mathsf{b} \in O_N$ only if no rule is applicable to another individual $\mathsf{a} \in O_N$ such that $\mathsf{a} \prec \mathsf{b}$.
- The completion rules are always applied in the following order. A step is skipped in case the corresponding set of applicable rules is empty.
  1. Apply all nongenerating rules (R$\sqcap$, R$\sqcup$, R$\forall$C, R$\forall_+$C, R$\forall_x$, R$\exists_{\leq n}$) as long as possible.
  2. Apply a generating rule (R$\exists$C, R$\exists_{\geq n}$) and restart with step 1 as long as possible.

In the following we always assume that rules are applied in accordance to this strategy. It ensures that the rules are applied to new individuals w.r.t. the ordering '$\prec$' which enforces a kind of breadth-first order.

Despite of the completion strategy, there might be more than rule applicable to different ABox assertions (cf. the R$\exists_{\leq n}$ rule) or a rule application results in more than one possible conclusion ABox (cf. the rule R$\sqcup$). In a computer program, nondeterminism must be implemented with search techniques. The branches of the search space are referred to as *choice points*. In a concrete implementation the application of rules stops immediately and backtracks to (possibly) remaining choice points, if a so-called clash is discovered.

Definition 25 (Clash, Clash Triggers, Completion) We assume the same naming conventions as used above. An ABox $\mathcal{A}$ contains a *clash* if one of the following *clash triggers* is applicable. If none of the clash triggers is applicable to $\mathcal{A}$, then $\mathcal{A}$ is called *clash-free*.

- *Primitive clash*: $\{\mathsf{a}:\mathsf{C}, \mathsf{a}:\neg\mathsf{C}\} \subseteq \mathcal{A}$
- *Number restriction merging clash*:
  $\exists \mathsf{S}_1, \ldots, \mathsf{S}_\mathsf{m} \in \mathsf{R}^{\downarrow} : \{\mathsf{a}:\exists_{\leq n}\mathsf{R}\} \cup \{(\mathsf{a}, \mathsf{b}_\mathsf{i}):\mathsf{S}_\mathsf{i} \mid \mathsf{i} \in 1..\mathsf{m}\} \cup$
  $\{\mathsf{b}_\mathsf{i} \neq \mathsf{b}_\mathsf{j} \mid \mathsf{i}, \mathsf{j} \in 1..\mathsf{m}, \mathsf{i} \neq \mathsf{j}\} \subseteq \mathcal{A}$ with $\mathsf{m} > \mathsf{n}$

A clash-free ABox $\mathcal{A}$ is called *complete* if no completion rule is applicable to $\mathcal{A}$. A complete ABox $\mathcal{A}'$ derived from an ABox $\mathcal{A}$ is also called a *completion* of $\mathcal{A}$.

Any ABox containing a clash is obviously unsatisfiable (w.r.t. an RBox $\mathcal{R}$). The purpose of the calculus is to generate a completion for an initial ABox $\mathcal{A}_\mathcal{T}$ that proves the consistency of $\mathcal{A}_\mathcal{T}$ (w.r.t. an RBox $\mathcal{R}$) or its inconsistency if no completion can be found.

As mentioned before, soundness and completeness (and termination) of this calculus has been published in [Haarslev & Möller, 2000b]. Hence, ABox reasoning in $\mathcal{ALCNH}_{R^+}$ is decidable. Since in Chapter 5 of this Habilitation Thesis the decidability of an extension of $\mathcal{ALCNH}_{R^+}$ is proven, the proofs given in [Haarslev & Möller, 2000b] are not repeated here.

4.2 Optimization Techniques for Inference Algorithms

On the one hand, the tableaux calculus introduced in the previous section is of theoretical interest for proving the decidability of the ABox consistency problem. On the other hand, a tableaux calculus provides the basis for developing an implementation of the calculus as part of an inference system. However, for practical purposes

such calculi are highly inefficient if nondeterminism is handled with simple backtracking search techniques. Therefore, the development of optimization techniques is an important research topic. This section briefly summarizes some of the already established optimization techniques known in the field in order to demonstrate the effectiveness of additional optimization techniques that are integrated into the architecture of the RACE system. RACE uses a highly optimized variant of the calculus for $\mathcal{ALCNH}_{R^+}$ supported by corresponding data structures.

These techniques form the basis of other 'modern' DL system implementations (e.g. FaCT and DLP [Horrocks & Patel-Schneider, 1999]). The techniques were developed for reasoners deciding only the concept consistency problem. With RACE we demonstrate that these techniques scale up and can be integrated into an ABox reasoning architecture [Haarslev & Möller, 1999b]. In order to explain the effect of new optimizations, some of the known techniques need to reviewed in the following paragraphs.[3]

For the introduction of the optimization techniques a few definitions are required. For brevity we refer to the ABox assertions involved in an application of the disjunction rule R$\sqcup$ or the number restriction merge rule R$\exists_{\leq n}$ as *or-constraints*. In addition, other ABox assertions used in a consistency proof are also referred to as *constraints* (see also [Schmidt-Schauss & Smolka, 1991] or [Buchheit et al., 1993a]). An alternative introduced by an or-constraint is also called a *disjunct*. Applying a rule to a set of constraints is also called *expanding the constraints*. If a completion can be derived based on the expansion of a constraint, the constraint (or a set of constraints) is said to be *satisfiable*. Generating rules (R$\exists$C and R$\exists_{\geq n}$) introduce constraints for new individuals. For brevity, the contraints for a new individual introduced by a generating rule are referred to as a *subtableau*.

4.2.1 Optimizing Concept Consistency Reasoning

In this section a very brief overview is given about the state-of-the-art techniques that have been incorporated into the RACE architecture. For detailed examples see the cited literature.

Normalization and Encoding

Normalization of concepts ensures that concepts are represented in a canonical forms. For instance, for the concept terms $\mathsf{A} \sqcap \mathsf{B}$ and $\mathsf{B} \sqcap \mathsf{A}$ the form $\mathsf{A} \sqcap \mathsf{B}$ might be chosen as a normalization. Encoding ensures that for each concept term with the same

[3]See also work on stochastic search techniques for proving the satisfiability of formulae [Selman et al., 1992]. Stochastic search techniques are incomplete, i.e. stochastic techniques can prove the consistency of a formula but not its inconsistency. Nevertheless, stochastic techniques could be investigated as optimization techniques in future work.

normal form, a unique identifier is used. In the FaCT system a number is used. The negation of a concept with number n is always encoded with the number $n+1$. The concept number can be used to index an array of records with additional information about concepts. For software engineering purposes, in RACE a different strategy is used. Rather than using numbers, which might be hard to interpret for debugging purposes, RACE encodes concepts with pointers to record structures. Concepts with the same normal form are represented by the same pointer. Each record structure contains a reference to the negated concept. Thus, the implementation of clash detection is a mere search for a pointer (if no number restrictions are involved). In RACE, there is no necessity to predefine the number of concepts that can be handled by allocating an array of a certain size. Due to the object-oriented architecture of Common Lisp, special print methods can be declared to provide appropriate output in situations where lists of concepts are to be debugged with an inspector toolkit.

The Trace Technique

Let us assume that the concept term $\exists\,\mathsf{R}\,.\,\mathsf{A} \sqcap \forall\,\mathsf{R}\,.\,\mathsf{B} \sqcap \exists\,\mathsf{S}\,.\,\mathsf{C} \sqcap \forall\,\mathsf{S}\,.\,\mathsf{D}$ is to be checked for consistency. It can be easily verified that the test can be carried out by considering the two subproblems $\mathsf{A} \sqcap \mathsf{B}$ and $\mathsf{C} \sqcap \mathsf{D}$ in succession. Thus, it is not necessary to actually represent the whole search space at a time. After checking the consistency of $\mathsf{A} \sqcap \mathsf{B}$ all constraints generated during the proof can be discarded before the consistency of $\mathsf{C} \sqcap \mathsf{D}$ is checked. For each of the subproblems, the same idea can be applied recursively. Thus, it is only required that a certain "trace" of the search space is actually represented in memory. Note that not only the polynomial space requirements are important. Reducing the number of ABox assertions (or constraints) to be considered at a time is a great advantage from an implementation-oriented point of view. Thus, all DL systems employ this technique whenever possible. A prerequisite of the trace technique is that the R$\forall$C rule is always applied right after (or together with) the generating rules R$\exists$C and R$\exists_{\geq n}$.

For description logic $\mathcal{ALC}$, the trace technique has been first presented in [Schmidt-Schauss & Smolka, 1991]. The applicability of the idea to more expressive languages such as $\mathcal{ALCR}$ and $\mathcal{ALCN}$ is shown in [Hollunder & Nutt, 1990]. For the logic $\mathcal{ALCH}f_{R^+}$ a variant of the trace technique is also applicable but due to the interaction of features and role hierarchies, the computation of the concept terms that must be tested in combination, i.e. the determination of the subproblems (partitions), is much more complex because, due to features, merging operations are necessary (for details see [Horrocks, 1997][p. 74]. For the logic $\mathcal{ALCNH}_{R^+}$ with number restrictions similar techniques for determining a partition to be tested as a subproblems is even more complex. At-most restrictions along the role hierarchy have to be taken into consideration. For ABoxes not only concept terms but also role assertions have to be considered.

Dependency-Directed Backtracking

As indicated above, implementing a nondeterministic algorithm on a practical computer requires search. Naive backtracking algorithms (i.e. jumping always to the last choice point) often explore regions of the search space rediscovering the same contradictions (clashes) repeatedly. Dependency-directed backtracking records the dependencies of expanded constraints and in case of a clash backtracks to or-constraints that are responsible for at least one of the clash culprits in the subtree. The initial ideas behind dependency-directed backtracking have been published in [Stallman & Sussman, 1977].

In the FaCT system implementation [Horrocks, 1997] dependency-directed backtracking was first systematically integrated into a DL tableaux algorithm (see also [Drollinger, 1993] about ideas about how to integrate dependency-directed backtracking into DL reasoning algorithms). In FaCT and RACE recursion is used to store the "state" involved in a choice point, i.e. in accordance with the completion strategy a recursive procedure applies the tableaux rules to a set of ABox assertions (constraints). After a constraint is expanded by applying a rule, the procedure is recursively called to process the new set of constraints. Before applying a rule, the procedure checks for a clash. If a clash is found, the procedure backtracks in such a way that the control flow continues at an appropriate choice point set up by a nondeterministic rule. This form of dependency-directed backtracking is also called *backjumping* [Ginsberg, 1993; Horrocks, 1997].

The backjumping point is controlled by a list of so-called *catcher constraints*. The catchers are those constraints to which a nondeterministic rule is applied. When a constraint is expanded, and a new constraint is generated by applying a rule, the precondition constraints of the rule are pushed onto the list of dependencies of the new constraints. In the RACE implementation, every or-constraint on the dependencies of a precondition constraint is also pushed onto the list of dependencies of the resulting constraint (dependency propagation). When a clash is found, the dependencies of the clash culprits (i.e. the or-constraints that generated these constraints) are stored on the list of catchers and backjumping is started. Thus, the control flow falls back out of recursion. Whenever a choice point is encountered during backjumping, it is checked whether the involved or-constraint is responsible for a clash culprit. If the or-constraint is *not* found in the list of catchers, the choice point can be safely bypassed. In case the or-constraint is found either the remaining alternatives are tried or the or-constraint is considered as unsatisfiable in the current subtree. The backtracking continues but removes the current or-constraint from the list of clash dependencies and adds the saved clash dependencies of the previously unsatisfiable alternatives.

The basic idea of dependency-directed backtracking implemented in RACE is due to

FaCT [Horrocks, 1997]. However, RACE provides support for number restrictions and, therefore, the dependency tracking mechanism had to be extended w.r.t. the rule $R\exists_{\leq n}$ and w.r.t. ABox reasoning.

One insight of empirical investigations with the FaCT system is that dependency-directed backtracking is indeed a very useful optimization technique for many description logic problems (as opposed to the results in [Freeman, 1995; Truemper, 1998] for propositional logic). Experiments with FaCT, DLP and RACE demonstrated that disabling dependency-directed backtracking does not make much sense and, therefore, the dependency management system is wired into the RACE architecture and its data structures to achieve maximum performance.

Analyses of dependency-directed backtracking can also be found in [Ginsberg, 1993]. In [Ginsberg, 1993] a related technique, called *dynamic backtracking* is introduced as well.[4]

Semantic Branching

Problems are usually given in conjunctive normal form (CNF). In contrast to syntactic branching, i.e. processing disjunctions in writing order, where redundant search spaces may be repeatedly explored, semantic branching uses a *splitting rule* which replaces the original problem by two smaller subproblems (see also [Freeman, 1995] and [Davis et al., 1962] for the original work concerning propositional logic). Semantic branching is usually supported by various techniques intending to speed up the search.

Boolean Constraint Propagation: A *lookahead algorithm* or *constraint propagator* tries to reduce the size of the open search space. If the satisfiability of a disjunct is not known, it is called *open*. After every expansion step, RACE propagates the truth value of a newly added constraint into all open disjuncts of all unexpanded or-constraints. As a result of this step, or-constraints might become satisfied (i.e. one disjunct is satisfied), deterministic (i.e. exactly one disjunct remains open), or might even clash (no disjunct leads to a completion).

Heuristics-Guided Search: Various *heuristics* are used to select the "next" unexpanded or-constraint and one of its associated disjuncts. Similar to FaCT, RACE employs a dynamic selection scheme. A so-called oldest-first strategy is used for selecting one or-constraint with at least two open disjuncts. The idea behind the oldest-first strategy is to maximize the effect of dependency-directed backtracking.

[4] See also [Ginsberg & McAllester, 1994] for work on dynamic backtracking in stochastic search algorithms.

For every new or-constraint which is generated during the search tree expansion a number is generated. The number represents the age of the or-constraint. Smaller numbers represent older or-constraints, i.e. or-constraints that have been generated early in the search. If a selected or-constraint leads to a clash, then backjumping is performed (see above). The idea of selecting the oldest or-constraints is to skip as much of the search space as possible by performing backjumping.

Once an or-constraint is selected for expansion, one of its open disjuncts must be chosen. In accordance with [Freeman, 1995] the selection heuristics is based on the number of the negated and non-negated occurrences for each open disjunct in *all* other unexpanded or-constraints. These numbers are used as input for a priority function that selects a disjunct by applying a certain weighting scheme. In order to minimize the search space, disjuncts are preferred that have a similar number of negated and non-negated occurrences. Furthermore, those disjuncts from the selected or-constraint are preferred that occur in other or-constraints with a small number of disjuncts. The idea behind the latter strategy is to maximize the effect of boolean constraint propagation (see above). In order to perform the counting required for implementing the priority function very quickly, for every or-constraint RACE precomputes two lists for cross-referencing other or-constraints that contain the or-constraint's disjuncts in negated or non-negated form.

Once a disjunct is selected, the priority function is also used to determine whether the constraint is tried in negated or non-negated form first. In case of a failure, the other alternative is explored. Since semantic branching is combined with dependency-directed backtracking, a specific dependency management is required for disjuncts during backtracking and forward checking.

Semantic branching optimization techniques have also been developed for provers in modal logics (see the so-called "SAT techniques" initially described in [Giunchiglia & Sebastiani, 1996]).

4.2.2 Optimizing TBox Reasoning

Dealing with TBoxes requires specific optimization techniques tailored to the inference services provided.

Lazy Unfolding

The tableaux calculus described above treats axioms of the form $\mathsf{C} \sqsubseteq \mathsf{D}$ by transforming them into universal concept restrictions of the form $\forall\, x\,.\, x : \neg\mathsf{C} \sqcup \mathsf{D}$. The rule R$\forall_x$ is applied to every new individual introduced by the generating rules R$\exists$C and R$\exists_{\geq n}$. This adds additional assertions to *every* subtableau. Even worse, the universal concept restrictions contain disjunctions, i.e. the search space is dramatically extended. Although techniques for efficiently managing the search space are

integrated into modern DL architectures (see above), in the average case it is always advantageous to reduce the number of assertions the prover has to deal with. Therefore, in the $\mathcal{KRIS}$ system a technique called *lazy* unfolding was investigated [Baader et al., 1992; Baader et al., 1994] for the logic $\mathcal{ALCNF}$.[5] The idea is to exploit special forms of GCIs in the TBox. If there is a GCI $\mathsf{A} \sqsubseteq \mathsf{C}$ in the TBox with a concept name A on the left-hand side and (i) there is no other GCI with A on the left-hand side, (ii) there is no GCI $\mathsf{C} \sqsubseteq \mathsf{A}$ in the TBox and (iii) the GCI $\mathsf{A} \sqsubseteq \mathsf{C}$ is not cyclic with respect to the TBox, then the GCI $\mathsf{A} \sqsubseteq \mathsf{C}$ is called a *primitive concept definition* (i.e. a definition with only necessary conditions). If there exists an additional GCI $\mathsf{C} \sqsubseteq \mathsf{A}$, the set $\{\mathsf{A} \sqsubseteq \mathsf{C}, \mathsf{C} \sqsubseteq \mathsf{A}\}$ is called a *concept definition* (i.e. a definition with necessary and sufficient conditions). For a concept definition the abbreviation $\mathsf{A} \doteq \mathsf{C}$ is used. In the following we call GCIs that represent primitive concept definitions *simple* GCIs. Other GCIs are also referred to as *true* GCIs. Concept definitions and primitive concept definition are also called *concept introductions*.

Now, given a TBox where concept definitions and primitive concept definitions are identified, then, by the introduction of additional rules the DL prover "expands" assertions of the form $\mathsf{i}\!:\!\mathsf{A}$ and $\mathsf{i}\!:\!\neg\mathsf{A}$ in a lazy way. Let us discuss the first case. If for A there exists a primitive concept definition $\mathsf{A} \sqsubseteq \mathsf{C}$, the definition of A is inserted into the ABox as an assertion $\mathsf{i}\!:\!\mathsf{C}$. If there exists a concept definition for A, $\mathsf{i}\!:\!\mathsf{A}$ is expanded as well in the same way as for primitive concept definitions. For concept definitions, assertions of the form $\mathsf{i}\!:\!\neg\mathsf{A}$ are treated in a similar way, i.e. the assertion is expanded by inserting $\mathsf{i}\!:\!\neg\mathsf{C}$ into the ABox. However, $\mathsf{i}\!:\!\neg\mathsf{A}$ need not be expanded using the above-mentioned technique if there exists a primitive concept definition.

Empirical tests indicate a significant speed gain if the lazy unfolding technique is implemented in a prover architecture [Baader et al., 1992; Baader et al., 1994].

GCI Transformations

The general idea of this technique is to eliminate true GCIs in a preprocessing phase by transforming them into concept definitions or simple GCIs [Horrocks, 1997]. The GCI transformation is illustrated by the following example. The two GCIs $\{\mathsf{A} \sqsubseteq \mathsf{E} \sqcup \mathsf{F}, \mathsf{A} \sqcap \mathsf{B} \sqsubseteq \mathsf{C} \sqcap \mathsf{D}\}$ are transformed into the single concept inclusion $\mathsf{A} \sqsubseteq (\mathsf{E} \sqcup \mathsf{F}) \sqcap (\neg\mathsf{B} \sqcup (\mathsf{C} \sqcap \mathsf{D}))$. Due to the transformation there remains only one concept inclusion with A on the left-hand side. Thus, we have a simple GCI. Since after the transformation there are fewer GCIs in the equisatisfiable TBox, the old GCIs are called *absorbed GCIs*.

Any non-simple GCI that could not be absorbed has to be represented by a universal concept restriction (see the corresponding rule in tableaux calculus presented in

[5] Probably, similar techniques have been used in other DL systems before $\mathcal{KRIS}$ as well. However, only the work on $\mathcal{KRIS}$ contains a systematic empirical evaluation of the effects.

Section 4.1). These restrictions usually add disjunctions to every subtableau and are a major source of complexity (see the discussion about lazy unfolding). As we have discussed before, for simple GCIs it is not necessary to introduce universal concept restrictions if the lazy unfolding technique is used. In summary, the goal of this compilation process is to keep the structure of the GCIs as "simple as possible" and, furthermore, to keep their number as small as possible. The remaining GCIs are also called *meta constraints* and are dealt with by the rule completion rule R$\forall_x$.

Marking and Propagation Techniques

The RACE architecture incorporates the TBox reasoning algorithms described in [Baader et al., 1992; Baader et al., 1994] for computing the sets of parents and children for each atomic concept used in a TBox. Let us assume there exists a lattice of concept names defined by the parent (or children) relation (see the definition of the inference problems in Section 2.3). Now let us assume that the parents and children of a new concept name are to be computed, i.e. the new concept name is "inserted" into the so-called subsumption lattice. The parents for a new concept name CN are computed in a so-called *top-search phase*. Starting from the top of the lattice, the so-called $\top$ concept, the subsumption relation between CN and the concept names already found in the lattice is determined. The search proceeds downwards until the most-specific concept names found in the lattice that subsume CN are identified (for details see [Baader et al., 1992; Baader et al., 1994]). The children of CN are identified in a analogous way in a so-called *bottom-search phase* proceeding from the bottom ($\bot$) of the lattice towards the top.

The construction of the subsumption lattice starts with an initial lattice consisting only of $\top$ and $\bot$. Then, all concept names are inserted one after another. In the worst case, constructing the lattice requires $O(n^2)$ subsumption tests where n is the number of atomic concepts. Therefore, marking and propagation techniques based on intermediate computation results in the subsumption lattice are analyzed in [Baader et al., 1992; Baader et al., 1994]. The main idea is to avoid "expensive" subsumption tests with full tableaux proofs by exploiting "obvious" information as far as possible (see also the work on KL-ONE and $\mathcal{KRIS}$ described in Chapter 3).

Flat Model Merging

The *model merging strategy* tries to avoid a consistency test for a conjunction of concepts, which relies on the "expensive" tableaux technique (see above). A model merging test is designed to be a "cheap" test operating on cached "concept models." It is a *sound* but incomplete consistency tester for a set of concepts. Minimal computational overhead and the avoidance of any nondeterminism are important characteristics of such a test. If a model merging test returns false, a sound and

complete tableaux calculus is applied. In order to be more precise, we use the term *pseudo model* instead of "concept model." A model is understood in the sense of an interpretation and a pseudo model as a data structure containing recorded information.

Model merging is particularly important for implementing TBox inference services. The test whether a concept C subsumes a concept D is preceded or may be even replaced by a merging test for so-called pseudo models of $\neg C$ and D. Pseudo models are data structures that capture the constraints represented by a completion used to prove the consistency of a concept term. If the constraints (pseudo models) found in the completions of two consistent concept terms do not interact, the conjunction of the concepts is consistent as well [Horrocks, 1997] (see below for a detailed definition). The idea is to store the constraints in such a way that the test whether the constraints interact can be executed in a very fast way.

If the pseudo models are mergable, C does not subsume D. This technique was first realized in the FaCT system for $\mathcal{ALCH}f_{R^+}$. RACE extends this technique for $\mathcal{ALCNH}_{R^+}$. The model merging test in RACE correctly deals with number restrictions, i.e. the model merging test is realized as an incomplete structural consistency test for the language $\mathcal{ALENH}$ ($\mathcal{ALE}$ plus number restrictions and role hierarchies). If non-$\mathcal{ALENH}$ language constructs are encountered, the answer of the structural subsumption test is "do not know." In this case the full sound and complete $\mathcal{ALCNH}_{R^+}$ tableaux calculus is used. In order to avoid redundancy, details are explained below in the context of proposed extensions.

4.2.3 ABox Partitioning and Multiple TBoxes or ABoxes

Only very few publications consider optimizations for ABox reasoning. A notable exception is [Hollunder, 1994]. In the context of the $\mathcal{KRIS}$ system it is suggested to automatically partition an ABox into independent subparts. This scheme of autopartitioning is supported by RACE. RACE offers additional optimizations which will be discussed in the next sections. As the $\mathcal{KRIS}$ system [Baader & Hollunder, 1991b; Baader & Hollunder, 1991a], RACE also supports the declaration of multiple ABoxes w.r.t. a certain TBox. TBox inferences are only required once. A TBox can serve as the "background knowledge" employed for specific inference problems about "data" (ABoxes). Similar to the $\mathcal{KRIS}$ system but in contrast to the FaCT system, RACE also supports multiple TBoxes, i.e. TBoxes are objects that can be stored in data structures etc.

4.3 Demanding Problems for Testing DL Systems

Although much has been achieved in recent years, there are many interesting application problems that could not be solved with existing DL technology before the development of RACE. With the new and extended optimization techniques described in this chapter RACE can cope with "complex" knowledge bases developed in the area of system verification and medical ontologies. In the former case the complexity is due to cyclic axioms whereas in the latter case it is the large number of axioms and concept names that contributes to complexity. We briefly introduce different variants of knowledge bases from both areas which, at the time of their development, could only be processed with the RACE system within appropriate time limits. Afterwards, a set of quasi-standard benchmark problems is discussed in order to compare RACE with other DL systems when extended and new optimization techniques implemented in RACE are introduced.

4.3.1 TBox for Specification Verification in Telecommunication Systems

The first application of DL technology uses the consistency inference problem in order to verify the specification of telecommunication systems. Since telecommunication systems offer many features which become more and more complex, it is often not apparent whether different features can be effectively combined. Features can interact in unwanted and unforeseen ways. If an unwanted feature interaction occurs in a telecommunication system which is already installed at the site of a customer, usually high reconfiguration costs are the consequence. Thus, checking the formal specification of a system before actually installing it can be essential. Although some progress has been made in this area, approaches where interactions are described in a formal logic and where interaction detection is formulated as a reasoning task, are relatively rare [Areces et al., 1999]. Automatically generated formal proofs, derived with verified provers can raise the confidence in a telecommunication system specification. Empirical studies have shown that most of the interactions occurring in telecommunication systems can be detected by considering systems with up to 7 users.

In [Areces et al., 1999] an approach for detecting feature interactions using description logics is presented. As a description logic $\mathcal{ALC}$ with TBoxes containing cyclic GCIs is used. In the following the main idea behind the encoding of the behavior of a phone system with a description logic TBox is briefly reviewed. The behavior of a phone system is modeled with a state space graph (see Figure 4.1). In a state space graph, vertices denote states, and edges represent actions. States are described by concept names, actions are represented by roles. Let us assume that a phone system is in a certain state. The state is an instance of one or more concepts. For the concepts, a set of axioms is defined. The axioms describe possible actions of the users

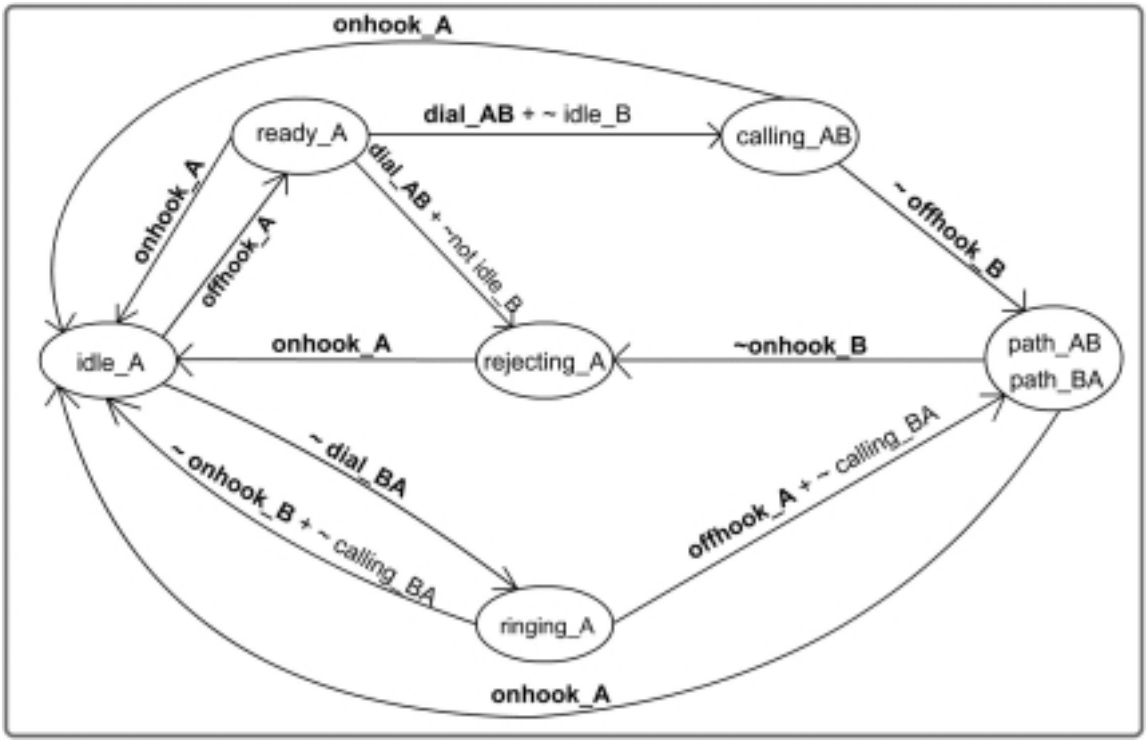


Figure 4.1: States (vertices) and actions (edges) of a telecommunication system with two subscribers (the symbol $\sim$ in front of an action means that the successor vertex denotes a system state with constraints for another subscriber).

(also called subscribers). The basic idea is to use exists restrictions ($\exists\, \mathsf{R}\,.\,\mathsf{C}$) on the right-hand side of the axioms. The role R of an exist restriction used for that purpose denotes the action of a certain subscriber. The concept C of the exists restriction models the state of the phone system after the action. Thus, a TBox is used to model the state-space graph of a so-called basic call system with n subscribers.

In Figure 4.1 a part of the state space of the telecommunication system is presented. Only two subscribers A and B are considered. As can be seen in Figure 4.1 the graph is cyclic. Hence, the TBox for describing the state space contains (indirectly) cyclic axioms.

In the approach described in [Areces et al., 1999], features of telephones such as 'call forwarding unconditional' (CFU) or 'terminating call screening' (i.e. rejecting calls, TCS) are modeled with a set of additional axioms for each feature and for each subscriber.

The specification of the basic call system is faulty if one of the concept names used in the initial TBox (called BCS) is inconsistent. Now, if certain features are activated

for certain phones, additional axioms are added. Even though the BCS might be coherent, adding additional axioms for activated features might result in an inconsistency to occur. In this case a feature interaction is detected. This might be the case when, for instance, a subscriber B redirects calls to another subscriber C in order to receive phone calls there, but is not aware that the subscriber C rejects calls from subscriber A. In this case the subscriber B unintentionally rejects calls from A.

In a BCS TBox (and possibly a TBox with additions for activated features) a very large state space is encoded for even few subscribers and, hence, checking the coherence of such a cyclic TBox is a hard problem for description logic inference systems. Due to the combinatorics adding another subscriber results in an exponential increase in runtime. For evaluating the performance of RACE, BCS-TBoxes for three to five subscribers are considered. The TBoxes are called 'BCS3', 'BCS4' and 'BCS5', respectively. The evaluation results are presented below.

The BCS problems can be considered as representatives for similar specification verification problems from other domains. Hence, solving the TBox coherence problem for BCS (and its extensions with certain features) demonstrates the relevance of DL inference technology for modern software engineering and verification problems and shows the practical usefulness of the RACE system in this context. For details of the BCS system formalization see [Areces et al., 1999].

4.3.2 Ontology Engineering in Bio-Informatics: UMLS-TBox

For web information systems, thesaurus-based information retrieval has proven to be a very effective means for providing answers that are better focused to the interests of users. Recently, a discipline called *ontology engineering* has emerged from various research fields in artificial intelligence and database theory.[6] The main idea of ontology engineering is to augment thesaurus-based but semi-structured representations with inheritance-based inference techniques.

As an example we consider a "reconstruction" of important parts of the UMLS (Unified Medical Language System [McCray & Nelson, 1995]). The reconstruction employs the DL $\mathcal{ALCNH}$ with cyclic axioms and is described in [Schulz & Hahn, 2000] and introduces a specific scheme that uses several concept names to represent subset as well as composition aspects of each word mentioned in the UMLS thesaurus. A TBox is (more or less) automatically generated from specifications in the UMLS thesaurus [Schulz & Hahn, 2000]. For instance, for the notion of a 'heart', the following axioms for heart structures (suffix 's'), heart parts (suffix 'p') and heart entities (no suffix) are declared (see [Schulz & Hahn, 2000] for details):

[6] For ontology engineering an organization has been founded, see `www.ontology.org`.

$$
\begin{aligned}
\textbf{ana\_heart} \sqsubseteq\ & \mathsf{ana\_heart\_s} \sqcap \\
& \mathsf{ana\_hollow\_viscus} \sqcap \\
& \mathsf{umls\_body\_part\_organ\_or\_organ\_component} \\
\textbf{ana\_heart\_s} \sqsubseteq\ & \mathsf{ana\_hollow\_viscus\_s} \sqcap \\
& \mathsf{ana\_cardiovascular\_system\_p} \\
\textbf{ana\_heart\_p} \sqsubseteq\ & \neg \mathsf{ana\_heart} \sqcap \\
& \mathsf{ana\_heart\_s} \sqcap \\
& \exists_{\geq 1}\ \mathsf{anatomical\_part\_of\_ana\_heart}
\end{aligned}
$$

Note the disjointness declaration between ana_heart_p and ana_heart. The following role axiom is generated as well.

$$\textbf{anatomical\_part\_of\_ana\_heart} \sqsubseteq \mathsf{anatomical\_part\_of\_ana\_hollow\_viscus}$$

It is beyond the scope of this chapter to discuss the pros and cons of specific modeling techniques used in the UMLS reconstruction.

In many application projects that are comparable to the UMLS example it is necessary to deal with TBoxes with a large number of axioms. In addition, in many applications only a small subset of the axioms are true generalized concept inclusions (GCIs). In most cases, axioms are concept introduction axioms. Usually it has been argued that only systems based on incomplete calculi can deal with knowledge bases with more than 100,000 axioms of this kind. With the empirical analysis of the performance of the description logic system RACE it is shown that description logic systems based on sound and complete algorithms are particularly useful for simple but large knowledge bases consisting mainly of primitive concept definitions. A knowledge base is called *simple* if no meta constraints remain after the absorption phase (see Section 4.2.2) and there exist (almost) no defined concepts.

The performance of the RACE system is evaluated with different versions of the UMLS knowledge base. UMLS-1 is a preliminary version that contains many inconsistent concept names. It consists of approximately 100,000 concept names and for almost all of them there exists a primitive concept definition $\mathsf{A} \sqsubseteq \mathsf{C}$ with C not being $\top$. In addition, in UMLS-1 80,000 role names are declared. Role names are arranged in a hierarchy. UMLS-2 is a new version in which the reasons for the inconsistencies have been removed. The version of UMLS-2 that is used for the empirical tests contains approximately 160,000 concept names. Furthermore, about 80,000 roles are declared.

Originally, the UMLS knowledge base has been developed with Loom 4.0 [MacGregor, 1994]. If Loom encounters a cyclic definition for a certain concept, then Loom does not classify this concept (and the concepts which use this concept). Due to Loom's treatment of cycles, the concepts are placed in a so-called `:implies` clause, i.e. these restrictions are only asserted for individuals in an ABox via the rule mechanism. For the same reason, the UMLS reconstruction uses `:implies` for domain and range restrictions for roles, i.e. domain and range restrictions are only asserted in the ABox.

With RACE, none of these pragmatic distinctions are necessary. However, in order to mimic the Loom behavior, for each of the knowledge base versions, UMLS-1 and UMLS-2, three different subversions were generated (indicated with letters **a**, **b** and **c**). Version **a** uses axioms of the style presented above, i.e. the `:implies` parts are omitted for TBox classification (and coherence checking). In version **b** the `:implies` part of the Loom knowledge base is indeed considered for classification by RACE. Thus, additional axioms of the following form are generated and added to the TBox.

$$\mathsf{ana\_heart} \sqsubseteq \exists\, \mathsf{has\_developmental\_fo}\,.\,\mathsf{ana\_fetal\_heart} \sqcap \exists\, \mathsf{surrounded\_by}\,.\,\mathsf{ana\_pericardium}$$

Version **c** is the hardest version. Additional axioms provide domain and range restrictions for roles. For example, for anatomical_part_of_ana_heart the following axioms are generated.

$$\exists\, \mathsf{anatomical\_part\_of\_ana\_heart}\,.\,\top \sqsubseteq \mathsf{ana\_heart\_p}$$
$$\top \sqsubseteq \forall\, \mathsf{anatomical\_part\_of\_ana\_heart}\,.\,\mathsf{ana\_heart}$$

Axioms of this kind cannot be absorbed by the GCI transformation algorithm used in FaCT. Thus, since there are 80,000 roles used in the UMLS-2c TBox, a DL system that does not deal with domain and range restrictions in a special way will instantly run into combinatorial explosion if GCIs of this kind are treated with the universal concept restriction rule R$\forall_x$ (see Section 4.1).

In summary, for the performance evaluation of RACE we have tested 6 different knowledge bases. Currently, RACE is the only DL system based on sound and complete inference algorithms that can cope with very large TBoxes such as those generated in the UMLS reconstruction approach. Details about the performance analysis of different optimization techniques are presented below.

4.3.3 Testing Methodology

Most of the tests for evaluating optimizations in RACE use CPU time as a measure of performance. Some researchers have argued that performance measurements

should not consider CPU time but rather "abstract" operations involved in an algorithm. Many of these abstract operations can be recorded in the RACE system and are available for debugging purposes (e.g. semantic splits for semantic branching, number of model merging tests, number of subsumption tests with the tableaux calculus, number of retracted models etc.). However, presenting internal details would result in a discussion which would be far too fine-grained to be of general interest in this context. Furthermore, many of the operations are specific to a certain implementation of a tableaux calculus (e.g. constraint system sizes). Therefore, for the evaluation results presented in this chapter absolute runtime is used as a very general indicator for the effect of certain optimization technology. Moreover, since many of the benchmarks consist of a sequence of problems each of which requires an exponentially increasing runtime, for instance, doubling the mere processing power results in at most one additional problem to be solved. With this kind of setting, only new algorithms and data structures can help in achieving a speed gain of an order of magnitude.

For the application problems mentioned in the two previous subsections, it is only important that they can be solved in a "reasonable amount of time", i.e. within a few hours. As we will see, at the time of this writing, RACE is the only system that can deal with these kinds of applications of DL technology. The effectiveness of optimization techniques is also evaluated with 11 additional TBoxes developed in other application projects. However, with modern systems, classification times range from a few seconds to a few minutes. These knowledge bases have been used in several system competitions and they are used to compare RACE with other systems. The 'Galen' TBox was already used in [Horrocks, 1997; Horrocks, 1998; Horrocks & Patel-Schneider, 1999]. Three different versions of 'Galen' use different DLs ranging from $\mathcal{ALE}$ to $\mathcal{ALCH}f_{R^+}$. The TBoxes 'ESPR', 'WISBER', 'CKB', and 'FSS' are enhanced versions of the TBoxes used in the DL'98 system comparison [Horrocks & Patel-Schneider, 1998a]. We restored role hierarchies and domain and/or range restrictions for primitive roles using GCIs [Haarslev & Möller, 1999b]. A set of nine new TBoxes 'Bike1' to 'Bike9' represent configuration knowledge about various types of bicycles using $\mathcal{ALCNH}$ with GCIs. 'Bike1' to 'Bike9' evaluate the efficiency of the implementation of the number restriction exists rule and the number restriction merge rule. For the Bike-TBoxes, corresponding ABox benchmarks are also available [Haarslev & Möller, 1999b].

In addition to various TBoxes derived from applications, a set of synthetic concept consistency and TBox benchmark problems is used to compare RACE with other systems. This set of benchmark has also been used for evaluating the $\mathcal{KRIS}$ system [Baader et al., 1994] and the FaCT system [Horrocks, 1997]. Furthermore, a new set of synthetic ABox benchmarks has been developed in order to measure the performance of the RACE system concerning ABox consistency testing [Haarslev &

Möller, 1999b]. Each synthetic benchmark contains 21 problem sets (or levels) that are constructed in such a way that runtimes increase exponentially from level to level if no specific optimization techniques are employed. Thus, using a faster machine should result in at most one additional level to be solved.

4.4 Extended and New Optimizations in RACE

Many new optimization techniques have been developed for the RACE system. Some of them are extensions or adaptations of previously published techniques. This section gives an overview about the utility of different techniques using empirical investigations.

4.4.1 Optimizations for Concept Consistency and TBox Reasoning

In this section we discuss TBox optimization techniques that are either novel or have been implemented and evaluated for the first time.

New Transformations on GCIs

Based on the results in [Horrocks, 1997], RACE transforms GCIs into a form such that lazy unfolding can be more effectively exploited. A formal analysis of GCI transformation techniques has been presented in [Horrocks & Tobies, 2000]. RACE used a slightly extended scheme for transforming GCIs.

Domain and range restrictions for roles might be given implicitly as GCIs. These GCIs are removed by RACE since domain and range restrictions are specially treated in the RACE architecture (see below). Furthermore, GCIs of the form $\top \sqsubseteq \exists_{\leq 0} \mathsf{R}$ (or $\top \sqsubseteq \forall \mathsf{R} \,.\, \bot$) are removed as well. Instead, the role R and all its subroles are declared as features. Features are handled in a special way in accordance with [Horrocks, 1997][p. 74f.].

Other transformations are motivated by a heuristic which tries to maximally simplify GCIs and to absorb GCIs (elements of a set G) into a set D ('concept definitions' and 'primitive concept definitions') whose elements are specially treated in RACE by the lazy unfolding technique (see above). Let us assume that all concepts are in negation normal form and the terms 'concept definition' and 'primitive concept definition' are defined as above. The transformation scheme starts with the sets D, G initialized as follows.

$$D := \{\mathsf{A} \sqsubseteq \mathsf{C} \mid \mathsf{A} \sqsubseteq \mathsf{C} \text{ is a primitive concept definition in } \mathcal{T}\}$$
$$G := \{\mathsf{C} \sqsubseteq \mathsf{D} \in \mathcal{T}\} \setminus D$$

The GCIs in G are iteratively transformed or removed from G and added to D. The process stops and returns G and D if no GCI transformation rule is applicable anymore. The process employs the transformation rules as given in [Horrocks, 1997] with the additional rule application steps roughly sketched below.

1. If two GCIs $\mathsf{A} \sqsubseteq \mathsf{C}$, $\mathsf{C} \sqsubseteq \mathsf{A}$ are in G such that $\{\mathsf{A} \sqsubseteq \mathsf{C}, \mathsf{C} \sqsubseteq \mathsf{A}\}$ is a concept definition in $D \cup G$, the two GCIs $\mathsf{A} \sqsubseteq \mathsf{C}$, $\mathsf{C} \sqsubseteq \mathsf{A}$ are removed from G and the term $\mathsf{A} \doteq \mathsf{C}$ is added to D.

2. If there exists a concept definition $\mathsf{C} \doteq \mathsf{D}_1 \sqcap \ldots \sqcap \mathsf{D}_n$ in D and $\mathsf{D}_i \sqsubseteq \mathsf{E} \in G$, then remove $\mathsf{C} \doteq \mathsf{D}_1 \sqcap \ldots \sqcap \mathsf{D}_n$ from D and add $\mathsf{C} \sqsubseteq \mathsf{D}_1 \sqcap \ldots \sqcap \mathsf{D}_n$ to D and $\mathsf{D}_1 \sqcap \ldots \sqcap \mathsf{D}_n \sqsubseteq \mathsf{C}$ to G. The condition $\mathsf{D}_i \sqsubseteq \mathsf{E} \in G$ ensures that the transformation is only performed if there is an inclusion axiom for one of the D_i concepts.

3. If there exists a GCI $\mathsf{C} \sqsubseteq \mathsf{D}_1 \sqcap \ldots \sqcap \mathsf{D}_n$ in G and there exists an i such that D_i is a negated concept[7], remove the GCI $\mathsf{C} \sqsubseteq \mathsf{D}_1 \sqcap \ldots \sqcap \mathsf{D}_n$ from G and add the GCIs $\{\mathsf{C} \sqsubseteq \mathsf{D}_1, \ldots, \mathsf{C} \sqsubseteq \mathsf{D}_n\}$ to G. Let us assume there exist a $\mathsf{D}_i = \neg\mathsf{A}$. Then, other transformation rules subsequently transform the GCI $\mathsf{C} \sqsubseteq \neg\mathsf{A}$ into $\mathsf{A} \sqsubseteq \neg\mathsf{C}$. If $\mathsf{A} \sqsubseteq \neg\mathsf{C}$ is a primitive concept definition in $D \cup G$ or can be absorbed such that a primitive concept definition remains, then the GCI $\mathsf{C} \sqsubseteq \mathsf{D}_1 \sqcap \ldots \sqcap \mathsf{D}_n$ can be more effectively treated. Furthermore, it might also be possible to absorb $\mathsf{C} \sqsubseteq \mathsf{D}_j$ ($j \neq i$) using other transformation rules.

4. Furthermore, if there are two GCIs $\mathsf{C}_1 \sqsubseteq \mathsf{D}$ and $\mathsf{C}_2 \sqsubseteq \mathsf{D}$ in G, then replace these GCIs by one GCI of the form $\mathsf{C}_1 \sqcap \mathsf{C}_2 \sqsubseteq \mathsf{D}$. This transformation might also be implemented in the FaCT system. It represents an application of the distributive law. Due to our experiences, for some optimization techniques mentioning D twice might lead to problems with the heuristics used to select disjuncts (see the section on heuristics guided search in Section 4.2.1).

Note that, in contrast to FaCT, RACE also supports absorption of GCIs into inclusions $\neg\mathsf{A} \sqsubseteq \mathsf{C}_1$ (but only if no inclusion $\mathsf{A} \sqsubseteq \mathsf{C}_2$ or definition $\mathsf{A} \doteq \mathsf{C}_2$ exists). Some knowledge bases can only be handled effectively with $\neg\mathsf{A} \sqsubseteq \mathsf{C}_1$ absorptions.

Empirical investigations of the new GCI transformation techniques are presented below.

[7] Actually, as a concept is required to be in NNF, the concept in the scope of a negation must be a concept name.

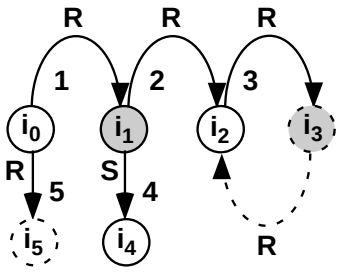
Caching in RACE

Caching of intermediate computation results is a necessary prerequisite to prove the (in)consistency of many concept terms [Horrocks & Patel-Schneider, 1999]. In tableaux calculi for testing concept consistency, sets of concepts representing a conjunction are manipulated. These concept sets are also called "labels". For instance, let us assume a concept set L contains the set $\{\exists \mathsf{R}.\mathsf{C}, \forall \mathsf{R}.\mathsf{D}, \forall \mathsf{R}.\mathsf{E}\}$ as a subset and there are no other all-concepts for the role R in L. The tableaux rule for treating some-concepts identifies corresponding all-concepts and checks if a completion of subtableau $\{\mathsf{C}, \mathsf{D}, \mathsf{E}\}$ can be found. The concepts in the original concept set L are not relevant for this test. It can be shown that even for $\mathcal{ALCNH}_{R^+}$ this trace technique can be applied for consistency testing (see Section 4.2.1). In other words, concept sets resulting from some-concepts (and corresponding all-concepts) can be independently tested for consistency because there is no interaction with the original concept set which contains the some- and all-concepts.

Once the solution for the consistency problem for a set of concepts has been computed, the answer is stored in a cache. The key of the cache is the set of concepts being considered.[8] The cache value indicates whether the conjunction of the key is satisfiable or not. Now, given such a cache, an "expensive" tableaux proof can be avoided if a "cheap" cache lookup indicates whether a set of concepts has already been proven to be consistent or inconsistent.

For *concept consistency*, the cache technique described above is sketched in [Horrocks & Patel-Schneider, 1999]. However, in an implementation of a tableaux calculus for deciding *ABox consistency*, the technique cannot be directly applied since it might be possible that the new individual which is generated by applying the rule for treating some-assertions must possibly be identified with an old individual, i.e. additional assertions are possibly imposed. Furthermore, if we consider the logic $\mathcal{ALCNH}_{R^+}$, number restrictions can also require the identification of individuals. However, when there are neither role assertions nor at-most restrictions imposed for the corresponding role, the above-mentioned situations can be ruled out and the trace technique becomes applicable. The assertions for the new individual do not interact with assertions in the original ABox, and the consistency of the new assertions can be tested by checking whether the new "partition" (a fresh ABox with just the concept constraints for the new individual) is consistent. Since there is only one individual and no role assertions, the set of concepts of the corresponding concept assertions can be checked against a cache, i.e. the same subtableaux caching technique as used in implementations for concept consistency algorithms can be employed.

[8] In many cases, sets are implemented as lists. Thus, in order to preserve the set semantics when using the key for cache retrieval, the list has to be sorted such that the elements in the list are in a canonical order.



1. $\{i_0 : E\}$
2. $\{i_0 : E, i_0 : \exists R \,.\, C\} \vee \{i_0 : E, i_0 : \exists R \,.\, D\}$
3. $\{i_1 : C\}$
4. $\{i_1 : C, i_1 : \exists R \,.\, D, i_1 : \exists S \,.\, X, i_1 : \forall S \,.\, \neg X \sqcap A\}$
5. $\{i_2 : D\}$
6. $\{i_2 : D, i_2 : \exists R \,.\, C\}$
7. $\{i_3 : C\}$
8. $\{i_4 : X, i_4 : A, i_4 : \neg X\}$
9. $\{i_5 : D\}$

Figure 4.2: Caching example with blocking (see text).

Although memory consumption might lead to problems and memory management strategies might be necessary, empirical tests indicate that the subtableaux caching technique is very effective for many ABox consistency problems (see below for a discussion of the effects when subtableaux caching is disabled). However, there are some benchmark problems where memory management techniques are indeed required in order to preserve an upper bound on memory consumption. Subtableaux caching has been first investigated with DLP. Only recently, FaCT has been extended with this technique. No details have been published, though.

Solving a consistency problem $\{i_0 : E\}$ w.r.t. the following (cyclic) inclusion axioms demonstrates that caching must depend on the "blocking context". If the blocking context is not considered carefully, invalid models might be entered into the cache. This is shown with an example. Let us consider the following TBox:

$$\mathbf{C} \sqsubseteq (\exists R \,.\, D) \sqcap (\exists S \,.\, X) \sqcap \forall S \,.\, (\neg X \sqcap A)$$
$$\mathbf{D} \sqsubseteq \exists R \,.\, C$$
$$\mathbf{E} \sqsubseteq (\exists R \,.\, C) \sqcup (\exists R \,.\, D)$$

Furthermore, the consistency of the concept E w.r.t. the TBox is to be tested. The proof steps are presented in Figure 4.2. Although the RACE system employs various heuristics for choosing the next assertion to be expanded, for presentation purposes we assume that the constituents of a conjunction or disjunction are considered in writing order. The effect described below would also occur when specific selection strategies are exploited but the presentation would be much more complex. In the

figure the sequence of "expansion" steps is indicated with numbers. Furthermore, the lazy unfolding strategy for dealing with inclusion axioms is used (there are no meta constraints).

In step 1, the initial problem $\{\mathsf{i}_0 : \mathsf{E}\}$ is presented. Since there is an axiom for E involving a disjunction we get two constraint systems (see line 2). Let us assume the first alternative is tried first. This leads to a subconstraint system (number 3). The assertion from step 3 is expanded w.r.t. the axioms and we get the constraint system in step 4. The first some-constraint is expanded first. In step 5 the corresponding subconstraint system is considered. The right-hand side of the axiom for D is inserted (step 6). The some-constraint yields another subconstraint system (step 7). Due to the blocking strategy (see Definition 22), the constraint system in step 7 is not expanded (see the constraint system in step 3 with the blocking individual i_1). Hence, the assertion $\mathsf{i}_2 : \exists\, \mathsf{R}\,.\,\mathsf{C}$ from step 6 is assumed to be satisfiable. In Figure 4.2 the corresponding interpretation is indicated with a dashed arrow.

An often-employed strategy is to cache intermediate results, i.e. the satisfiability of D is stored as a pseudo model $\{\mathsf{D}, \exists\, \mathsf{R}\,.\,\mathsf{C}\}$ (see below for details). Let us assume that at the end of step 7 a pseudo model for D is stored as indicated above. In our example, there are some proof steps pending. In step 8 the remaining constraints from step 5 are considered. Obviously, the second some-constraint for the role S together with the value restriction for S causes a clash. Therefore, the second alternative in step 2 has to be considered. The corresponding subconstraint system is presented as step 9. If the consistency of D is checked by examining a cache entry for D, the overall result will be "E is consistent". Obviously, this is erroneous. The reason is that the caching principle described above does not consider the dependency on the satisfiability of C. Therefore, a dependency tracking mechanism for cache entries is implemented in RACE. Once the system detects the inconsistency of a concept (or constraint system) on which a cached pseudo model is dependent, the corresponding cache entries are (recursively) invalidated.

If, during a certain tableaux proof for the consistency of a concept term, a some-constraint (e.g. $\exists\, \mathsf{R}\,.\,\mathsf{C}$ interacts with all-constraints $\forall\, \mathsf{R}\,.\,\mathsf{D}$ and $\forall\, \mathsf{R}\,.\,\mathsf{E}$), then another strategy is to cache the result of checking the consistency of the subproblem $\{\mathsf{i}_{\mathsf{new}} : \mathsf{C}, \mathsf{i}_{\mathsf{new}} : \mathsf{D}, \mathsf{i}_{\mathsf{new}} : \mathsf{E}\}$. Again, the dependency tracking mechanism implemented in RACE ensures correct behavior of the subtableaux caching strategy in the case of blocking.

RACE supports different caching policies. A cache for finding information about (sorted) sets of concepts is used for checking whether a set of concepts is satisfiable or unsatisfiable (so-called *equal cache* implemented as a hash table). Furthermore, a pair of caches for satisfiable as well as unsatisfiable concept sets is provided. These caches support queries concerning already encountered supersets and subsets of a given set of concepts, respectively. If the equal cache is enabled, it is the first reference. Only if a

cache lookup fails, the superset or the subset caches are consulted. If the equal cache is enabled, all retrieval results from the superset of subset caches are also entered into the equal cache. The data structures for the subset/superset caches are inspired by [Hoffmann & Köhler, 1999]. The default mode of RACE uses subset/superset caching only. In the context of DL systems, subset/superset caching has first been investigated with the RACE system (see also [Giunchiglia & Tacchella, 2000] for an investigation of this technique for modal logic provers that do not require blocking).

Summary of Empirical Results for the BCS TBoxes

In Table 4.1 the runtimes for the example problems BCS3 to BCS5 are shown. The tests have been performed with Macintosh Common Lisp on a 400MHz PowerMac in a partition of about 90MByte. Without subtableaux caching, none of the problems can be solved in a reasonable amount of time. For the tests in Table 4.1 equal caching is always enabled. The tests indicate that with subset/superset caching also enabled (setting 1) a speed gain can be achieved. However, runtimes do not increase dramatically if subset/superset caching is disabled (setting 2). The settings 3 and 4 correspond to setting 1 and 2, respectively, but the enhanced GCI transformation techniques are disabled. Execution times for knowledge bases in which feature interactions concerning CFU and TCS are detected require comparable runtimes.

The system FaCT has been extended with subtableaux caching only recently (only equal caching and no subset/superset caching) [Horrocks, 2000]. No runtimes for the BCS problems are available. Details are about caching in FaCT have not been published. The initial version of FaCT could not cope with BCS5. The DLP system does not support GCI absorption and, therefore, cannot deal with the BCS problems, either. As we have seen, RACE can handle specifications for systems with up to 5 users. So, some progress has been made to support the practical verification of system specifications with automatic theorem provers. However, dealing with systems of 6 or even 7 users is currently beyond reach (cf. Section 4.3.1).

Table 4.1: BCS TBox classification tests with caching (times in seconds).

KB	**(1)**	**(2)**	**(3)**	**(4)**
BCS3	4.8	5.7	18.5	19.5
BCS4	105	146	180	203
BCS5	47460	51175	57030	83251

Exploiting Deep Models for TBox Reasoning

In Section 4.2.2 we have discussed that a subsumption test or a test to check whether a conjunction of concepts is satisfiable can be replace by a so-called flat-model merging test. In the following we present and analyze a technique called *deep model merging* that generalizes the original model merging approach [Horrocks, 1997] in two ways. (1) We extend the model merging technique to the logic $\mathcal{ALCNH}_{R^+}$. (2) We introduce *deep* pseudo models which are *recursively* traversed and checked for possible clashes. To the best of our knowledge this is the first formal treatment showing the soundness of model merging.

Definition 26 (Pseudo Model) A pseudo model for a concept term C is defined as follows. Let $\mathsf{A} \in C$ be a concept name, $\mathsf{R} \in R$ a role name, $\mathsf{F} \in F$ a feature name. If C is inconsistent, the *pseudo model* of C is defined as $\bot$. If C is consistent, then there exists a set of completions $\mathcal{C}$ for the ABox $\mathcal{A} = \{\mathsf{a}:\mathsf{C}\}$. A completion $\mathcal{A}' \in \mathcal{C}$ is selected and a pseudo model M for a concept C is defined as the tuple $\langle M^{\mathsf{A}}, M^{\neg\mathsf{A}}, M^{\exists}, M^{\forall}\rangle$ of concept sets using the following definitions:

$$M^{\mathsf{A}} = \{\mathsf{A} \mid \mathsf{a}:\mathsf{A} \in \mathcal{A}'\}, \quad M^{\neg\mathsf{A}} = \{\mathsf{A} \mid \mathsf{a}:\neg\mathsf{A} \in \mathcal{A}'\}$$
$$M^{\exists} = \{\exists\mathsf{R}.\mathsf{C} \mid \mathsf{a}:\exists\mathsf{R}.\mathsf{C} \in \mathcal{A}'\} \cup \{\exists_{\geq n}\mathsf{R} \mid \mathsf{a}:\exists_{\geq n}\mathsf{R} \in \mathcal{A}'\}$$
$$M^{\forall} = \{\forall\mathsf{R}.\mathsf{C} \mid \mathsf{a}:\forall\mathsf{R}.\mathsf{C} \in \mathcal{A}'\} \cup \{\exists_{\leq n}\mathsf{R} \mid \mathsf{a}:\exists_{\leq n}\mathsf{R} \in \mathcal{A}'\} \cup \{\exists\mathsf{F}.\mathsf{C} \mid \mathsf{a}:\exists\mathsf{F}.\mathsf{C} \in \mathcal{A}'\}$$

For brevity a pseudo model is called a *pmodel*. Note that the set $M^{\exists}$ contains all exists- and at-least concepts while $M^{\forall}$ contains all at-most- and all-concepts as well as all exists-concepts for features. This guarantees the correct treatment of features.

The procedure **mergable** shown in Procedure 1 implements the flat and deep model merging test. In case of deep merging it has to test for a blocking situation, i.e. whether the actual pmodel set *MS* is a member of the set *VM* of visited pmodel sets. The initial call of mergable has the empty set as value for *VM*. The third parameter *D*? controls whether the deep or flat mode (see below) of mergable will be used.

We assume a procedure **get_pmodel** that retrieves for a concept C its cached pmodel. In case the pmodel does not yet exist, it is computed.

The procedure **atoms_mergable** tests for a possible primitive clash between pairs of pmodels. It is applied to a set of pmodels *MS* and returns *false* if there exist $\{M_1, M_2\} \subseteq MS$ with $(M_1^{\mathsf{A}} \cap M_2^{\neg\mathsf{A}}) \neq \emptyset$ or $(M_1^{\neg\mathsf{A}} \cap M_2^{\mathsf{A}}) \neq \emptyset$. Otherwise it returns *true*.

The procedure **critical_at_most** tests for a potential number restriction clash in a set of pmodels and tries to avoid *true* answers which are too conservative. It is

Procedure 1 mergable(MS, VM, $D?$)

```
 1: if MS = ∅ ∨ MS ∈ VM then
 2:    return true
 3: else if ⊥ ∈ MS ∨ ¬atoms_mergable(MS) then
 4:    return false
 5: else
 6:    for all M ∈ MS do
 7:       for all C ∈ M^∃ do
 8:          if critical_at_most(C, M, MS) then
 9:             return false
10:          else
11:             MS' ← collect_pmodels(C, MS)
12:             if (¬D? ∧ MS' ≠ ∅) ∨ ¬mergable(MS', VM ∪ {MS}, D?) then
13:                return false
14:             end if
15:          end if
16:       end for
17:    end for
18: end if
19: return true
```

applied to a concept C of the form $\exists\,\mathsf{S}\,.\,\mathsf{D}$ or $\exists_{\geq n}\,\mathsf{S}$, a pmodel M (the current model) and a set of pmodels $MS = \{M_1, \ldots, M_k\}$ and returns *true* if there exists a pmodel $M' \in (MS \setminus M)$ and a role $\mathsf{R} \in \mathsf{S}^{\uparrow}$ with $\exists_{\leq m}\,\mathsf{R} \in M'^{\forall}$ such that $\sum_{\mathsf{E}\in N} \mathrm{num}_{RS}(\mathsf{E}) > m$, $N = \cup_{i\in 1..k}\, M_i^{\exists}$, $RS = \mathsf{S}^{\uparrow} \cap \mathsf{R}^{\downarrow}$. In all other cases critical_at_most returns *false*. The procedure $\mathrm{num}_{RS}(\mathsf{E})$ returns 1 for concepts of the form $\mathsf{E} = \exists\,\mathsf{R}'\,.\,\mathsf{D}$ and n for $\mathsf{E} = \exists_{\geq n}\,\mathsf{R}'$, provided $\mathsf{R}' \in RS$.

The procedure **collect_pmodels** is applied to a concept C of the form $\exists\,\mathsf{S}\,.\,\mathsf{D}$ or $\exists_{\geq n}\,\mathsf{S}$ and a set of pseudo models MS. It computes the pmodels of the set Q of "qualifications". We define $Q' = \{\mathsf{D}\}$ if $\mathsf{C} = \exists\,\mathsf{S}\,.\,\mathsf{D}$ and $Q' = \emptyset$ otherwise.

$$Q = Q' \cup \{\mathsf{E} \mid \exists\, M \in MS, \mathsf{R} \in \mathsf{S}^{\uparrow} : (\forall\,\mathsf{R}\,.\,\mathsf{E} \in M^{\forall} \vee \exists\,\mathsf{R}\,.\,\mathsf{E} \in M^{\forall})\} \cup$$
$$\{\forall\,\mathsf{T}\,.\,\mathsf{E} \mid \exists\, M \in MS, \mathsf{R} \in \mathsf{S}^{\uparrow}, \mathsf{T} \in T \cap \mathsf{S}^{\uparrow} \cap \mathsf{R}^{\downarrow} : \forall\,\mathsf{R}\,.\,\mathsf{E} \in M^{\forall}\}$$

The procedure collect_pmodels returns the set $\{\mathrm{get\_pmodel}(\mathsf{C}) \mid \mathsf{C} \in Q\}$. Observe that $\exists\,\mathsf{R}\,.\,\mathsf{E} \in M^{\forall}$ implies that R is a feature.

In the following we prove the soundness of the procedure mergable for the description logic $\mathcal{ALCN}\mathcal{H}_{R^+}$. Note that mergable depends on the clash triggers of the particular tableaux calculus chosen since it has to detect potential clashes in a set of pmodels.

Proposition 27 (Soundness of mergable) Let $D?$ have either the value *true* or *false*, $CS = \{\mathsf{C}_1, \ldots, \mathsf{C}_n\}$, $M_{C_i} = \mathrm{get\_pmodel}(\mathsf{C}_i)$, and $PM = \{M_{C_i} \mid \mathsf{i} \in 1..\mathsf{n}\}$. If the

procedure call mergable($PM, \emptyset, D?$) returns *true*, the concept $\mathsf{C_1} \sqcap \ldots \sqcap \mathsf{C_n}$ is consistent.

Proof. This is proven by contradiction and induction. Let us assume that the call mergable($PM, \emptyset, D?$) returns *true* but the ABox $\mathcal{A} = \{\mathsf{a}:(\mathsf{C_1} \sqcap \ldots \sqcap \mathsf{C_n})\}$ is inconsistent, i.e. there exists no completion of $\mathcal{A}$. Every concept $\mathsf{C_i}$ must be satisfiable, otherwise we would have $\bot \in PM$ and mergable would return *false* due to line 3 in Procedure 1. Let us assume a finite set $\mathcal{C}$ containing all contradictory ABoxes encountered during the consistency test of $\mathcal{A}$. Without loss of generality we can select an arbitrary $\mathcal{A}' \in \mathcal{C}$ and make a case analysis of its possible clash culprits.

1. We have a primitive clash for the "root" individual a, i.e. $\{\mathsf{a}:\mathsf{D}, \mathsf{a}:\neg\mathsf{D}\} \subseteq \mathcal{A}'$. Thus, $\mathsf{a}:\mathsf{D}$ and $\mathsf{a}:\neg\mathsf{D}$ have not been propagated to a via role assertions and there have to exist $\mathsf{C_i}, \mathsf{C_j} \in CS$, $\mathsf{i} \neq \mathsf{j}$ such that $\mathsf{a}:\mathsf{D}$ ($\mathsf{a}:\neg\mathsf{D}$) is derived from $\mathsf{a}:\mathsf{C_i}$ ($\mathsf{a}:\mathsf{C_j}$) due to the satisfiability of the concepts $\mathsf{C_i}$, $\mathsf{i} \in \mathsf{1..n}$. It holds for the associated pmodels $M_{C_i}, M_{C_j} \in PM$ that $\mathsf{D} \in M_{C_i}^{\mathsf{A}} \cap M_{C_j}^{\neg\mathsf{A}}$. However, due to our assumption the call of mergable($PM, \emptyset, D?$) returned *true*. This is a contradiction since mergable called atoms_mergable with PM (line 3 in Procedure 1) which returned *false* since $\mathsf{D} \in M_{C_i}^{\mathsf{A}} \cap M_{C_j}^{\neg\mathsf{A}}$.
2. A number restriction clash in $\mathcal{A}'$ is detected for a, i.e. $\mathsf{a}:\exists_{\leq m}\,\mathsf{R} \in \mathcal{A}'$ and there exist $l > m$ distinct R-successors of a.[9] These successors can only be derived from assertions of the form $\mathsf{a}:\exists\,\mathsf{S_j}\,.\,\mathsf{E_j}$ or $\mathsf{a}:\exists_{\geq n_j}\,\mathsf{S_j}$ with $\mathsf{S_j} \in \mathsf{R}^{\downarrow}$, $\mathsf{j} \in \mathsf{1..k_1}$. The concepts $\mathsf{C_i} \in CS$, $\mathsf{i} \in \mathsf{1..n}$ are satisfiable and there has to exist a subset $CS' = \{\mathsf{C_{i_1}}, \ldots, \mathsf{C_{i_{k_2}}}\} \subseteq CS$ such that $\exists_{\leq m}\,\mathsf{R} \in \cup_{C_i \in CS'} M_{C_i}^{\forall}$ and $\sum_{\mathsf{E'} \in N} \mathrm{num}_{RS}(\mathsf{E'})$ $\geq l$, $N = \cup_{C_i \in CS'} M_{C_i}^{\exists}$, $RS = (\cup_{\mathsf{j} \in \mathsf{1..k_1}}\, \mathsf{S_j}^{\uparrow}) \cap \mathsf{R}^{\downarrow}$. However, due to our assumption the call of mergable($PM, \emptyset, D?$) returned *true*. This is a contradiction since there exists an $\mathsf{i'} \in \mathsf{1..k_2}$ and a concept $\mathsf{E'} \in M_{C_{i'}}^{\exists}$ such that mergable called critical_at_most($\mathsf{E'}, M_{C_{i'}}, PM$) (lines 6-8 in Procedure 1) which returned *true* since $\sum_{\mathsf{E'} \in N} \mathrm{num}_{RS}(\mathsf{E'}) \geq l > m$.
3. Let the individual $\mathsf{a_n}$ be a successor of $\mathsf{a_0}$ via a chain of role assertions $(\mathsf{a_0}, \mathsf{a_1}):\mathsf{R_1}, \ldots, (\mathsf{a_{n-1}}, \mathsf{a_n}):\mathsf{R_n}$, $\mathsf{n} > \mathsf{0}$ and we now assume that a clash for $\mathsf{a_n}$ is discovered.
   (a) In case of a primitive clash we have $\{\mathsf{a_n}:\mathsf{D}, \mathsf{a_n}:\neg\mathsf{D}\} \subseteq \mathcal{A}'$. These clash culprits are derived from assertions for $\mathsf{a_{n-1}}$ of the form $\mathsf{a_{n-1}}:\exists_{\geq m}\,\mathsf{R_n}$ or $\mathsf{a_{n-1}}:\exists\,\mathsf{R_n}\,.\,\mathsf{E_1}$, and $\mathsf{a_{n-1}}:\forall\,\mathsf{S}\,.\,\mathsf{E_2}$ and/or $\mathsf{a_{n-1}}:\forall\,\mathsf{S'}\,.\,\mathsf{E_3}$ with $\mathsf{S}, \mathsf{S'} \in \mathsf{R_n}^{\uparrow}$. Due to the clash there exists a pair $\mathsf{E_i}, \mathsf{E_j}$ with $\mathsf{D} \in M_{E_i}^{\mathsf{A}} \cap M_{E_j}^{\neg\mathsf{A}}$ for some $\mathsf{i}, \mathsf{j} \in \mathsf{1..3}$, $\mathsf{i} \neq \mathsf{j}$. Each role assertion in the chain between $\mathsf{a_0}$ and $\mathsf{a_{n-1}}$ can only be derived from an assertion of the form $\mathsf{a_{k-1}}:\exists\,\mathsf{R_k}\,.\,\mathsf{E_k}$ or $\mathsf{a_{k-1}}:\exists_{\geq m_k}\,\mathsf{R_k}$ with $\mathsf{k} \in \mathsf{1..n-1}$. The call graph of mergable($PM, \emptyset, D?$) contains a chain of calls resembling the chain of role assertions. By induction on the call

[9] Due to our syntax restriction all elements of $\mathsf{R}^{\downarrow}$ are not transitive.

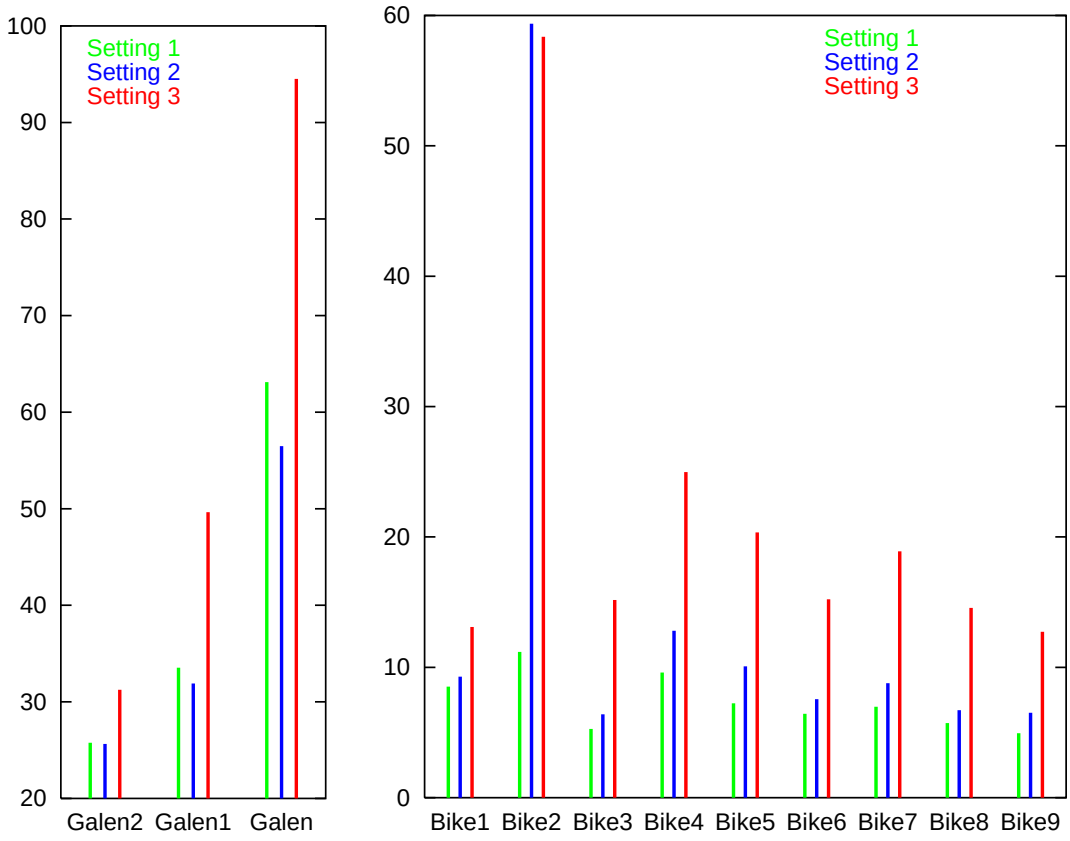
graph we know that the node resembling $\mathsf{a_{n-1}}$ of this call graph chain contains the call mergable(PM', VM', *true*) such that $\{M_{E_i}, M_{E_j}\} \subseteq PM'$ and atoms_mergable has been called with a set MS' and $\{M_{E_i}, M_{E_j}\} \subseteq MS'$. The call of atoms_mergable has returned *false* since $\mathsf{D} \in M_{E_i}^{\mathsf{A}} \cap M_{E_j}^{\neg\mathsf{A}}$. This contradicts our assumption that mergable($PM, \emptyset, D?$) returned *true*.

(b) In case of a number restriction clash we can argue in an analogous way. Again, we have a chain of role assertions where a number restriction clash is detected for the last individual of the chain. It exists a corresponding call graph chain where by induction the last call of mergable called critical_at_most with a set of pmodels for which critical_at_most returned *true*. This contradicts the assumption that mergable($PM, \emptyset, D?$) returned *true*.

It is easy to see that this proof also holds in the case the value of $D?$ is *false* since the "flat mode" is more conservative than the "deep" one, i.e. it will always return *false* instead of possibly *true* if the set of collected pmodels M' is not empty (line 12 in Procedure 1) □

The advantage of the deep vs. the flat mode of the model merging technique is demonstrated by empirical tests using a set of "quasi-standard" application TBoxes/ABoxes [Horrocks & Patel-Schneider, 1998a; Horrocks & Patel-Schneider, 1999; Haarslev & Möller, 1999b]. Figure 4.3a-b shows the runtimes for computing the subsumption lattice of these TBoxes. Each TBox is iteratively classified using 3 different parameter settings. The first setting has all optimization techniques enabled, in the second one the subtableaux caching technique (see above) is disabled. The third setting has both subtableaux caching and the deep mode of model merging disabled but the flat mode of model merging is still enabled. The 3 different settings are justified by the order in which these optimization techniques are applied if a subtableau is tested for consistency in RACE. First, subtableaux caching is applied. If no cache entry exists, (deep) model merging is tried. If it returns *false* the standard tableaux test is invoked. Thus, tableaux caching might reduce the number of encountered model merging tests and the advantage of the deep against the flat mode of model merging can only be accurately evaluated if one compares the runtimes between the second and third setting. The comparison between these settings indicates a speed gain in runtimes of a factor $1.5 - 2$ for almost all TBoxes if the deep mode is enabled. The comparison between the first and second setting clearly demonstrates that the deep mode can sometimes compensate the disabled subtableaux caching technique.

The empirical results for the application TBoxes (see Figure 4.3) indicate a speed gain of up to one order of magnitude if the above-mentioned additional GCI transformation rules are applied (compare the runtimes of 'all 3 techniques enabled' vs. 'no enh. gci transformation'). There is no speed gain for the 'Galen' TBoxes since 'Galen2' and 'Galen1' do not contain any GCIs and the procedure from [Horrocks,



(a) Galen TBoxes

(b) Bike TBoxes

Figure 4.3: Evaluation of model merging techniques (3 runs for each TBox, left-right order corresponds to top-bottom order in the legend).

1997] already absorbs all GCIs in 'Galen.'

Dealing with Domain and Range Restrictions

Domain and range restrictions for roles are often-used modeling constructs in applications. A domain restriction C for a role R can be expressed with the GCI $\exists\,\mathsf{R}\,.\,\top \sqsubseteq \mathsf{C}$. Whenever an individual i is set into relation to another object via a role R it is implied that the individual i is an instance of the concept C. A range restriction C for a role R can also be expressed with a GCI. $\top \sqsubseteq \forall\,\mathsf{R}\,.\,\mathsf{C}$ specifies that all individuals are instances of the concept $\forall\,\mathsf{R}\,.\,\mathsf{C}$.

In order to avoid disjunctions, RACE deals with GCIs for domain restrictions with a generalized kind of lazy unfolding. In a similar way as for names, all situations

where unfolding of concept terms $\exists\,\mathsf{R}\,.\,\mathsf{D}$ w.r.t. axioms of the form $\exists\,\mathsf{R}\,.\,\top \sqsubseteq \mathsf{C}$ must occur can be easily identified, i.e. unfolding of a domain restriction for a role R is applied whenever a constraint $\mathsf{i}:\exists\,\mathsf{R}\,.\,\mathsf{C}$ for an arbitrary $\mathcal{ALCNH}_{R^+}$ concept term is encountered in an ABox (see above for the $\mathcal{ALCNH}_{R^+}$ tableaux calculus).[10]

Although it is possible to absorb a domain restriction such as the one expressed by $\exists\,\mathsf{anatomical\_part\_of\_ana\_heart}\,.\,\top \sqsubseteq \mathsf{ana\_heart\_p}$ into an equivalent inclusion axiom $\neg\mathsf{ana\_heart\_p} \sqsubseteq \forall\,\mathsf{anatomical\_part\_of\_ana\_heart}\,.\,\bot$, lazy unfolding cannot be easily applied if an inclusion axiom for $\mathsf{ana\_heart\_p} \sqsubseteq \ldots$ exists. However, this is the case for UMLS. Note that, in principle, RACE also supports absorption of GCIs into inclusions $\neg\mathsf{A} \sqsubseteq \mathsf{C_1}$ (but only if no inclusion $\mathsf{A} \sqsubseteq \mathsf{C_2}$ or definition $\mathsf{A} \doteq \mathsf{C_2}$ exists – see above). Some knowledge bases can only be handled effectively with $\neg\mathsf{A} \sqsubseteq \mathsf{C_1}$ absorptions.

In contrast to domain restrictions, range restrictions for roles do not introduce disjunctions. However, in a practical implementation it is advantageous to keep the number of internal data structures to be managed as small as possible. Therefore, range restrictions $\forall\,\mathsf{R}\,.\,\mathsf{C}$ are "considered" only if an existential restriction for R or a subrole of R are imposed for a certain individual i. These cases can also be easily detected.

Summary of Empirical Results for Standard TBoxes

In Table 4.2 the runtimes for classifying realistic TBoxes from the DL'98 benchmarks with quite many terminological axioms are listed (see the column named "Num. of concepts", for details see also [Horrocks & Patel-Schneider, 1998a]). For some knowledge bases, different version have been tested (the extensions "roles" and "GCIs" indicates that domain and range restrictions for roles are ignored or appropriately represented by GCIs, respectively. The tests have been carried out for all systems, RACE, FaCT and $\mathcal{KRIS}$ using the same test environment as indicated above (Macintosh Common Lisp). We used version 1.11 of FaCT which does not support number restrictions (FaCT can handle $\mathcal{ALCH}f_{R^+}$ TBoxes). The new version of FaCT which supports qualified number restrictions does not apply model merging, so runtimes could not be compared. Furthermore, $\mathcal{KRIS}$ does not support GCIs, so some tests are left out (indicated with NA). The columns RACE and

[10]Implementation note: Due to our experiences, domain restrictions cannot be easily considered in the (recursive) encoding process for concepts. Encoding a concept term `(some r d)` as $\mathsf{C} \sqcap \exists\,\mathsf{R}\,.\,\mathsf{D}$ (with C being the domain restriction for R) would cause all kinds of trouble concerning the negation `(not (some r d))` of this term. The negation of this term would still be $\forall\,\mathsf{R}\,.\,\neg\mathsf{D}$ and not $\neg\mathsf{C} \sqcup \forall\,\mathsf{R}\,.\,\neg\mathsf{D}$ as suggested when `(some r d)` were encoded as $\mathsf{C} \sqcap \exists\,\mathsf{R}\,.\,\mathsf{D}$. So, a special treatment is necessary for these terms. But what if $\mathsf{C} \sqcap \exists\,\mathsf{R}\,.\,\mathsf{D}$ happens to be a concept term used in the knowledge base itself? Then, the negation definitely would be $\neg\mathsf{C} \sqcup \forall\,\mathsf{R}\,.\,\neg\mathsf{D}$. In this case, the encoding procedure can hardly guarantee uniqueness of the encoding result (which is essential for clash detection).

RACE **d** indicate the runtimes of RACE without deep model merging and with deep model merging enabled, respectively.

Table 4.2: Realistic TBox classification tests (times in seconds).

KB	**Num. of concepts**	**Time**			
		RACE	RACE **d**	FaCT	$\mathcal{KRIS}$
ckb-roles	79	0.13	0.15	NA	0.28
ckb GCIs	79	13.70	13.94	NA	NA
datamont-roles	120	0.38	0.21	NA	0.53
espr-roles	142	0.19	0.22	NA	0.41
espr GCIs	142	12.74	12.53	NA	NA
fss-roles	132	0.34	0.50	NA	0.67
fss GCIs	132	23.25	23.55	NA	NA
wisber-roles	140	0.35	0.36	NA	0.69
wisber GCIs	140	4.81	4.43	NA	NA
galen GCIs	2748	125.95	74.57	97.18	NA
galen1	2728	56.84	38.88	43.07	>2000
galen2	3926	32.39	27.80	25.93	189.97

Topological Sorting for Achieving Quasi Definition Order

For TBox classification the RACE system employs the marking and propagation techniques introduced in [Baader et al., 1994]. We have discussed these techniques in Section 4.2.2. For large knowledge bases it is particularly important to avoid as many traversals as possible. Let us assume, a TBox to be classified can be transformed such that no meta constraints but maybe cyclic (primitive) concept definitions exist. Then, if concepts are classified in a so-called 'definition order', the bottom search phase can be omitted for concept names for which only a primitive concept definition exists [Baader et al., 1994]. According to [Baader et al., 1994] we assume that a concept name A 'directly uses' a concept name B if B occurs in the concept on the right-hand side of the definition of A.[11] The relation 'uses' is the transitive closure of 'directly uses'. If A uses B then A precedes B in the definition order. For acyclic TBoxes (i.e. the uses relation is irreflexive) with concept introduction axioms only, the set of concepts can be processed in definition order, i.e. a concept is not classified until all of the concepts used in its definition are classified. In this case the set of children of a concept name consists only of the bottom concept.

[11] If lazy unfolding of domain restrictions is applied, domain restriction have to considered in a special way w.r.t. the 'directly uses' relation.

Thus, a common syntactical restriction for description logic systems is to accept only TBox declarations that do not include so-called forward references. However, for a language such as $\mathcal{ALCNH}_{R^+}$, which offers cyclic axioms and GCIs, in general, the bottom search phase cannot be skipped [Horrocks, 1997, page 103].

Unfortunately, in the UMLS examples there are many forward references involved in value restrictions and existential restrictions (i.e. modalities). Thus, the definition order of concept names has to be computed in a preprocessing step. In addition, a slightly less strict notion of definition order has been developed. We assume a relation 'directly refers-to' similar to 'directly uses' but without considering references occurring in the scope of quantifiers. Again 'uses' is the transitive closure of 'directly refers-to'. For acyclic concepts the 'refers-to' relation induces a partial order relation on concept names. All concept names involved in a cycle are treated as one node (i.e. a set S_i) w.r.t. the partial order. Using a topological sorting algorithm the partial order can be serialized such that a total order between concept names (or sets of concept names) is defined. We call the serialization a "quasi definition order".

Topological sorting is of order $n + e$ where e is the number of given 'refers-to' relationships. Thus, we have approximately $\mathrm{O}(n \log n)$ steps while the bottom search procedure requires $\mathrm{O}(n^2)$ steps in the worst case. Note that in [Baader et al., 1994] no experiments are discussed that involve the computation of a serialization given a TBox with axioms not already in (strict) definition order.

During classification of a TBox with RACE the concept names are processed in the order given by the linearization w.r.t. topological sorting. For each atomic concept A that is not a member of a set S_i and for which no inclusion $\neg\mathsf{A} \sqsubseteq \mathsf{C}$ is available (see Section 4.4.1), the bottom search can be omitted. The 'refers-to' relation and the quasi definition order serialization ensures that either all concepts that are potential subconcepts of a certain primitive concept A are inserted after A has been inserted into the subsumption lattice or the bottom search is indeed performed. The quasi definition order is conservative w.r.t. the potential subsumers (note that $\mathcal{ALCNH}_{R^+}$ does not support inverse roles). Moreover, in a basic subsumption test the subsumption lattice under construction is never referred to. Thus, strict definition order classification is not necessary.

In order to effectively apply the topological sorting optimization, incorporating domain restrictions into the tableaux calculus was a necessary prerequisite because meta constraints must not exist for topological sorting to be a valid optimization.

Clustering

A problem with large knowledge bases is that the set of children of some concept names can get very large. Thus, the top-search procedures exhibits worst case performance, i.e. the optimization techniques presented in [Baader et al., 1994] are not

effective. Therefore, in the implementation of RACE a special clustering technique is employed.

If more than θ concept names are found to be children of a certain concept name, the θ children are grouped into a so-called bucket $\mathsf{A}_{\mathsf{new}}$, i.e. a (virtual) concept definition $\mathsf{A}_{\mathsf{new}} \doteq \mathsf{A}_1 \sqcup \ldots \sqcup \mathsf{A}_\theta$ is assumed and $\mathsf{A}_{\mathsf{new}}$ is virtually inserted into the subsumption lattice with $\mathsf{A}_1 \ldots \mathsf{A}_\theta$ being its children. Note that bucket concepts $\mathsf{A}_{\mathsf{new}}$ are virtual concepts in the sense that they are not mentioned in the set of children or parents of the concept names mentioned in a TBox.

Let us assume, a certain concept name A is inserted. Instead of testing whether each A_i ($\mathsf{i} \in \{1..\theta\}$) subsumes A during the top-search phase, our findings suggest that it is more effective to initially test whether $\mathsf{A}_{\mathsf{new}}$ does not subsume A using the model merging technique. For the subsumption test, only the pmodel of $\neg\mathsf{A}_{\mathsf{new}}$ is computed. If a subsumption relation indeed exists, then clustering introduces some overhead. But, since in most cases, no subsumption relation can be found between any A_i and A, with clustering one model merging test possibly replaces θ single model merging tests. For almost all concept names only primitive concept definitions are included in the TBox. Therefore, the pmodel of $\neg\mathsf{A}_{\mathsf{new}}$ being used for model merging is very "simple". The pmodel basically consists only of a set of negated concept names (see Section 4.4.1 for further details about model merging in RACE). Performing a single model merging test with the pmodel for $\neg\mathsf{A}_{\mathsf{new}}$ causes less overhead than performing θ steps testing subsumption with each of the pmodels for $\neg\mathsf{A}_\mathsf{i}$. If a new parent is computed for concept names involved in a bucket, the bucket structure must be recomputed.

For best performance, the number of concepts to be kept in a bucket depends on the number of subconcepts of the concept. However, the number of subconcepts of a concept can hardly be estimated. Therefore, the following strategy is used. If more and more concept names are "inserted" into the subsumption lattice, the number of buckets increases as well. If a new bucket is to be created for a certain concept A and there are already σ buckets clustering the subconcepts of A, then two buckets (those buckets with the smallest number of children) are merged. Merging of buckets $\mathsf{A}_{\mathsf{new}} \doteq \mathsf{A}_1 \sqcup \ldots \sqcup \mathsf{A}_\mathsf{n}$ and $\mathsf{B}_{\mathsf{new}} \doteq \mathsf{B}_1 \sqcup \ldots \sqcup \mathsf{B}_\mathsf{m}$ means that the bucket $\mathsf{A}_{\mathsf{new}}$ is "redefined" as $\mathsf{A}_{\mathsf{new}} \doteq \mathsf{A}_1 \sqcup \ldots \sqcup \mathsf{A}_\mathsf{n} \sqcup \mathsf{B}_1 \sqcup \ldots \sqcup \mathsf{B}_\mathsf{m}$ and the bucket $\mathsf{B}_{\mathsf{new}}$ is reused for the new bucket to be created (see above).[12] Whether hierarchical clustering techniques lead to performance improvements is subject to further research.

The current evaluation of clustering with buckets uses a setting with $\theta = 10$ and $\sigma = 15$.

[12]Note that due to subsequent merging operations, n and m need not be equal to θ.

Exploiting Disjointness Declarations

As has been discussed in [Baader et al., 1994], it is important to derive told subsumers for each concept name for marking and propagation processes. Besides told subsumers, RACE exploits also the set of "told disjoint concepts". In the 'heart' example presented in Section 4.3.2, ana_heart is computed as a told disjoint concept of ana_heart_p by examining inclusion axioms. If it is known that a concept B is a subsumer of a concept A then A cannot be a subsumee of the told disjoints of B. This kind of information is recorded (and propagated) with appropriate non-subsumer marks (see [Baader et al., 1994] for details about marking and propagation operations) such that this information is not rediscovered with a model merging or even a tableaux-based subsumption test. Exploiting disjointness information has not been investigated in [Baader et al., 1994].

Traversing the subsumption lattice is also needed for ABox realization. The idea is to exploit disjointness information to speed up the realization process as follows. Whenever an instance checking test i : A returns 'yes', it is obvious that i cannot be an instance of a concept that is a member of the set of told disjoint concepts of A. Thus, in the subsumption lattice, the told disjoint concepts are marked accordingly and an instance checking test for these concepts, which possibly involves an "expensive" ABox consistency test, is not necessary. Since a large number of instance checking tests must be performed, the exploitation of disjointness information is particularly effective for ABox realization (see below for experimental results).

Summary of Empirical Results for the UMLS TBoxes

In Section 4.3.2 the basic idea of the encoding of the UMLS thesaurus with a TBox has been introduced. Two major versions (1 and 2) and corresponding minor versions (a-c) have been investigated. UMLS-1 contains 100,000 concept names and UMLS-2 contains 160,000 names. In both versions approximately 80,000 role names are used. In this section the effectiveness of optimization techniques introduced in the previous sections is evaluated with the UMLS TBoxes. All measurements have been performed on a Sun UltraSPARC 2 with 1.4 GByte main memory and Solaris 2.6. RACE is implemented in ANSI Common Lisp and for the tests Franz Allegro Common Lisp 5.0.1 has been used. The results are as follows.

Without clustering and topological sorting, classifying UMLS-1a can be done in approximately 11 hours (1636 concepts are incoherent). With clustering and topological sorting enabled, only 5.5 hours are necessary to compute the same result for UMLS-1a. The second version UMLS-1b requires 3.6 hours with optimization and 6.1 hours without optimization. The reason for enhanced performance with more constraints is that in this version already 47855 concepts are inconsistent. With domain and range restrictions we found that even 60246 concepts are inconsistent.

Table 4.3: Evaluation of the classifications of the UMLS-2 knowledge bases (T = topological sorting, C = clustering, R = runtime [hours:minutes], NST = number of subsumption tests, NM = number of cached models, MaxNC = maximal number of children, NB = number of buckets).

UMLS	**T**	**C**	**R**	**NST** ($\times 10^6$)	**NM** ($\times 10^3$)	**MaxNC**	**NB**
2a	on	on	10:13	232	251	26,874	3,110
	on	off	25:06	2,341	237	"	NA
	off	on	22:40	1,256	362	"	2,740
	off	off	31:26	2,796	281	"	NA
2b	on	on	10:11	232	251	26,874	3,110
	on	off	24:33	2,341	237	"	NA
	off	on	22:00	1,200	360	"	2,700
	off	off	30:18	2,796	281	"	NA
2c	on	on	14:53	222	255	21,298	3,273
	on	off	40:54	3,723	240	"	NA
	off	on	>120:00	?	?	"	?
	off	off	61:18	5,814	277	"	NA

The computation times with RACE are 3.4 hours with optimization and 8.7 hours without optimization. Up to 500 MBytes of memory are required to compute the classification results. For UMLS-1, checking TBox coherence requires approximately 10 minutes.

The new second version UMLS-2 contains an additional part of the UMLS and, therefore, is harder to deal with. Furthermore, there are no inconsistent concepts, i.e. classification is much harder because there are much more nodes in the subsumption lattice. In UMLS-1, due to the large number of inconsistent concepts, the subsumption lattice is rather small because many concept "disappear" as synonyms of the bottom concept. For UMLS-2, checking TBox coherence requires between 15 and 50 minutes (2a: 16 min, 2b: 19 min, 2c: 51 min).

More detailed performance evaluations of RACE applied to UMLS-2 and TBox classification are presented in Table 4.3. In order to provide a machine-independent evaluation, not only runtimes are given but also the number of subsumption tests as well as other indicators are presented. It should be noted that the tableaux algorithm is needed (only) for computing pseudo models. In other words, all subsumption tests are determined by deep model merging tests.

A comparison of setting 1 (both topological sorting and clustering enabled) and setting 2 (clustering disabled) reveals that clustering is a very effective optimization technique for the UMLS-2 TBoxes. The result for UMLS-2a and setting 3 (topological

sorting disabled and clustering enabled) supports the fact that topological sorting is also very effective. The reason for the extraordinary runtime result for setting 3 and UMLS-2c is not completely clear. Perhaps, it is due to removed buckets. A bucket is removed if a member of this bucket is assigned a new parent (see above). This is very likely if topological sorting is disabled. However, since this effect is not apparent for version 2a and 2b, it might be possible that the effect is due to a bug in the Allegro Common Lisp memory management system. For UMLS-2 up to 800 MBytes are required.

If, in setting 4, both clustering and topological sorting are disabled, runtimes increase as well. Moreover, according to the evaluation results, the second version UMLS-2b does not require more computational resources than UMLS-2a (see the discussion about `:implies` in Section 4.3.2). Only the incorporation of domain and range restrictions cause runtimes to increase. For other benchmark TBoxes (e.g. Galen with approx. 3000 concepts) our results suggest that there is neither overhead imposed by the clustering and topological sorting optimization techniques nor is there a significant gain to be observed.

In summary, the results for the UMLS TBoxes clearly demonstrate that clustering is particularly effective in conjunction with topological sorting establishing a quasi-definition order. The work reported here indicates that sound and complete description logic systems can now effectively deal with some instances of very large knowledge bases.

4.4.2 Optimizations for ABox Reasoning

Now we consider new optimization techniques for ABox reasoning that have been investigated for the first time during the development of RACE. In the following paragraph all inference problems are considered w.r.t. a role box $\mathcal{R}$. The role box $\mathcal{R}$ is omitted for brevity.

Exploiting Flat Models for ABox Reasoning

An ABox is realized through a sequence of instance checking tests. The *realization* of an individual a occurring in an ABox $\mathcal{A}$ w.r.t to a TBox $\mathcal{T}$ computes the direct types of a (w.r.t. $\mathcal{A}$ and $\mathcal{T}$). For instance, in order to compute the direct types of a for a given subsumption lattice of the concepts $\mathsf{D}_1, \ldots, \mathsf{D}_n$, a sequence of ABox consistency tests for $\mathcal{A}_{\mathsf{D}_i} = \mathcal{A} \cup \{\mathsf{a} : \neg \mathsf{D}_i\}$ might be required. However, individuals are usually members of only a small number of concepts and the ABoxes $\mathcal{A}_{\mathsf{D}_i}$ are proven as consistent in most cases. The basic idea is to design a cheap but *sound* model merging test for the focused individual a and the concept terms $\neg \mathsf{D}_i$ without explicitly considering role assertions and concept assertions for the other individuals mentioned

in $\mathcal{A}$ since these interactions are reflected in the "individual pseudo model" of a. This is the motivation for devising the novel *individual model merging* technique.

A pseudo model for an individual a mentioned in a consistent ABox $\mathcal{A}$ w.r.t. a TBox $\mathcal{T}$ is defined as follows. Since $\mathcal{A}$ is consistent, there exists a set of completions $\mathcal{C}$ of $\mathcal{A}$. Let $\mathcal{A}' \in \mathcal{C}$. An *individual pseudo model* M for an individual a in $\mathcal{A}$ is defined as the tuple $\langle M^{\mathsf{A}}, M^{\neg\mathsf{A}}, M^{\exists}, M^{\forall} \rangle$ w.r.t. $\mathcal{A}'$ and $\mathcal{A}$ using the following definitions.

$$M^{\mathsf{A}} = \{\mathsf{A} \,|\, \mathsf{a}{:}\mathsf{A} \in \mathcal{A}'\}, \quad M^{\neg\mathsf{A}} = \{\mathsf{A} \,|\, \mathsf{a}{:}\neg\mathsf{A} \in \mathcal{A}'\}$$
$$M^{\exists} = \{\exists\,\mathsf{R}\,.\,\mathsf{C} \,|\, \mathsf{a}{:}\exists\,\mathsf{R}\,.\,\mathsf{C} \in \mathcal{A}'\} \cup \{\exists_{\geq n}\,\mathsf{R} \,|\, \mathsf{a}{:}\exists_{\geq n}\,\mathsf{R} \in \mathcal{A}'\} \cup \{\exists_{\geq 1}\,\mathsf{R} \,|\, (\mathsf{a},\mathsf{b}){:}\mathsf{R} \in \mathcal{A}\}$$
$$M^{\forall} = \{\forall\,\mathsf{R}\,.\,\mathsf{C} \,|\, \mathsf{a}{:}\forall\,\mathsf{R}\,.\,\mathsf{C} \in \mathcal{A}'\} \cup \{\exists_{\leq n}\,\mathsf{R} \,|\, \mathsf{a}{:}\exists_{\leq n}\,\mathsf{R} \in \mathcal{A}'\} \cup \{\exists\,\mathsf{F}\,.\,\mathsf{C} \,|\, \mathsf{a}{:}\exists\,\mathsf{F}\,.\,\mathsf{C} \in \mathcal{A}'\}$$

The procedure **get_ind_pmodel** called with an individual a mentioned in a consistent ABox $\mathcal{A}$ (w.r.t. a TBox $\mathcal{T}$) either appropriately creates a pmodel for a or retrieves the cached pmodel of a.

Individual model merging is accomplished by applying the procedure mergable to the models computed by *get_ind_pmodel(*a*)* and *get_pmodel(*$\neg\mathsf{C}$*)*. The flat mode is used, i.e. the third parameter is set to false.

Proposition 28 (Soundness of Individual Model Merging) Let a be an individual mentioned in a consistent ABox $\mathcal{A}$ w.r.t. a TBox $\mathcal{T}$, $\neg\mathsf{C}$ be a satisfiable concept, M_a ($M_{\neg C}$) denote the pmodel returned by *get_ind_pmodel(*a*)* (*get_pmodel(*$\neg\mathsf{C}$*)*), and the set PM be defined as $\{M_a, M_{\neg C}\}$. If the procedure call mergable($PM, \emptyset,$ *false*) returns *true*, the ABox $\mathcal{A} \cup \{\mathsf{a}{:}\neg\mathsf{C}\}$ is consistent, i.e. a is not an instance of C.

Proof. This is proven by contradiction. Let us assume that the procedure call mergable($\{M_a, M_{\neg C}\}, \emptyset,$ *false*) returns *true* but the ABox $\mathcal{A}' = \mathcal{A} \cup \{\mathsf{a}{:}\neg\mathsf{C}\}$ is inconsistent, i.e. there exists no completion of $\mathcal{A}'$. Let us assume a finite set $\mathcal{C}$ containing all contradictory ABoxes encountered during the consistency test of $\mathcal{A}'$. Without loss of generality we can select an arbitrary $\mathcal{A}'' \in \mathcal{C}$ and make a case analysis of its possible clash culprits. In the following the same definitions as in Procedure 1 (see page 98) are assumed.

1. A clash is detected for an individual b in $\mathcal{A}''$ that is distinct to a. Since $\mathcal{A}$ is consistent the individual b must be a successor of a via a chain of role assertions $(\mathsf{a},\mathsf{b}_1){:}\mathsf{R}_1, \ldots, (\mathsf{b}_n,\mathsf{b}){:}\mathsf{R}_{n+1}, \mathsf{n} \geq 0$ and one of the clash culprits must be derived from the newly added assertion $\mathsf{a}{:}\neg\mathsf{C}$ and propagated to b via the role assertion chain originating from a with $(\mathsf{a},\mathsf{b}_1){:}\mathsf{R}_1$. Since $\neg\mathsf{C}$ is satisfiable and $\mathcal{A}$ is consistent we have an "interaction" via the role or feature R_1. This implies for the associated pmodels $M_a, M_{\neg C}$ that $(M_a^{\exists} \cap M_{\neg C}^{\forall}) \cup (M_a^{\forall} \cap M_{\neg C}^{\exists}) \neq \emptyset$. This

contradicts the assumption that mergable($\{M_a, M_{\neg C}\}, \emptyset, \mathit{false}$) returned *true* since mergable eventually called collect_pmodels for $M_a, M_{\neg C}$ which returned a non-empty set (line 11 in Procedure 1).

2. In case of a primitive clash for a we have $\{\mathsf{a}:\mathsf{D}, \mathsf{a}:\neg\mathsf{D}\} \subseteq \mathcal{A}''$. Since $\mathsf{a}:\neg\mathsf{C}$ is a concept assertion we know that $\mathsf{a}:\mathsf{D}$ and $\mathsf{a}:\neg\mathsf{D}$ cannot both be propagated to a via role assertions. Thus, either $\mathsf{a}:\mathsf{D}$ or $\mathsf{a}:\neg\mathsf{D}$ must be derived from $\mathsf{a}:\neg\mathsf{C}$ and we have $\mathsf{D} \in (M_a^{\mathsf{A}} \cap M_{\neg C}^{\neg\mathsf{A}}) \cup (M_a^{\neg\mathsf{A}} \cap M_{\neg C}^{\mathsf{A}})$. This contradicts the assumption that mergable($\{M_a, M_{\neg C}\}, \emptyset, \mathit{false}$) returned *true* since mergable called atoms_mergable($\{M_a, M_{\neg C}\}$) which returned *false* (line 3 in Procedure 1) since $\mathsf{D} \in (M_a^{\mathsf{A}} \cap M_{\neg C}^{\neg\mathsf{A}}) \cup (M_a^{\neg\mathsf{A}} \cap M_{\neg C}^{\mathsf{A}})$.
3. A number restriction clash in $\mathcal{A}''$ is detected for a, i.e. $\mathsf{a}:\exists_{\leq m}\mathsf{R} \in \mathcal{A}''$ and there exist $l > m$ distinct R-successors of a in $\mathcal{A}''$. This implies that the set $N = M_a^{\exists} \cup M_{\neg C}^{\exists}$ contains concepts of the form $\exists\,\mathsf{S_j}\,.\,\mathsf{E_j}$ or $\exists_{\geq n_j}\,\mathsf{S_j}$,[13] $\mathsf{S_j} \in \mathsf{R}^{\downarrow}$, $\mathsf{j} \in \mathsf{1..k}$, such that $\sum_{\mathsf{E}' \in N} \mathrm{num}_{RS}(\mathsf{E}') \geq l$, $RS = (\cup_{\mathsf{j}\in\mathsf{1..k}}\,\mathsf{S_j}^{\uparrow}) \cap \mathsf{R}^{\downarrow}$. This contradicts the assumption that mergable($\{M_a, M_{\neg C}\}, \emptyset, \mathit{false}$) returned *true* since mergable called critical_at_most (lines 6-8 in Procedure 1) which returned *true* since $\sum_{\mathsf{E}' \in N} \mathrm{num}_{RS}(\mathsf{E}') \geq l > m$.

□

The performance gain by the individual model merging technique has been empirically evaluated using a set of five ABoxes containing between 15 and 25 individuals. Each of these ABoxes is realized w.r.t. to the application TBoxes Bike7-9 derived from a bike configuration task. The TBoxes especially vary on the degree of explicit disjointness declarations between atomic concepts. Figure 4.4 shows the runtimes for the realization of the ABoxes 1-5. Each ABox is realized with three different parameter settings. The first setting has all optimization techniques enabled, in the second setting an optimization technique is disabled that considers disjointness between concepts (see Section 4.4.1),[14] and additionally the third setting has the individual model merging technique disabled. The comparison between setting two and three reveals a speed gain of at least one order of magnitude if the individual model merging technique is used. Note the use of a logarithmic scale.

Role Path Contraction

In order to make ABox realization as fast as possible a transformation technique for ABoxes has been developed that maximizes the effect of caching techniques. As we have seen above, caching whether a concept is consistent or not is an important

[13] Any role assertion of the form $(\mathsf{a},\mathsf{b}):\mathsf{R} \in \mathcal{A}$ implies that $\exists_{\geq 1}\,\mathsf{R} \in M_a^{\exists}$.

[14] This technique interacts with individual model merging since it prunes the search space for realization and thus decreases the number of instance checking tests that can be solved by individual model merging.

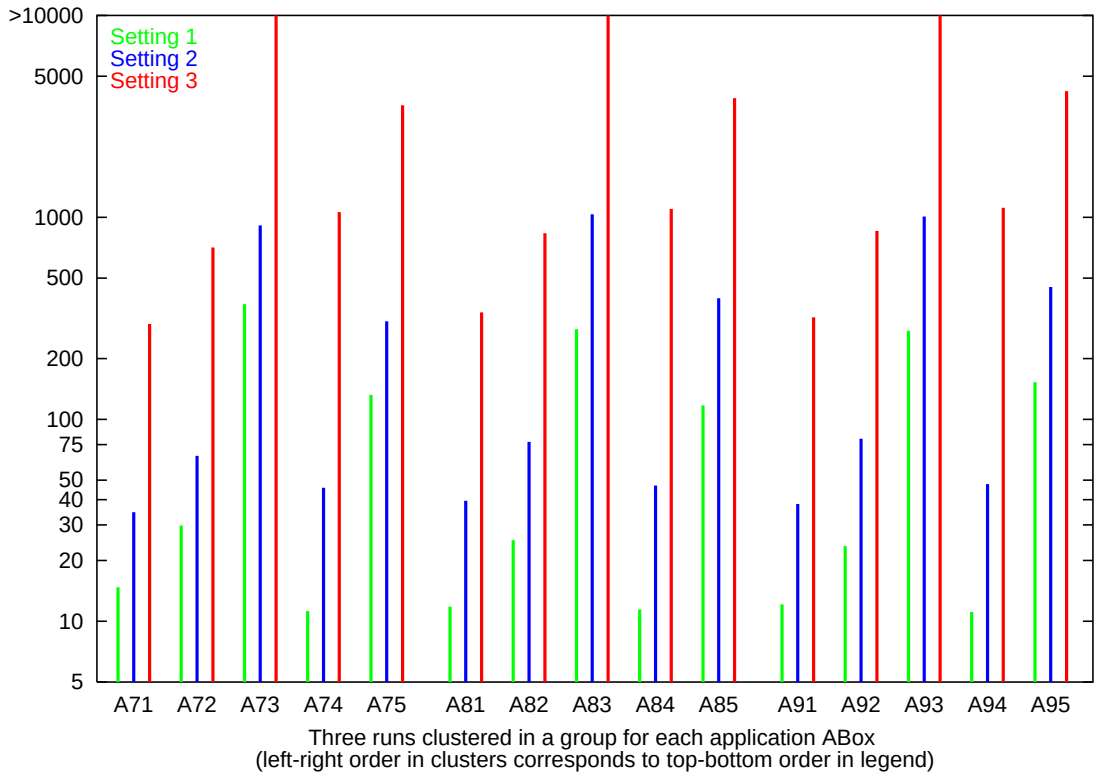


Figure 4.4: Evaluation of model merging techniques for Bike ABoxes (3 runs for each ABox, left-right order corresponds to top-bottom order in the legend).

optimization technique. If ABox consistency for an ABox $\mathcal{A}$ can be decided by transforming $\mathcal{A}$ (or parts of $\mathcal{A}$) such that a concept is computed that is consistent iff the transformed ABox is consistent then the caching techniques developed for the concept consistency problem become applicable also to ABox reasoning. As we will see in the following for some ABoxes or parts thereof a transformation is indeed possible. Empirical investigations indicate that the transformation overhead can be neglected.

Informally, the role path contraction idea is the following. Let us assume, the direct types of a certain individual a have to be computed. We transform the original ABox in such a way that acyclic "role paths" between individuals in "tree-like" ABox structures are represented by an appropriate some-concept. The corresponding concept and role assertions representing the role paths are deleted from the ABox. In order to transform an ABox, the following transformation rule is applied as often as possible (a is the individual being realized).

We illustrate this contraction idea by an example presented in Figure 4.5. The indi-

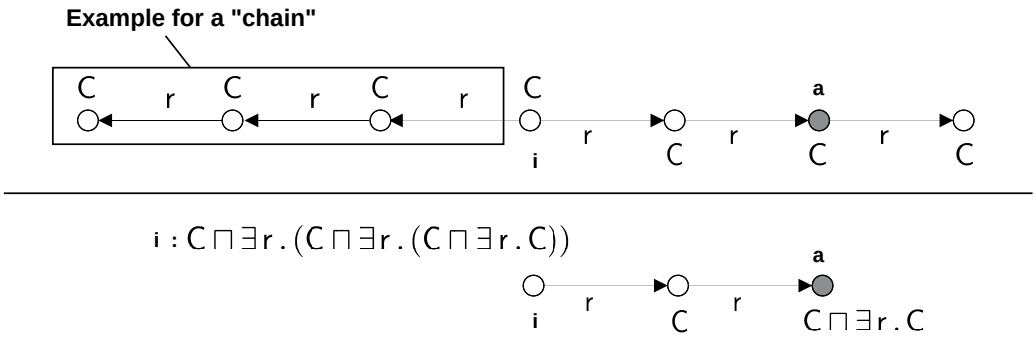


Figure 4.5: Example for ABox chain contraction (see text). The upper ABox is transformed into the lower one.

vidual a is represented by a gray circle. A some-concept "representing" a contracted role path is added as a concept assertion to the individual starting the contracted path.

RC1 Contraction Rule for $\mathcal{ALC}$.
if 1. $(\mathsf{i},\mathsf{j}):\mathsf{R} \in \mathcal{A}, \mathsf{j}:\mathsf{C}_1 \in \mathcal{A}, \ldots, \mathsf{j}:\mathsf{C}_\mathsf{n} \in \mathcal{A}, \mathsf{i} \neq \mathsf{a}$, $\neg\exists\,(\mathsf{j},\mathsf{k}):\mathsf{R}' \in \mathcal{A}$, and
2. $\neg\exists\,(\mathsf{l},\mathsf{j}):\mathsf{R}'' \in \mathcal{A}, \mathsf{l} \neq \mathsf{i}$, $\neg\exists\,\mathsf{j}:\mathsf{C}_{\mathsf{n}+1} \in \mathcal{A} : \forall\,\mathsf{i} \in 1..\mathsf{n} : \mathsf{C}_{\mathsf{n}+1} \neq \mathsf{C}_\mathsf{i}$
then $\mathcal{A}' := (\mathcal{A} \setminus \{(\mathsf{i},\mathsf{j}):\mathsf{R}, \mathsf{j}:\mathsf{C}_1, \ldots, \mathsf{j}:\mathsf{C}_\mathsf{n}\}) \cup \{\mathsf{i}:\exists\,\mathsf{R}\,.\,\mathsf{C}_1 \sqcap \ldots \sqcap \mathsf{C}_\mathsf{n}\}$

We define an algorithm *contraction_alc*($\mathcal{A}$) that iteratively applies the rule **RC1** to a consistent input ABox $\mathcal{A}$ as long as possible. The computed ABox $\mathcal{A}'$ is used as $\mathcal{A}$ in subsequent steps. Finally, when **RC1** is no longer applicable, the algorithm returns the ABox $\mathcal{A}'$.

Proposition 29 The algorithm *contraction_alc* transforms a consistent $\mathcal{ALC}$ ABox $\mathcal{A}$ into an ABox $\mathcal{A}'$ that is consistent iff $\mathcal{A}$ is consistent.

In every step **RC1** removes a role assertion of the form $(\mathsf{i},\mathsf{j}):\mathsf{R}$. In the premise of **RC1** a role assertion is used as a precondition for applying the rule. Thus, if there are n role assertions in the original ABox $\mathcal{A}$, the algorithm *contraction_alc*($\mathcal{A}$) terminates after n steps, at the latest. If the rule **RC1** is applied, the ABox $\mathcal{A}'$ does not contain assertions for the individual j. However, considering the new some-concept assertion for i and the sound and complete rules of the underlying tableaux calculus (see Chapter 2) we can see that the tableaux rules will create a new individual j' with the same concept assertions as for j in the original ABox. The additional assertion $\mathsf{j}':\mathsf{C}_1 \sqcap \ldots \sqcap \mathsf{C}_\mathsf{n}$ is expanded into $\mathsf{j}':\mathsf{C}_1, \ldots, \mathsf{j}':\mathsf{C}_\mathsf{n}$.

The contraction rule **RC1** can be used for $\mathcal{ALC}$ ABoxes (whose associated TBoxes are also $\mathcal{ALC}$ TBoxes). In the presence of number restrictions, the contraction is not applied to an ABox if there exist more than one role filler for a certain role R. The

reason is that individuals explicitly mentioned in the ABox must not be eliminated due to the unique name assumption. If these individuals are "represented" by some-concepts, their uniqueness information is lost. Thus, for $\mathcal{ALCNH}_{R^+}$ ABoxes we need a more conservative contraction rule **RC2**.

RC2 Contraction Rule for $\mathcal{ALCNH}_{R^+}$.
if 1. $(\mathsf{i},\mathsf{j}):\mathsf{R} \in \mathcal{A}, \mathsf{j}:\mathsf{C}_1 \in \mathcal{A}, \ldots, \mathsf{j}:\mathsf{C}_\mathsf{n} \in \mathcal{A}, \mathsf{j} \neq \mathsf{a}$, and
2. $(\mathsf{j},\mathsf{k}):\mathsf{R}' \notin \mathcal{A}$, and
3. $(\mathsf{i},\mathsf{j}):\mathsf{R}'' \notin \mathcal{A}, \mathsf{R}'' \neq \mathsf{R}$, and
4. $(\mathsf{l},\mathsf{j}):\mathsf{R}''' \notin \mathcal{A}, \mathsf{l} \neq \mathsf{i}$, and
5. $(\mathsf{i},\mathsf{o}):\mathsf{S} \notin \mathcal{A}, \mathsf{o} \neq \mathsf{j}, \mathsf{R}^\uparrow \cap \mathsf{S}^\uparrow \neq \emptyset$, and
6. $\mathsf{j}:\mathsf{C}_{\mathsf{n}+1} \notin \mathcal{A} : \forall\, \mathsf{i} \in \mathsf{1..n} : \mathsf{C}_{\mathsf{n}+1} \neq \mathsf{C}_\mathsf{i}$

then $\mathcal{A} := (\mathcal{A} \setminus \{(\mathsf{i},\mathsf{j}):\mathsf{R}, \mathsf{j}:\mathsf{C}_1, \ldots, \mathsf{j}:\mathsf{C}_\mathsf{n}\}) \cup \{\mathsf{i}:\exists\,\mathsf{R}\,.\,\mathsf{C}_1 \sqcap \ldots \sqcap \mathsf{C}_\mathsf{n}\}$

In the same way as above, we define an algorithm *contraction_alcnhr+*($\mathcal{A}$) that iteratively applies **RC2** to a consistent input ABox $\mathcal{A}$.

Proposition 30 The algorithm *contraction_alcnhr+*($\mathcal{A}$) transforms a consistent ABox $\mathcal{A}$ into an ABox $\mathcal{A}'$ that is consistent iff $\mathcal{A}$ is consistent.

As in the argumentation for *contraction_alc*($\mathcal{A}$) it is easy to see that the algorithm terminates and is sound and complete. The additional requirement in the premise guarantees that role assertions are not transformed into appropriate some-concept assertions if there exist more than one role assertion $(\mathsf{i},\mathsf{j}):\mathsf{R}$ for R where i occurs on the left-hand side. If there exists a common superrole, number restrictions might be imposed.

The role path contraction technique for ABoxes has several advantages:

1. The number of individuals and role assertions is reduced.
2. The some-concepts resulting from the contraction are automatically subject to the subtableaux and concept caching and merging techniques.
3. The ABox realization multiplies the savings effect due to iterated instance checking tests for the same individual over all named concepts in a TBox.

Our empirical tests with instance checking problems [Haarslev et al., 1999a] indicate that the role path contraction technique is very effective. The graph in Figure 4.6 shows execution times of different problems with ABox realization and in Figure 4.7 with instance checking. Due to our experiences, the set of ABox assertions is reduced by several orders of magnitude by the contraction technique. The speed gain of this technique is indicated in Figures 4.6 and 4.7. It shows that the contraction technique can result in a speed gain of about one order of magnitude. However, many of these

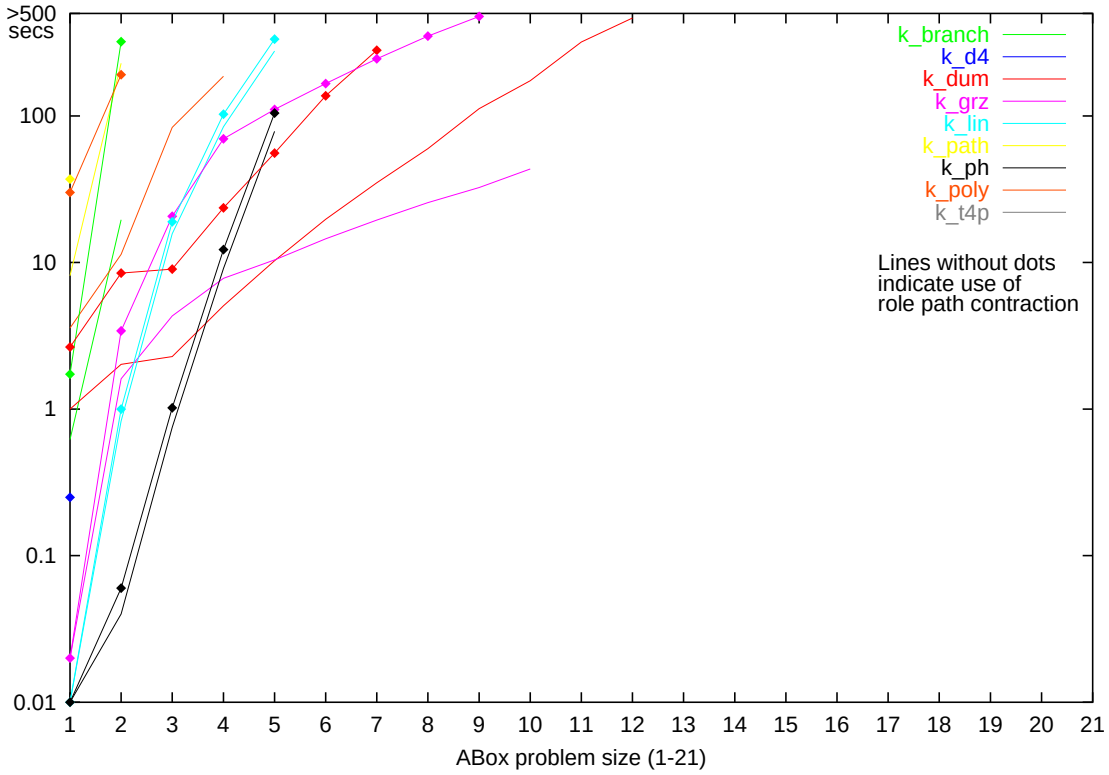


Figure 4.6: RACE: ABox realization with and without role path contraction. Lines without dots indicate the use of role path contraction. The lines of the same color with and without dots should be compared.

problems are still very hard and ask for the design of more sophisticated optimization techniques. The performance gain by using instance checking instead of realization is up to several orders of magnitude. Older ABox DL systems such as $\mathcal{KRIS}$ [Baader et al., 1992; Baader et al., 1994] are not able to check ABox consistency for any problem with size greater than 1 and even time out for some problems of size 1.

4.5 Summary: Benefit of the RACE Technology

The empirical tests reveal that new applications can be implemented using the RACE technology. First, we have considered a specification checking scenario in a telecommunications application. Second, a logical reconstruction of the medical knowledge implicitly represented in the UMLS thesaurus is now possible with a very large knowledge base. In the first context, the RACE technology made specification checking using automated theorem proving techniques possible. In both contexts,

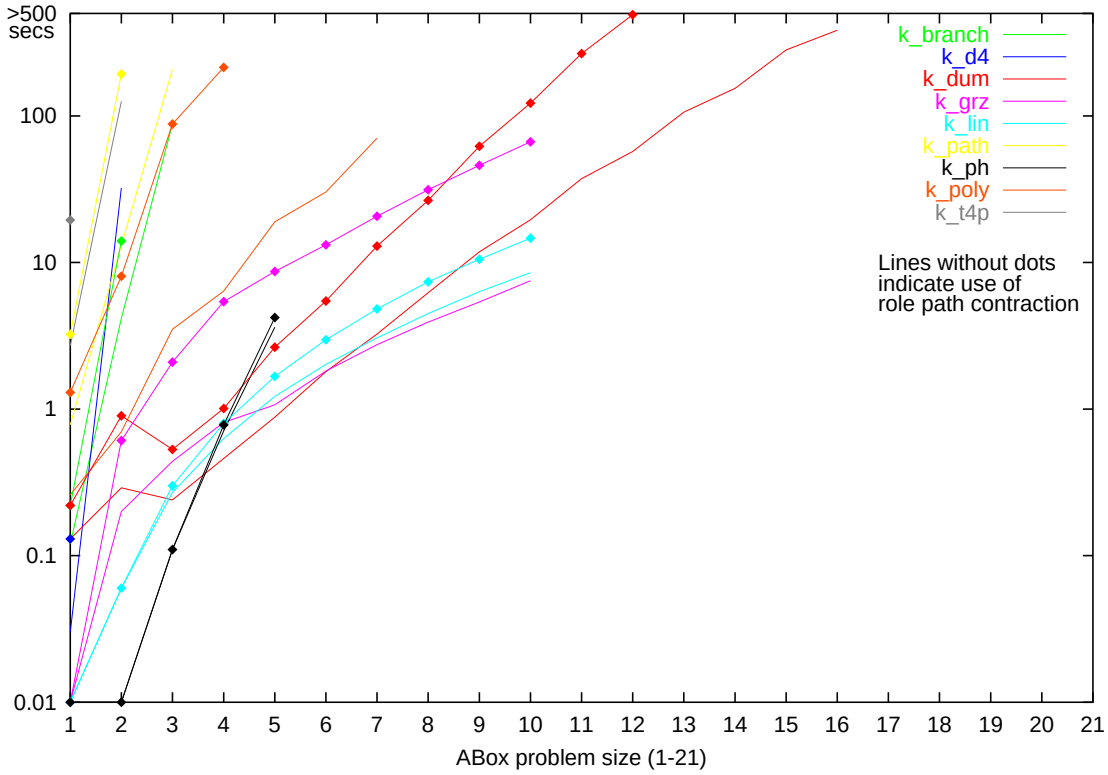


Figure 4.7: RACE: ABox instance checking with and without role path contraction. Lines without dots indicate the use of role path contraction. The lines of the same color with and without dots should be compared.

more reliable applications are possible due to sound and complete reasoning with RACE. Although much has been achieved, there are still instances of inference problems in both contexts that cannot be solved yet in a reasonable amount of time. Although RACE supports literal retraction of ABox assertions and incremental additions to TBoxes and ABox at the functional level, these operations might still be time-consuming. Additional research is necessary to either integrate known optimization techniques into the RACE architecture or to develop new optimization techniques that have not been used before.

Chapter 5

$\mathcal{ALCNH}_{R^+}$ Extended with Concrete Domains

An application domain where DLs have been successfully applied is *configuration*. Since effective reasoners for expressive languages are available only recently, most studies have been performed with DLs with limited expressivity (see e.g. [Wright et al., 1993]) or with undecidable logics [Buchheit et al., 1994; Buchheit et al., 1995]. The description logic $\mathcal{ALCNH}_{R^+}$ is an expressive but decidable representation language and with RACE an effective inference engine is available. However, the requirements derived from practical applications of DLs ask for even more expressive languages with features not covered in the previous chapters. It is well-known that reasoning about objects from other domains (so-called concrete domains, e.g. for the reals) is very important for practical applications as well [Baader & Hanschke, 1991a; Baader & Hanschke, 1991b]. Thus, an extension of the $\mathcal{ALCNH}_{R^+}$ knowledge representation system RACE with concrete domain is investigated.

Unfortunately, adding concrete domains to expressive description logics might lead to undecidable inference problems. For instance, in [Baader & Hanschke, 1992] it is proven that the logic $\mathcal{ALC}(\mathcal{D})$ plus an operator for the transitive closure of roles is undecidable. $\mathcal{ALCNH}_{R^+}$ offers transitive roles but no operator for the transitive closure of roles (see [Sattler, 1996, p. 342] for a detailed discussion about expressivity differences). In [Lutz, 1999a] it is shown that $\mathcal{ALC}(\mathcal{D})$ with generalized inclusion axioms (GCIs) is undecidable. Thus, if termination and soundness are to be retained, there is no way extending an $\mathcal{ALCNH}_{R^+}$ DL system such as RACE with concrete domains as in $\mathcal{ALC}(\mathcal{D})$ without losing completeness. Even if GCIs were discarded, $\mathcal{ALCNH}_{R^+}$ with concrete domains would be undecidable because $\mathcal{ALCNH}_{R^+}$ offers role hierarchies and transitive roles, which provide the same expressivity as GCIs. With role hierarchies it is possible to (implicitly) declare a universal role, which can be used in combination with a value restriction to achieve the same effect as with

GCIs.

Thus, $\mathcal{ALCNH}_{R^+}$ can only be extended with concrete domain operators with limited expressivity. In order to support practical modeling requirements at least to some extent, we pursue a pragmatic approach by supporting a limited kind of predicate exists restriction which supports only features (and no feature chains as in $\mathcal{ALC}(\mathcal{D})$, for details see below). The resulting language is called $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$. By proving soundness and completeness (and termination) of a tableaux calculus, the decidability of inference problems w.r.t. the language $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$ is proved. As shown in this section, $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$ can be used, for instance, as a basis for building practical application systems for solving certain classes of configuration problems, see also [Buchheit et al., 1995; Schröder et al., 1996].

5.1 The Description Logic $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$

The description logic $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$ augments $\mathcal{ALCNH}_{R^+}$ with so-called concrete domains.

5.1.1 The Concept Language of $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$

For presenting the syntax and semantics of the language $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$ a few additional definitions to those introduced for $\mathcal{ALCNH}_{R^+}$ (see Chapter 2) are required.

Definition 31 (Features) Let F be a set of *feature names* which is disjoint from R, the set of *role names*. For brevity, a feature name is also called a feature.

In accordance with [Baader & Hanschke, 1991a] we also define the notion of a concrete domain.

Definition 32 (Concrete Domain) A *concrete domain* $\mathcal{D}$ is a pair $(\Delta_\mathcal{D}, \Phi_\mathcal{D})$, where $\Delta_\mathcal{D}$ is a set called the domain, and $\Phi_\mathcal{D}$ is a set of predicate names. Each predicate name $\mathsf{P}_\mathcal{D}$ from $\Phi_\mathcal{D}$ is associated with an arity n and an n-ary predicate $\mathsf{P}_\mathcal{D}$. A concrete domain $\mathcal{D}$ is called *admissible* iff:

- The set of predicate names $\Phi_\mathcal{D}$ is closed under negation and $\Phi_\mathcal{D}$ contains a name $\top_\mathcal{D}$ for $\Delta_\mathcal{D}$,
- The satisfiability problem $\mathsf{P}_1^{n_1}(\mathsf{x}_{11}, \ldots, \mathsf{x}_{1n_1}) \wedge \ldots \wedge \mathsf{P}_\mathsf{m}^{n_\mathsf{m}}(\mathsf{x}_{\mathsf{m}1}, \ldots, \mathsf{x}_{\mathsf{m}n_\mathsf{m}})$ is decidable (m is finite, $\mathsf{P}_\mathsf{i}^{n_\mathsf{i}} \in \Phi_\mathcal{D}$, n_i is the arity of P, and x_{jk} is a name for an object from $\Delta_\mathcal{D}$).

We assume that $\bot_\mathcal{D}$ is the negation of the predicate $\top_\mathcal{D}$. Using the definitions from above, the syntax of concept terms in $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$ is extended as follows.

Definition 33 (Additional Concept Terms) Let $\mathsf{P} \in \Phi_{\mathcal{D}}$ be a predicate of the concrete domain, and let $\mathsf{f}, \mathsf{f}_1, \ldots, \mathsf{f}_k \in F$ be features, then the following expressions are also concept terms:

- $\exists\, \mathsf{f}_1, \ldots, \mathsf{f}_k \,.\, \mathsf{P}$ *(predicate exists restriction).*
- $\forall\, \mathsf{f} \,.\, \bot_{\mathcal{D}}$ *(no concrete domain filler restriction).*

The next definition gives a model-theoretic semantics to the language introduced above. Let $\mathcal{D} = (\Delta_{\mathcal{D}}, \Phi_{\mathcal{D}})$ be a concrete domain.

Definition 34 (Semantics) An *interpretation* $\mathcal{I}_{\mathcal{D}} = (\Delta^{\mathcal{I}}, \Delta^{\mathcal{D}}, \cdot^{\mathcal{I}})$ consists of a set $\Delta^{\mathcal{I}}$ (the abstract domain), a set $\Delta^{\mathcal{D}}$ (the domain of the 'concrete domain' $\mathcal{D}$) and an interpretation function $\cdot^{\mathcal{I}}$. The interpretation function $\cdot^{\mathcal{I}}$ maps each concept name C to a subset $\mathsf{C}^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, each role name R from R to a subset $\mathsf{R}^{\mathcal{I}}$ of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. Each feature f from F is mapped to a partial function $\mathsf{f}^{\mathcal{I}}$ from $\Delta^{\mathcal{I}}$ to $\Delta^{\mathcal{D}}$ where $\mathsf{f}^{\mathcal{I}}(\mathsf{a}) = \mathsf{x}$ will be written as $(\mathsf{a}, \mathsf{x}) \in \mathsf{f}^{\mathcal{I}}$. Each predicate name P from $\Phi_{\mathcal{D}}$ with arity n is mapped to a subset $\mathsf{P}^{\mathcal{I}}$ of $\Delta^n_{\mathcal{D}}$. Let $\mathsf{f}, \mathsf{f}_1, \ldots, \mathsf{f}_n$ be features and let P be a predicate name. Then, the interpretation function given in Definition 6 is extended to additional concept terms as follows:

$$
\begin{aligned}
(\exists\, \mathsf{f}_1, \ldots, \mathsf{f}_n \,.\, \mathsf{P})^{\mathcal{I}} := \{a \in \Delta^{\mathcal{I}} \mid \exists x_1, \ldots, x_n \in \Delta^{\mathcal{D}} : \\
(a, x_1) \in \mathsf{f}_1{}^{\mathcal{I}}, \ldots, (a, x_n) \in \mathsf{f}_n{}^{\mathcal{I}}, \\
(x_1, \ldots, x_n) \in \mathsf{P}^{\mathcal{I}}\} \\
(\forall\, \mathsf{f} \,.\, \bot_{\mathcal{D}})^{\mathcal{I}} := \{a \in \Delta^{\mathcal{I}} \mid \neg \exists x_1 \in \Delta^{\mathcal{D}} : (a, x_1) \in \mathsf{f}^{\mathcal{I}}\}
\end{aligned}
$$

The form of the terminological axioms for $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$ is the same as form $\mathcal{ALCNH}_{R^+}(\mathcal{D})$.

5.1.2 The Assertional Language of $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$

The assertional language is extended as follows.

Definition 35 (Additional Assertional Axioms) Let O_C be a set of names for concrete objects ($O_C \cap O = \emptyset$). If $\mathsf{f} \in F$ is a feature, $\mathsf{a} \in O_O$ is an individual name and $\mathsf{x}, \mathsf{x}_1, \ldots, \mathsf{x}_n \in O_C$ are names for concrete objects, then the following expressions are *assertional axioms* or *ABox assertions*:

- $(\mathsf{a}, \mathsf{x}) : \mathsf{f}$ *(concrete domain feature assertion),*
- $(\mathsf{x}_1, \ldots, \mathsf{x}_n) : \mathsf{P}$ *(concrete domain predicate assertion).*

Note that concrete domain objects are not considered as successors or predecessors in the sense defined in Section 2.1.

The interpretation function $\cdot^{\mathcal{I}}$ of the interpretation $\mathcal{I}_{\mathcal{D}}$ can be extended to the assertional language. Concrete objects from O_C are mapped to elements of $\Delta^{\mathcal{D}}$. An interpretation satisfies an assertional axiom $(\mathsf{a}, \mathsf{x}) : \mathsf{f}$ iff $(\mathsf{a}^{\mathcal{I}}, \mathsf{x}^{\mathcal{I}}) \in \mathsf{f}^{\mathcal{I}}$ and an assertional axiom $(\mathsf{x}_1, \ldots, \mathsf{x}_n) : \mathsf{P}$ iff $({\mathsf{x}_1}^{\mathcal{I}}, \ldots, {\mathsf{x}_n}^{\mathcal{I}}) \in \mathsf{P}^{\mathcal{I}}$.

5.2 Solving Configuration Problems with $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$

According to [Buchheit et al., 1994; Buchheit et al., 1995] configuration problem solving processes can be formalized as synthesis inference tasks. Following this approach, a solution of a configuration task is defined to be a *logical model* of the given knowledge base consisting of both the conceptual domain model (TBox, RBox) as well as the task specification (ABox).[1] The TBox and the RBox describe the configuration space.

Note that specific languages for describing the configuration space might be used. For instance, BHIBS [Cunis, 1991] is a configuration frame language which allows one to describe the properties of instances by specifying restrictions for the required values of named slots. The values can be either single objects or sets of objects, and the restrictions can be specified extensionally by directly giving concrete values like numbers, symbols or instances of concepts, or by intensionally describing sets and sequences of objects. The following example of an expression of the BHIBS-language describes the concept of a cylinder:

```
(is! (a Cylinder)
     (a Motorpart
        (part-of (a Motor))
        (displacement [1ccm 1000ccm])
        (has-parts
           (:set #[(a Cylinderpart) 4 6] :=
                 #[(a Piston)       1 1]
                 #[(a Piston-Rod)   1 1]
                 #[(a Valve)        2 4]))))
```

A Cylinder is required to be a Motorpart, to be part_of a Motor, to have a displacement of 1 to 1000ccm, and to have a set of 4 to 6 parts (has_parts) which are all Cylinderparts and it consists of exactly 1 Piston, exactly 1 Piston_Rod, and 2 to 4 Valves. This expression can be transformed to a terminological inclusion axiom of a description logic providing concrete domains. Let the concrete domain $\Re$ be defined as in [Baader & Hanschke, 1991b]: $\Re = (\mathbb{R}, \Phi_{\Re})$ where $\Phi_{\Re}$ is a set of predicates which are based

[1] [Buchheit et al., 1995] additionally considers relations defined with definite clauses.

on polynomial equations or inequations. The concrete domain $\Re$ is admissible (see also [Baader & Hanschke, 1991b]).

The cylinder example is translated as follows. Using the language $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$ the cylinder specification from above can be translated into description logics. First of all, a role box $\mathcal{R}$ is defined to contain the following role inclusion axioms.

$$\begin{aligned}
\textbf{has_cylinder_part} &\sqsubseteq \mathsf{has_part} \\
\textbf{has_piston_part} &\sqsubseteq \mathsf{has_part} \\
\textbf{has_piston_rod_part} &\sqsubseteq \mathsf{has_part} \\
\textbf{has_valve_part} &\sqsubseteq \mathsf{has_part}
\end{aligned}$$

Then, a TBox $\mathcal{T}$ is given as a set of terminological axioms. For instance, the following range restrictions are declared.

$$\begin{aligned}
&\top \sqsubseteq \forall\, \mathsf{has_cylinder_part} \,.\, \mathsf{Cylinder} \\
&\top \sqsubseteq \forall\, \mathsf{has_piston_part} \,.\, \mathsf{Piston} \\
&\top \sqsubseteq \forall\, \mathsf{has_piston_rod_part} \,.\, \mathsf{Piston_Rod} \\
&\top \sqsubseteq \forall\, \mathsf{has_valve_part} \,.\, \mathsf{Valve}
\end{aligned}$$

For Cylinderpart a so-called cover axiom is given. Moreover, additional axioms ensure the disjointness of more specific subconcepts of Cylinderpart.

$$\begin{aligned}
\textbf{Cylinderpart} &\sqsubseteq \mathsf{Piston} \sqcup \mathsf{Piston_Rod} \sqcup \mathsf{Valve} \\
\textbf{Piston} &\sqsubseteq \neg\mathsf{Piston_Rod} \sqcap \neg\mathsf{Valve} \\
\textbf{Piston_Rod} &\sqsubseteq \neg\mathsf{Piston} \sqcap \neg\mathsf{Valve} \\
\textbf{Valve} &\sqsubseteq \neg\mathsf{Piston} \sqcap \neg\mathsf{Piston_Rod}
\end{aligned}$$

Now another axiom relates a Cylinder to its parts. We assume that displacement is declared as a feature.

$$
\begin{aligned}
\textbf{Cylinder} \sqsubseteq\ & \mathsf{Motorpart} \sqcap \\
& \exists_{=1}\, \mathsf{part\_of} \sqcap \\
& \exists\, \mathsf{displacement} \,.\, \mathsf{displacement\_range} \sqcap \\
& \forall\, \mathsf{has\_parts} \,.\, \mathsf{Cylinderpart} \sqcap \\
& \exists_{\geq 4}\, \mathsf{has\_cylinder\_part} \sqcap \\
& \exists_{\leq 6}\, \mathsf{has\_cylinder\_part} \sqcap \\
& \exists_{=1}\, \mathsf{has\_piston\_part} \sqcap \\
& \exists_{=1}\, \mathsf{has\_piston\_rod\_part} \sqcap \\
& \exists_{\geq 2}\, \mathsf{has\_valve\_part} \sqcap \\
& \exists_{\leq 4}\, \mathsf{has\_valve\_part}
\end{aligned}
$$

The term $\mathsf{displacement\_range}$ denotes a unary predicate $\lambda_{\mathrm{Vol}}\, c \,.\, 0.001 \leq c \leq 1$ of a numeric concrete domain for the dimension *volume* with base unit m^3. Furthermore, we assume a predicate $\lambda_{\mathrm{Vol}}\, c \,.\, c \geq 0.5$ with the name $\mathsf{medium\_to\_large\_displacement}$ is declared for the concrete domain.

In our example, the ABox being used is very simple:

$\mathcal{A} = \{\mathsf{a} : \mathsf{Cylinder} \sqcap \exists\, \mathsf{displacement} \,.\, \mathsf{medium\_to\_large\_displacement}\}$.

In order to solve the configuration problem, the knowledge base $(\mathcal{T}, \mathcal{R}, \mathcal{A})$ is tested for consistency. If the knowledge base is consistent, there exists a model. The model can be interpreted as a solution to the configuration problem [Buchheit et al., 1994]. Note that $(\mathcal{T}, \mathcal{R}, \mathcal{A})$ is only a very simplified example for a representation of a configuration problem. For instance, using an ABox with additional assertions it is possible to explicitly specify some required cylinder parts etc.

In order to actually compute a solution to a configuration problem, a sound and complete calculus for the $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$ knowledge base consistency problem is required that terminates on any input. If the calculus returns "consistent" then (parts of) the internal structures used in the proof can be printed as a problem solution in a convenient form. We will return to this point after the discussion of the tableaux calculus for $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$.

5.3 Decidability of $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$

In the following a calculus to decide the consistency of an $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$ knowledge base $(\mathcal{T}, \mathcal{R}, \mathcal{A})$ is devised. The calculus is given as an extension to the calculus for $\mathcal{ALCNH}_{R^+}$ discussed in Section 4.1.

The following additional transformation rules are required to compute the negation normal form of $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$ concepts.

- $\neg\exists\, \mathsf{f}_1, \ldots, \mathsf{f}_n \,.\, \mathsf{P} \rightarrow \exists\, \mathsf{f}_1, \ldots, \mathsf{f}_n \,.\, \overline{\mathsf{P}} \sqcup \forall\, \mathsf{f}_1 \,.\, \bot_{\mathcal{D}} \sqcup \ldots \sqcup \forall\, \mathsf{f}_n \,.\, \bot_{\mathcal{D}}$
  where $\overline{\mathsf{P}}$ is the negation of P.
- $\neg\forall\, \mathsf{f} \,.\, \bot_{\mathcal{D}} \rightarrow \exists\, \mathsf{f} \,.\, \top_{\mathcal{D}}$

Definition 36 (Fork, Fork Elimination) If it holds that $\{(\mathsf{a}, \mathsf{x}_1) : \mathsf{f}, (\mathsf{a}, \mathsf{x}_2) : \mathsf{f}\} \subseteq \mathcal{A}$ then there exists a *fork* in $\mathcal{A}$. In case of a fork w.r.t. $\mathsf{x}_1, \mathsf{x}_2$, the replacement of every occurrence of x_2 in $\mathcal{A}$ by x_1 is called *fork elimination*.

In an augmented ABox for the language $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$ all forks have to be eliminated. It is easy to see that with forks being eliminated, Lemma 19 also holds for $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$.

Definition 37 (Concrete Object Ordering) A *concrete object ordering* '$\prec_C$' for elements of O_C occurring in an ABox $\mathcal{A}$ is defined as follows. If $\mathsf{y} \in O_C$ is introduced in $\mathcal{A}$, then $\mathsf{x} \prec_C \mathsf{y}$ for all concrete objects x already present in $\mathcal{A}$.

The completion rules of $\mathcal{ALCNH}_{R^+}$ are extended with an additional rule as follows:

Definition 38 (Additional Completion Rule)

R∃P The predicate exists rule (generating).
if 1. $\mathsf{a} : \exists\, \mathsf{f}_1, \ldots, \mathsf{f}_n \,.\, \mathsf{P} \in \mathcal{A}$, and
2. $\neg\exists \mathsf{x}_1, \ldots, \mathsf{x}_n \in O_C : \{(\mathsf{a}, \mathsf{x}_1) : \mathsf{f}_1, \ldots (\mathsf{a}, \mathsf{x}_n) : \mathsf{f}_n, (\mathsf{x}_1, \ldots, \mathsf{x}_n) : \mathsf{P}\} \subseteq \mathcal{A}$

then $\mathcal{A}' = \mathcal{A} \cup \{(\mathsf{a}, \mathsf{x}_1) : \mathsf{f}_1, \ldots (\mathsf{a}, \mathsf{x}_n) : \mathsf{f}_n, (\mathsf{x}_1, \ldots, \mathsf{x}_n) : \mathsf{P}\}$
where $\mathsf{x}_1, \ldots, \mathsf{x}_n \in O_C$ are not used in $\mathcal{A}$,
eliminate all forks $\{(\mathsf{a}, \mathsf{x}) : \mathsf{f}_i, (\mathsf{a}, \mathsf{x}_i) : \mathsf{f}_i\} \subseteq \mathcal{A}$
such that $(\mathsf{a}, \mathsf{x}) : \mathsf{f}_i$ remains in $\mathcal{A}$ if $\mathsf{x} \prec_C \mathsf{x}_i, \mathsf{i} \in 1..\mathsf{n}$

The rule R∃P is called a generating rule since it generates concrete objects.

Proposition 39 (Invariance) Let $\mathcal{A}$ and $\mathcal{A}'$ be ABoxes and $\mathcal{R}$ be a role box. Then:

1. If $\mathcal{A}'$ is derived from $\mathcal{A}$ w.r.t. $\mathcal{R}$ by applying a deterministic rule, then $\mathcal{A}$ is consistent w.r.t. $\mathcal{R}$ iff $\mathcal{A}'$ is consistent w.r.t. $\mathcal{R}$.

2. If $\mathcal{A}'$ is derived from $\mathcal{A}$ w.r.t. $\mathcal{R}$ by applying a nondeterministic rule, then $\mathcal{A}$ is consistent w.r.t. $\mathcal{R}$ if $\mathcal{A}'$ is consistent w.r.t. $\mathcal{R}$. Conversely, if $\mathcal{A}$ is consistent w.r.t. $\mathcal{R}$ and a nondeterministic rule is applicable to $\mathcal{A}$, then it can be applied in such a way that it yields an ABox $\mathcal{A}'$ consistent w.r.t. $\mathcal{R}$.

Proof.

1. "$\Leftarrow$" Due to the structure of the deterministic rules one can immediately verify that $\mathcal{A}$ is a subset of $\mathcal{A}'$. Therefore, $\mathcal{A}$ is consistent w.r.t. $\mathcal{R}$ if $\mathcal{A}'$ is consistent w.r.t. $\mathcal{R}$.

"⇒" In order to show that $\mathcal{A}'$ is consistent w.r.t. $\mathcal{R}$ after applying a deterministic rule to the consistent ABox $\mathcal{A}$, we examine each applicable rule separately. We assume that $\mathcal{I}_\mathcal{D} = (\Delta^\mathcal{I}, \Delta^\mathcal{D}, \cdot^\mathcal{I})$ satisfies $\mathcal{A}$ and $\mathcal{R}$. Then, we observe that it is an obvious consequence that $\mathsf{R}^\mathcal{I} \subseteq \mathsf{S}^\mathcal{I}$ iff $(\mathsf{R}, \mathsf{S}) \in \sqsubseteq^*_\mathcal{R}$.

If the conjunction rule is applied to $\mathsf{a}\!:\!\mathsf{C} \sqcap \mathsf{D} \in \mathcal{A}$, then we get a new ABox $\mathcal{A}' = \mathcal{A} \cup \{\mathsf{a}\!:\!\mathsf{C},\ \mathsf{a}\!:\!\mathsf{D}\}$. Since $\mathcal{I}_\mathcal{D}$ satisfies $\mathsf{a}\!:\!\mathsf{C} \sqcap \mathsf{D}$, $\mathcal{I}_\mathcal{D}$ satisfies $\mathsf{a}\!:\!\mathsf{C}$ and $\mathsf{a}\!:\!\mathsf{D}$ and therefore $\mathcal{A}'$.

If the role value restriction rule is applied to $\mathsf{a}\!:\!\forall\,\mathsf{R}\,.\,\mathsf{C} \in \mathcal{A}$, then there must be a role assertion $(\mathsf{a}, \mathsf{b})\!:\!\mathsf{S} \in \mathcal{A}$ with $\mathsf{S} \in \mathsf{R}^\downarrow$ such that $\mathcal{A}' = \mathcal{A} \cup \{\mathsf{b}\!:\!\mathsf{C}\}$. Since $\mathcal{I}_\mathcal{D}$ satisfies $\mathcal{A}$ and $\mathcal{R}$, it holds that $(\mathsf{a}^\mathcal{I}, \mathsf{b}^\mathcal{I}) \in \mathsf{S}^\mathcal{I}, \mathsf{S}^\mathcal{I} \subseteq \mathsf{R}^\mathcal{I}$. Since $\mathcal{I}_\mathcal{D}$ satisfies $\mathsf{a}\!:\!\forall\,\mathsf{R}\,.\,\mathsf{C}$, it holds that $\mathsf{b}^\mathcal{I} \in \mathsf{C}^\mathcal{I}$. Thus, $\mathcal{I}_\mathcal{D}$ satisfies $\mathsf{b}\!:\!\mathsf{C}$ and therefore $\mathcal{A}'$.

If the transitive role value restriction rule is applied to $\mathsf{a}\!:\!\forall\,\mathsf{R}\,.\,\mathsf{C} \in \mathcal{A}$, there must be an assertion $(\mathsf{a}, \mathsf{b})\!:\!\mathsf{S} \in \mathcal{A}$ with $\mathsf{S} \in \mathsf{T}^\downarrow$ for some $\mathsf{T} \in T$ and $\mathsf{T} \in \mathsf{R}^\downarrow$ such that we get $\mathcal{A}' = \mathcal{A} \cup \{\mathsf{b}\!:\!\forall\,\mathsf{T}\,.\,\mathsf{C}\}$. Since $\mathcal{I}_\mathcal{D}$ satisfies $\mathcal{A}$ and $\mathcal{R}$, we have $\mathsf{a}^\mathcal{I} \in (\forall\,\mathsf{R}\,.\,\mathsf{C})^\mathcal{I}$ and $(\mathsf{a}^\mathcal{I}, \mathsf{b}^\mathcal{I}) \in \mathsf{S}^\mathcal{I}, \mathsf{S}^\mathcal{I} \subseteq \mathsf{T}^\mathcal{I} \subseteq \mathsf{R}^\mathcal{I}$. Since $\mathsf{a}^\mathcal{I} \in (\forall\,\mathsf{T}\,.\,\mathsf{C})^\mathcal{I}$, $\mathcal{I}_\mathcal{D}$ satisfies $\mathsf{a}\!:\!\forall\,\mathsf{T}\,.\,\mathsf{C}$ and $\mathsf{T} \in T, \mathsf{T} \in \mathsf{R}^\downarrow$, it holds that $\mathsf{b}^\mathcal{I} \in (\forall\,\mathsf{T}\,.\,\mathsf{C})^\mathcal{I}$ unless there exists a successor c of b such that $(\mathsf{b}, \mathsf{c})\!:\!\mathsf{S}' \in \mathcal{A}$, $(\mathsf{b}^\mathcal{I}, \mathsf{c}^\mathcal{I}) \in \mathsf{S}'^\mathcal{I} \subseteq \mathsf{T}^\mathcal{I}$ and $\mathsf{c}^\mathcal{I} \notin \mathsf{C}^\mathcal{I}$. It follows from $(\mathsf{a}^\mathcal{I}, \mathsf{b}^\mathcal{I}) \in \mathsf{T}^\mathcal{I}$, $(\mathsf{b}^\mathcal{I}, \mathsf{c}^\mathcal{I}) \in \mathsf{T}^\mathcal{I}$, and $\mathsf{T} \in T$ that $(\mathsf{a}^\mathcal{I}, \mathsf{c}^\mathcal{I}) \in \mathsf{T}^\mathcal{I}, \mathsf{T}^\mathcal{I} \subseteq \mathsf{R}^\mathcal{I}$ and $\mathsf{a}^\mathcal{I} \notin (\forall\,\mathsf{R}\,.\,\mathsf{C})^\mathcal{I}$ in contradiction to the assumption. Thus, $\mathcal{I}_\mathcal{D}$ satisfies $\mathsf{b}\!:\!\forall\,\mathsf{T}\,.\,\mathsf{C}$ and therefore $\mathcal{A}'$.

If the universal concept restriction rule is applied to an individual a in $\mathcal{A}$ because of $\forall\,x\,.\,x\!:\!\mathsf{C} \in \mathcal{A}$, then $\mathcal{A}' = \mathcal{A} \cup \{\mathsf{a}\!:\!\mathsf{C}\}$. Since $\mathcal{I}_\mathcal{D}$ satisfies $\mathcal{A}$ and $\mathcal{R}$, it holds that $\mathsf{C}^\mathcal{I} = \Delta^\mathcal{I}$. Thus, it holds that $\mathsf{a}^\mathcal{I} \in \mathsf{C}^\mathcal{I}$ and $\mathcal{I}_\mathcal{D}$ satisfies $\mathcal{A}'$.

If the role exists restriction rule is applied to $\mathsf{a}\!:\!\exists\,\mathsf{R}\,.\,\mathsf{C} \in \mathcal{A}$, then we get the ABox $\mathcal{A}' = \mathcal{A} \cup \{(\mathsf{a}, \mathsf{b})\!:\!\mathsf{R},\ \mathsf{b}\!:\!\mathsf{C}\}$. Since $\mathcal{I}_\mathcal{D}$ satisfies $\mathcal{A}$ and $\mathcal{R}$, there exists a $y \in \Delta^\mathcal{I}$ such that $(\mathsf{a}^\mathcal{I}, y) \in \mathsf{R}^\mathcal{I}$ and $y \in \mathsf{C}^\mathcal{I}$. We define the interpretation function $\cdot^{\mathcal{I}'}$ such that $\mathsf{b}^{\mathcal{I}'} := y$ and $x^{\mathcal{I}'} := x^\mathcal{I}$ for $x \neq \mathsf{b}$. It is easy to show that $\mathcal{I}'_\mathcal{D} = (\Delta^\mathcal{I}, \Delta^\mathcal{D}, \cdot^{\mathcal{I}'})$ satisfies $\mathcal{A}'$.

If the number restriction exists rule is applied to $\mathsf{a}\!:\!\exists_{\geq n}\,\mathsf{R} \in \mathcal{A}$, then we get the ABox $\mathcal{A}' = \mathcal{A} \cup \{(\mathsf{a}, \mathsf{b}_\mathsf{k})\!:\!\mathsf{R} \mid \mathsf{k} \in 1..\mathsf{n}\} \cup \{\mathsf{b}_\mathsf{i} \neq \mathsf{b}_\mathsf{j} \mid \mathsf{i}, \mathsf{j} \in 1..\mathsf{n}, \mathsf{i} \neq \mathsf{j}\}$. Since $\mathcal{I}_\mathcal{D}$ satisfies $\mathcal{A}$ and $\mathcal{R}$, there must exist n distinct individuals $y_i \in \Delta^\mathcal{I}$, $i \in 1..n$ such that $(\mathsf{a}^\mathcal{I}, y_i) \in \mathsf{R}^\mathcal{I}$. We define the interpretation function $\cdot^{\mathcal{I}'}$ such that $\mathsf{b}_\mathsf{i}^{\mathcal{I}'} := y_i$ and $x^{\mathcal{I}'} := x^\mathcal{I}$ for $x \notin \{\mathsf{b}_1, \ldots, \mathsf{b}_\mathsf{n}\}$. It is easy to show that $\mathcal{I}'_\mathcal{D} = (\Delta^\mathcal{I}, \Delta^\mathcal{D}, \cdot^{\mathcal{I}'})$ satisfies $\mathcal{A}'$.

If the predicate exists rule is applied to $\mathsf{a}\!:\!\exists\,\mathsf{f}_1, \ldots, \mathsf{f}_\mathsf{n}\,.\,\mathsf{P} \in \mathcal{A}$, then we get the ABox $\mathcal{A}' = \mathcal{A} \cup \{(\mathsf{x}_1, \ldots, \mathsf{x}_\mathsf{n})\!:\!\mathsf{P}, (\mathsf{a}, \mathsf{x}_1)\!:\!\mathsf{f}_1, \ldots, (\mathsf{a}, \mathsf{x}_\mathsf{n})\!:\!\mathsf{f}_\mathsf{n}\}$. After fork elimination, some x_i may be replaced by z_i with $\mathsf{z}_\mathsf{i} \prec_C \mathsf{x}_\mathsf{i}$. Since $\mathcal{I}_\mathcal{D}$ satisfies $\mathcal{A}$ and $\mathcal{R}$, there exist $y_1, \ldots, y_n \in \Delta^\mathcal{D}$ such that $\forall i \in \{1, \ldots, n\} : (\mathsf{a}^\mathcal{I}, y_i) \in \mathsf{f}_\mathsf{i}^\mathcal{I}$ and $(y_1, \ldots, y_n) \in \mathsf{P}^\mathcal{I}$. We define the interpretation function $\cdot^{\mathcal{I}'}$ such that $\mathsf{x}_\mathsf{i}^{\mathcal{I}'} := y_i$ for all x_i not replaced by z_i and $(y_1, \ldots, y_n) \in \mathsf{P}^{\mathcal{I}'}$. The fork elimination strategy used in the R∃P rule guarantees that concrete objects introduced in previous steps are not eliminated. Thus, it is

ensured that the interpretation of x_i is not changed in $\mathcal{I}'_\mathcal{D}$. It is easy to see that $\mathcal{I}'_\mathcal{D} = (\Delta^\mathcal{I}, \Delta^\mathcal{D}, \cdot^{\mathcal{I}'})$ satisfies $\mathcal{A}'$.

2. "$\Leftarrow$" Assume that $\mathcal{A}'$ is satisfied by $\mathcal{I}'_\mathcal{D} = (\Delta^\mathcal{I}, \Delta^\mathcal{D}, \cdot^{\mathcal{I}'})$. By examining the nondeterministic rules we show that $\mathcal{A}$ is also consistent w.r.t. $\mathcal{R}$.

If $\mathcal{A}'$ is obtained from $\mathcal{A}$ by applying the disjunction rule, then $\mathcal{A}$ is a subset of $\mathcal{A}'$ and therefore satisfied by $\mathcal{I}'_\mathcal{D}$.

If $\mathcal{A}'$ is obtained from $\mathcal{A}$ by applying the number restriction merge rule to $\mathsf{a} : \exists_{\leq n}\, \mathsf{R} \in \mathcal{A}$, then there exist $\mathsf{b}_\mathsf{i}, \mathsf{b}_\mathsf{j}$ in $\mathcal{A}$ such that $\mathcal{A}' = \mathcal{A}[\mathsf{b}_\mathsf{i}/\mathsf{b}_\mathsf{j}]$. We define the interpretation function $\cdot^\mathcal{I}$ such that $\mathsf{b}_\mathsf{i}{}^\mathcal{I} := \mathsf{b}_\mathsf{j}{}^{\mathcal{I}'}$ and $\mathsf{x}^\mathcal{I} := \mathsf{x}^{\mathcal{I}'}$ for every $\mathsf{x} \neq \mathsf{b}_\mathsf{i}$. Obviously, $\mathcal{I}_\mathcal{D} = (\Delta^\mathcal{I}, \Delta^\mathcal{D}, \cdot^\mathcal{I})$ satisfies $\mathcal{A}$ and $\mathcal{R}$.

"$\Rightarrow$" We suppose that $\mathcal{I}_\mathcal{D} = (\Delta^\mathcal{I}, \Delta^\mathcal{D}, \cdot^\mathcal{I})$ satisfies $\mathcal{A}$ and $\mathcal{R}$ and a nondeterministic rule is applicable to an individual a in $\mathcal{A}$.

If the disjunction rule is applicable to $\mathsf{a} : \mathsf{C} \sqcup \mathsf{D} \in \mathcal{A}$ and $\mathcal{A}$ is consistent w.r.t. $\mathcal{R}$, it holds $\mathsf{a}^\mathcal{I} \in (\mathsf{C} \sqcup \mathsf{D})^\mathcal{I}$. It follows that either $\mathsf{a}^\mathcal{I} \in \mathsf{C}^\mathcal{I}$ or $\mathsf{a}^\mathcal{I} \in \mathsf{D}^\mathcal{I}$ (or both). Hence, the disjunction rule can be applied in a way that $\mathcal{I}_\mathcal{D}$ also satisfies the ABox $\mathcal{A}'$.

If the number restriction merge rule is applicable to $\mathsf{a} : \exists_{\leq n}\, \mathsf{R} \in \mathcal{A}$ and $\mathcal{A}$ is consistent w.r.t. $\mathcal{R}$, it holds $\mathsf{a}^\mathcal{I} \in (\exists_{\leq n}\, \mathsf{R})^\mathcal{I}$ and $\|\{b \,|\, (a, b) \in \mathsf{R}^\mathcal{I}\}\| \leq n$. However, it also holds $\|\{\mathsf{b} \,|\, (\mathsf{a}^\mathcal{I}, \mathsf{b}^\mathcal{I}) \in \mathsf{R}^\mathcal{I}\}\| > m$ with $m \geq n$. Without loss of generality we only need to consider the case that $m = n+1$. Thus, we can conclude by the Pigeonhole Principle that there exist at least two R-successors $\mathsf{b}_\mathsf{i}, \mathsf{b}_\mathsf{j}$ of a such that $\mathsf{b}_\mathsf{i}{}^\mathcal{I} = \mathsf{b}_\mathsf{j}{}^\mathcal{I}$. Since $\mathcal{I}_\mathcal{D}$ satisfies $\mathcal{A}$ and $\mathcal{R}$, at least one of the two individuals must be a new individual. Let us assume $\mathsf{b}_\mathsf{i} \in O_N$, then $\mathcal{I}_\mathcal{D}$ obviously satisfies $\mathcal{A}[\mathsf{b}_\mathsf{i}/\mathsf{b}_\mathsf{j}]$. □

Given an ABox $\mathcal{A}$, more than one rule might be applicable to $\mathcal{A}$. The completion strategy is the same a for $\mathcal{ALCN}\mathcal{H}_{R^+}$ (see Definition 24). However, a set of additional clash triggers are required for $\mathcal{ALCN}\mathcal{H}_{R^+}(\mathcal{D})^-$.

Definition 40 (Additional Clash Triggers) In addition to the clash triggers in Definition 25 the following additional clash triggers are applied.

- *No concrete domain feature clash*: $\{(\mathsf{a}, \mathsf{x}) : \mathsf{f}, \mathsf{a} : \forall\, \mathsf{f} \,.\, \bot_\mathcal{D}\} \subseteq \mathcal{A}$.
- *Concrete domain predicate clash*: $(\mathsf{x}_1^{(1)}, \ldots, \mathsf{x}_\mathsf{n}^{(1)}) : \mathsf{P}_1 \in \mathcal{A}, \ldots,$ $(\mathsf{x}_1^{(\mathsf{k})}, \ldots, \mathsf{x}_{\mathsf{n}_\mathsf{k}}^{(\mathsf{k})}) : \mathsf{P}_\mathsf{k} \in \mathcal{A}$ and the conjunction $\bigwedge_{\mathsf{i}=1}^{\mathsf{k}} \mathsf{P}_\mathsf{i}(\mathsf{x}_1^{(\mathsf{i})}, \ldots, \mathsf{x}_{\mathsf{n}_\mathsf{i}}^{(\mathsf{i})})$ is not satisfiable in $\mathcal{D}$. Note that this can be decided since $\mathcal{D}$ is required to be admissible.

Any ABox containing a clash is obviously unsatisfiable (w.r.t. an RBox $\mathcal{R}$). The purpose of the calculus is to generate a completion for an initial ABox $\mathcal{A}_\mathcal{T}$ that proves the consistency of $\mathcal{A}_\mathcal{T}$ (w.r.t. an RBox $\mathcal{R}$) or its inconsistency if no completion can be found.

The following lemma proves that whenever a generating rule has been applied to an individual $\mathsf{a} \in O_N$, the concept set $\sigma(\cdot, \mathsf{a})$ of a does not change for succeeding ABoxes. Note that the original ABox does not contain elements from O_N (see Definition 7).

Lemma 41 (Stability) Let $\mathcal{A}$ be an ABox and $\mathsf{a} \in O_N$ be in $\mathcal{A}$. Let a generating rule be applicable to a according to the completion strategy. Let $\mathcal{A}'$ be any ABox derivable from $\mathcal{A}$ by any (possibly empty) sequence of rule applications. Then:

1. No rule is applicable in $\mathcal{A}'$ to an individual $\mathsf{b} \in O_N$ with $\mathsf{b} \prec \mathsf{a}$
2. $\sigma(\mathcal{A}, \mathsf{a}) = \sigma(\mathcal{A}', \mathsf{a})$, i.e. the concept set of a remains unchanged in $\mathcal{A}'$.
3. If $\mathsf{b} \in O_N$ is in $\mathcal{A}$ with $\mathsf{b} \prec \mathsf{a}$ then b is an individual in $\mathcal{A}'$, i.e. the individual b is not substituted by another individual.

Proof. Since in the original input ABox no elements of O_N are mentioned, a rule must have been applied if $\mathsf{b} \prec \mathsf{a}$ holds. **1.** By contradiction: Suppose $\mathcal{A} = \mathcal{A}_0 \rightarrow_* \cdots \rightarrow_* \mathcal{A}_n = \mathcal{A}'$, where $*$ is element of the completion rules and a rule is applicable to an individual b with $\mathsf{b} \prec \mathsf{a}$ in $\mathcal{A}'$. Then there has to exist a minimal i with $i \in 1..n$ such that this rule is also applicable in $\mathcal{A}_i$. If a rule is applicable to a in $\mathcal{A}$ then no rule is applicable to b in $\mathcal{A}$ due to our strategy. So no rule is applicable to any individual c such that $\mathsf{c} \prec \mathsf{a}$ in $\mathcal{A}_0, \ldots, \mathcal{A}_{i-1}$. It follows that from $\mathcal{A}_{i-1}$ to $\mathcal{A}_i$ a rule is applied to a or to a d such that $\mathsf{a} \prec \mathsf{d}$. Using an exhaustive case analysis of all rules we can show that no new assertion of the form $\mathsf{b}\!:\!\mathsf{C}$ or $(\mathsf{b}, \mathsf{e})\!:\!\mathsf{R}$ can be added to $\mathcal{A}_{i-1}$. Therefore, no rule is applicable to b in $\mathcal{A}_i$. This is a contradiction to our assumption.

2. By contradiction: Suppose $\sigma(\mathcal{A}, \mathsf{a}) \neq \sigma(\mathcal{A}', \mathsf{a})$. Let b be the direct predecessor of a with $\mathsf{b} \prec \mathsf{a}$. A rule must have been applied to a and not to b because of point 1. Due to our strategy only generating rules are applicable to a that cannot add new elements to $\sigma(\cdot, \mathsf{a})$. This is an obvious contradiction.

3. This follows from point 1 and the completion strategy. □

In order to define a canonical interpretation from a completion $\mathcal{A}$, the notion of a specific blocking individual is introduced. This blocking individual is called a witness.

In order to show soundness of the calculus a so-called canonical interpretation is constructed from a completion. For the construction process the notion of a witness is defined.

Definition 42 (Witness) Let $\mathcal{A}$ be an ABox and $\mathsf{a}, \mathsf{b} \in O_N$ be individuals in $\mathcal{A}$. We call a the *witness* of b if the following conditions hold:

1. $\sigma(\mathcal{A}, \mathsf{a}) \supseteq \sigma(\mathcal{A}, \mathsf{b})$

2. $\mathsf{a} \prec \mathsf{b}$
3. $\neg\exists\, \mathsf{c}$ in $\mathcal{A} : \mathsf{c} \in O_N, \mathsf{c} \prec \mathsf{a}, \sigma(\mathcal{A}, \mathsf{c}) \supseteq \sigma(\mathcal{A}, \mathsf{b})$.

The next lemma proves the uniqueness of a witness for a blocked individual.

Lemma 43 Let $\mathcal{A}'$ be an ABox and a be a new individual in $\mathcal{A}'$. If a is blocked then

1. a has no direct successor (individual from O) and
2. a has exactly one witness.

Proof. **1.** By contradiction: Suppose that a is blocked in $\mathcal{A}'$ and $(\mathsf{a}, \mathsf{b}) : \mathsf{R} \in \mathcal{A}'$. There must exist an ancestor ABox $\mathcal{A}$ where a generating rule has been applied to a in $\mathcal{A}$. It follows from the definition of the generating rules that for every new individual c with $\mathsf{c} \prec \mathsf{a}$ in $\mathcal{A}$ we had $\sigma(\mathcal{A}, \mathsf{c}) \not\supseteq \sigma(\mathcal{A}, \mathsf{a})$. Since $\mathcal{A}'$ has been derived from $\mathcal{A}$ we can use Lemma 41 and conclude that for every new individual c with $\mathsf{c} \prec \mathsf{a}$ in $\mathcal{A}'$ we also have $\sigma(\mathcal{A}', \mathsf{c}) \not\supseteq \sigma(\mathcal{A}', \mathsf{a})$. Thus there cannot exist a blocking individual c for a in $\mathcal{A}'$. This is a contradiction to our hypothesis.

2. This follows directly from condition 3 in Definition 42. □

Definition 44 (Canonical Interpretation) Let $\mathcal{A}$ be a complete ABox that has been derived by the calculus from an augmented ABox $\mathcal{A}_\mathcal{T}$ w.r.t. the role box $\mathcal{R}$. Since $\mathcal{A}$ is clash-free, there exists a variable assignment α that satisfies (the conjunction of) all occurring assertions $(\mathsf{x}_1, \ldots, \mathsf{x}_\mathsf{n}) : \mathsf{P} \in \mathcal{A}$. We define the *canonical interpretation* $\mathcal{I}_\mathcal{C} = (\Delta^{\mathcal{I}_\mathcal{C}}, \Delta^\mathcal{D}, \cdot^{\mathcal{I}_\mathcal{C}})$ w.r.t. $\mathcal{A}$ and $\mathcal{R}$ as follows:

1. $\Delta^{\mathcal{I}_\mathcal{C}} := \{\mathsf{a} \mid \mathsf{a}$ is an individual in $\mathcal{A}\}$
2. $\mathsf{a}^{\mathcal{I}_\mathcal{C}} := \mathsf{a}$ iff a is mentioned in $\mathcal{A}$
3. $\mathsf{x}^{\mathcal{I}_\mathcal{C}} := \alpha(\mathsf{x})$ iff x is mentioned in $\mathcal{A}$
4. $\mathsf{a} \in \mathsf{A}^{\mathcal{I}_\mathcal{C}}$ iff $\mathsf{a} : \mathsf{A} \in \mathcal{A}$
5. $(\mathsf{a}, \alpha(\mathsf{x})) \in \mathsf{f}^{\mathcal{I}_\mathcal{C}}$ iff $(\mathsf{a}, \mathsf{x}) : \mathsf{f} \in \mathcal{A}$
6. $(\mathsf{a}, \mathsf{b}) \in \mathsf{R}^{\mathcal{I}_\mathcal{C}}$ iff $\exists\, \mathsf{c}_0, \ldots, \mathsf{c}_\mathsf{n}, \mathsf{d}_0, \ldots, \mathsf{d}_{\mathsf{n}-1}$ mentioned in $\mathcal{A}$:[2],
   (a) $\mathsf{n} \geq 1, \mathsf{c}_0 = \mathsf{a}, \mathsf{c}_\mathsf{n} = \mathsf{b}$, and
   (b) $(\mathsf{a}, \mathsf{c}_1) : \mathsf{S}_1$, $(\mathsf{d}_1, \mathsf{c}_2) : \mathsf{S}_2$, $\ldots$ $(\mathsf{d}_{\mathsf{n}-2}, \mathsf{c}_{\mathsf{n}-1}) : \mathsf{S}_{\mathsf{n}-1}$, $(\mathsf{d}_{\mathsf{n}-1}, \mathsf{b}) : \mathsf{S}_\mathsf{n} \in \mathcal{A}$, and

[2]Note that the variables $\mathsf{c}_0, \ldots, \mathsf{c}_\mathsf{n}, \mathsf{d}_0, \ldots, \mathsf{d}_{\mathsf{n}-1}$ not necessarily denote different individual names.

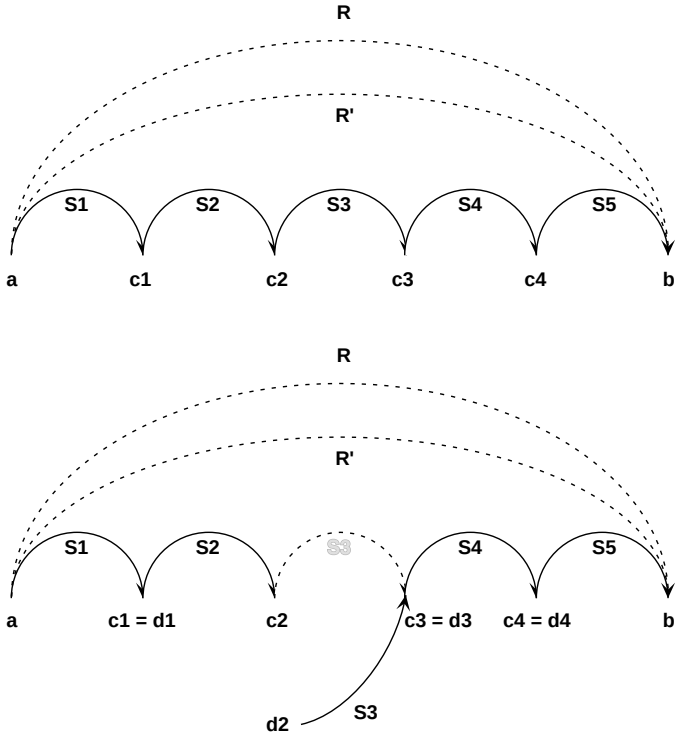


Figure 5.1: Construction of the canonical interpretation (two examples for case 6). In the lower example we assume that the individual $\mathsf{d2}$ is a witness for $\mathsf{c2}$ (see text).

(c) $\forall \mathsf{i} \in 0..\mathsf{n}-1$:
$\mathsf{d_i} = \mathsf{c_i}$ or
$\mathsf{d_i}$ is a witness for $\mathsf{c_i}$, and $(\mathsf{d_i}, \mathsf{c_{i+1}}) : \mathsf{S_{i+1}} \in \mathcal{A}$, and

(d) if $\mathsf{n} > 1$
$\forall\, \mathsf{i} \in 1..\mathsf{n} : \exists \mathsf{R}' \in T$, $\mathsf{R}' \in \mathsf{R}^{\downarrow}$, $\mathsf{S_i} \in \mathsf{R}'^{\downarrow}$
else
$\mathsf{S_1} \in \mathsf{R}^{\downarrow}$.

The construction of the canonical interpretation for the case 6 is illustrated with two examples in Figure 5.1. In the lower example the dotted arrow which is constructed due to case (6c) in the definition of the canonical interpretation, is called a *gap*. The following cases can be seen as special cases of case 6 introduced above ($\mathsf{n} = 1, \mathsf{c_0} = \mathsf{a}, \mathsf{c_1} = \mathsf{b}$):

- $\mathsf{c_0} = \mathsf{d_0}$: $(\mathsf{a}, \mathsf{b}) \in \mathsf{R}^{\mathcal{I}_C}$ iff $(\mathsf{c_0}, \mathsf{c_1}) : \mathsf{S_1} \in \mathcal{A}$ for a role $\mathsf{S_1} \in \mathsf{R}^{\downarrow}$.
- $\mathsf{c_0} \neq \mathsf{d_0}$: $(\mathsf{a}, \mathsf{b}) \in \mathsf{R}^{\mathcal{I}_C}$ iff $\mathsf{d_0}$ is a witness for $\mathsf{c_0}$, and
$(\mathsf{d_0}, \mathsf{c_1}) : \mathsf{S_1} \in \mathcal{A}$, for a role $\mathsf{S_1} \in \mathsf{R}^{\downarrow}$.

Due to Lemma 43, the canonical interpretation is well-defined because there exists a unique blocking individual (witness) for each individual that is blocked.

Theorem 45 (Soundness) Let $\mathcal{A}$ be a complete ABox that has been derived by the calculus from an augmented ABox $\mathcal{A}_\mathcal{T}$ w.r.t. the role box $\mathcal{R}$, then $\mathcal{A}_\mathcal{T}$ has a model which also satisfies all role axioms in $\mathcal{R}$.

Proof. Let $\mathcal{I}_\mathcal{C} = (\Delta^{\mathcal{I}_\mathcal{C}}, \Delta^\mathcal{D}, \cdot^{\mathcal{I}_\mathcal{C}})$ be the canonical interpretation for the ABox $\mathcal{A}$ constructed w.r.t. the TBox $\mathcal{T}$. $\mathcal{A}$ is clash-free.

Features are interpreted in the correct way: There can be no forks in $\mathcal{A}$ because (i) there are no forks in the augmented ABox $\mathcal{A}_\mathcal{T}$ and (ii) forks are immediately eliminated after an application of the R$\exists$P rule. This rule is the only rule that introduces new assertions of the form $(\mathsf{a}, \mathsf{x}) : \mathsf{f} \in \mathcal{A}$. Note that forks cannot be introduced by the R$\exists_{\leq n}$ rule due to the completion strategy. Thus, $\mathcal{I}_\mathcal{C}$ maps features to (partial) functions because the variable assignment α is a function.

All role inclusions in the RBox $\mathcal{R}$ are satisfied: For every $\mathsf{S} \sqsubseteq \mathsf{R}$ in $\mathcal{R}$ it holds that $\mathsf{S}^{\mathcal{I}_\mathcal{C}} \subseteq \mathsf{R}^{\mathcal{I}_\mathcal{C}}$ This can be shown as follows. If $(\mathsf{a}^{\mathcal{I}_\mathcal{C}}, \mathsf{b}^{\mathcal{I}_\mathcal{C}}) \in \mathsf{S}^{\mathcal{I}_\mathcal{C}}$, case 6 of Definition 44 must be applicable. Hence, there exists a chain of subroles possibly with gaps and witnesses (see Definition 44, case 6). Thus, the corresponding construction for $\mathcal{I}_\mathcal{C}$ adding $(\mathsf{a}^{\mathcal{I}_\mathcal{C}}, \mathsf{b}^{\mathcal{I}_\mathcal{C}})$ to $\mathsf{S}^{\mathcal{I}_\mathcal{C}}$ is also applicable to R since $\mathsf{S} \in \mathsf{R}^\downarrow$ (see 6d). Therefore, there is also tuple $(\mathsf{a}^{\mathcal{I}_\mathcal{C}}, \mathsf{b}^{\mathcal{I}_\mathcal{C}}) \in R^{\mathcal{I}_\mathcal{C}}$.

All transitivity axioms in the RBox $\mathcal{R}$ are satisfied, i.e. transitive roles are interpreted in the correct way: $\forall\, \mathsf{transitive}(\mathsf{R}) \in \mathcal{R} : \mathsf{R}^{\mathcal{I}_\mathcal{C}} = (\mathsf{R}^{\mathcal{I}_\mathcal{C}})^+$. If there exist $(\mathsf{a}^{\mathcal{I}_\mathcal{C}}, \mathsf{b}^{\mathcal{I}_\mathcal{C}}) \in R^{\mathcal{I}_\mathcal{C}}$ and $(\mathsf{b}^{\mathcal{I}_\mathcal{C}}, \mathsf{c}^{\mathcal{I}_\mathcal{C}}) \in R^{\mathcal{I}_\mathcal{C}}$ then case 6 in Definition 44 must have been applied for each tuple. But then, a chain of roles from a to c exists as well (possibly with gaps and witnesses) such that $(\mathsf{a}^{\mathcal{I}_\mathcal{C}}, \mathsf{c}^{\mathcal{I}_\mathcal{C}})$ is added to $\mathsf{R}^{\mathcal{I}_\mathcal{C}}$ as well.

In the following we prove that $\mathcal{I}_\mathcal{C}$ satisfies every assertion in $\mathcal{A}$.

For any $\mathsf{a} \neq \mathsf{b} \in \mathcal{A}$ or $(\mathsf{a}, \mathsf{b}) : \mathsf{R} \in \mathcal{A}$, $\mathcal{I}_\mathcal{C}$ satisfies them by definition.

For any $(\mathsf{a}, \mathsf{x}) : \mathsf{f} \in \mathcal{A}$, $\mathcal{I}_\mathcal{C}$ satisfies them by definition.

For any $(\mathsf{x}_1, \ldots, \mathsf{x}_\mathsf{n}) : \mathsf{P} \in \mathcal{A}$, $\mathcal{I}_\mathcal{C}$ satisfies them by definition. Since $\mathcal{A}$ is clash-free there exists a variable assignment such that the conjunction of all predicate assertions is satisfied. The variable assignment can be computed because the concrete domain is required to be admissible.

Next we consider assertions of the form $\mathsf{a} : \mathsf{C}$. We show by induction on the structure of C that $\mathsf{a}^{\mathcal{I}_\mathcal{C}} \in \mathsf{C}^{\mathcal{I}_\mathcal{C}}$.

If C is a concept name, then $\mathsf{a}^{\mathcal{I}_\mathcal{C}} \in \mathsf{C}^{\mathcal{I}_\mathcal{C}}$ by definition of $\mathcal{I}_\mathcal{C}$.

If $\mathsf{C} = \neg\mathsf{D}$, then D is a concept name since all concepts are in negation normal form (see Definition 18). $\mathcal{A}$ is clash-free and cannot contain $\mathsf{a} : \mathsf{D}$. Thus, $\mathsf{a}^{\mathcal{I}_\mathcal{C}} \notin \mathsf{D}^{\mathcal{I}_\mathcal{C}}$, i.e. $\mathsf{a}^{\mathcal{I}_\mathcal{C}} \in \Delta^{\mathcal{I}_\mathcal{C}} \setminus \mathsf{D}^{\mathcal{I}_\mathcal{C}}$. Hence $\mathsf{a}^{\mathcal{I}_\mathcal{C}} \in (\neg\mathsf{D})^{\mathcal{I}_\mathcal{C}}$.

If $\mathsf{C} = \mathsf{C}_1 \sqcap \mathsf{C}_2$ then (since $\mathcal{A}$ is complete) $\mathsf{a}:\mathsf{C}_1 \in \mathcal{A}$ and $\mathsf{a}:\mathsf{C}_2 \in \mathcal{A}$. By induction hypothesis, $\mathsf{a}^{\mathcal{I}_\mathcal{C}} \in {\mathsf{C}_1}^{\mathcal{I}_\mathcal{C}}$ and $\mathsf{a}^{\mathcal{I}_\mathcal{C}} \in {\mathsf{C}_2}^{\mathcal{I}_\mathcal{C}}$. Hence $\mathsf{a}^{\mathcal{I}_\mathcal{C}} \in (\mathsf{C}_1 \sqcap \mathsf{C}_2)^{\mathcal{I}_\mathcal{C}}$.

If $\mathsf{C} = \mathsf{C}_1 \sqcup \mathsf{C}_2$ then (since $\mathcal{A}$ is complete) either $\mathsf{a}:\mathsf{C}_1 \in \mathcal{A}$ or $\mathsf{a}:\mathsf{C}_2 \in \mathcal{A}$. By induction hypothesis, $\mathsf{a}^{\mathcal{I}_\mathcal{C}} \in {\mathsf{C}_1}^{\mathcal{I}_\mathcal{C}}$ or $\mathsf{a}^{\mathcal{I}_\mathcal{C}} \in {\mathsf{C}_2}^{\mathcal{I}_\mathcal{C}}$. Hence $\mathsf{a}^{\mathcal{I}_\mathcal{C}} \in (\mathsf{C}_1 \sqcup \mathsf{C}_2)^{\mathcal{I}_\mathcal{C}}$.

If $\mathsf{C} = \forall\, \mathsf{R}\,.\,\mathsf{D}$, then we have to show that for all $\mathsf{b}^{\mathcal{I}_\mathcal{C}}$ with $(\mathsf{a}^{\mathcal{I}_\mathcal{C}}, \mathsf{b}^{\mathcal{I}_\mathcal{C}}) \in \mathsf{R}^{\mathcal{I}_\mathcal{C}}$ it holds that $\mathsf{b}^{\mathcal{I}_\mathcal{C}} \in \mathsf{D}^{\mathcal{I}_\mathcal{C}}$. If $(\mathsf{a}^{\mathcal{I}_\mathcal{C}}, \mathsf{b}^{\mathcal{I}_\mathcal{C}}) \in \mathsf{R}^{\mathcal{I}_\mathcal{C}}$, then according to Definition 44, b is a successor of a via a chain of roles $\mathsf{S}_\mathsf{i} \in \mathsf{R}^\downarrow$ or there exist corresponding witnesses as domain elements of $\mathsf{S}_\mathsf{i} \in \mathsf{R}^\downarrow$, i.e. the chain might contain "gaps" with associated witnesses (see Figure 5.1). Since $(\mathsf{a}^{\mathcal{I}_\mathcal{C}}, \mathsf{b}^{\mathcal{I}_\mathcal{C}}) \in \mathsf{R}^{\mathcal{I}_\mathcal{C}}$ and ${\mathsf{S}_\mathsf{i}}^{\mathcal{I}_\mathcal{C}} \subseteq \mathsf{R}^{\mathcal{I}_\mathcal{C}}$ there exists tuples $({\mathsf{c}_\mathsf{i}}^{\mathcal{I}_\mathcal{C}}, {\mathsf{c}_{\mathsf{i}+1}}^{\mathcal{I}_\mathcal{C}}) \in {\mathsf{S}_\mathsf{i}}^{\mathcal{I}_\mathcal{C}}$. Due to Definition 44 it holds that $\forall \mathsf{i} \in 1..\mathsf{n} : \exists \mathsf{R}' \in T,\ \mathsf{R}' \in \mathsf{R}^\downarrow,\ \mathsf{S}_\mathsf{i} \in \mathsf{R}'^\downarrow$. Therefore $\mathsf{c}_\mathsf{k}:\forall\, \mathsf{R}'\,.\,\mathsf{D} \in \mathcal{A}, (\mathsf{k} \in 1..\mathsf{n}-1)$ because $\mathcal{A}$ is complete. For the same reason $\mathsf{b}:\mathsf{D} \in \mathcal{A}$. By induction hypothesis it holds that $\mathsf{b}^{\mathcal{I}_\mathcal{C}} \in \mathsf{D}^{\mathcal{I}_\mathcal{C}}$. As mentioned before, the chain of roles can have one or more "gaps" (see Figure 5.1). However, due to Definition 44 in case of a "gap" there exists a witness such that a similar argument as in case 6 can be applied, i.e. in case of a gap between c_i and $\mathsf{c}_{\mathsf{i}+1}$ with witness d_i for c_i, the blocking condition ensures that the concept set of the witness is a superset of the concept set of the blocked individual. Since it is assumed that $(\mathsf{d}_\mathsf{i}, \mathsf{c}_{\mathsf{i}+1}):\mathsf{S}_{\mathsf{i}+1} \in \mathcal{A}$ and $\mathcal{A}$ is complete it holds that $\mathsf{c}_{\mathsf{i}+1}:\forall\, \mathsf{R}'\,.\,\mathsf{D} \in \mathcal{A}$. Applying the same argument inductively, we can conclude that $\mathsf{c}_{\mathsf{n}-1}:\forall\, \mathsf{R}'\,.\,\mathsf{D} \in \mathcal{A}$ and again, we have $\mathsf{b}^{\mathcal{I}_\mathcal{C}} \in \mathsf{D}^{\mathcal{I}_\mathcal{C}}$ by induction hypothesis.

If $\mathsf{C} = \exists\, \mathsf{R}\,.\,\mathsf{D}$, then we have to show that there exists an individual $\mathsf{b}^{\mathcal{I}_\mathcal{C}} \in \Delta^{\mathcal{I}_\mathcal{C}}$ with $(\mathsf{a}^{\mathcal{I}_\mathcal{C}}, \mathsf{b}^{\mathcal{I}_\mathcal{C}}) \in \mathsf{R}^{\mathcal{I}_\mathcal{C}}$ and $\mathsf{b}^{\mathcal{I}_\mathcal{C}} \in \mathsf{D}^{\mathcal{I}_\mathcal{C}}$. Since ABox $\mathcal{A}$ is complete, we have either $(\mathsf{a}, \mathsf{b}):\mathsf{S} \in \mathcal{A}$ with $\mathsf{S} \in \mathsf{R}^\downarrow$ and $\mathsf{b}:\mathsf{D} \in \mathcal{A}$ or a is blocked by an individual c and $(\mathsf{c}, \mathsf{b}):\mathsf{S} \in \mathcal{A}$ (again $\mathsf{S} \in \mathsf{R}^\downarrow$). In the first case we have $(\mathsf{a}^{\mathcal{I}_\mathcal{C}}, \mathsf{b}^{\mathcal{I}_\mathcal{C}}) \in \mathsf{R}^{\mathcal{I}_\mathcal{C}}$ by the definition of $\mathcal{I}_\mathcal{C}$ (case 6, $\mathsf{n} = 1, \mathsf{c}_\mathsf{i} = \mathsf{d}_\mathsf{i}$) and $\mathsf{b}^{\mathcal{I}_\mathcal{C}} \in \mathsf{D}^{\mathcal{I}_\mathcal{C}}$ by induction hypothesis. In the second case there exists the witness c with $\mathsf{c}:\exists\, \mathsf{S}\,.\,\mathsf{D} \in \mathcal{A}$ and $\mathsf{S} \in \mathsf{R}^\downarrow$. By definition c cannot be blocked and by hypothesis $\mathcal{A}$ is complete. So we have an individual b with $(\mathsf{c}, \mathsf{b}):\mathsf{S} \in \mathcal{A}$ and $\mathsf{b}:\mathsf{D} \in \mathcal{A}$. By induction hypothesis we have $\mathsf{b}^{\mathcal{I}_\mathcal{C}} \in \mathsf{D}^{\mathcal{I}_\mathcal{C}}$ and by the definition of $\mathcal{I}_\mathcal{C}$ (case 6, $\mathsf{n} = 1, \mathsf{c}_\mathsf{i} \neq \mathsf{d}_\mathsf{i}$, d_i is a witness for c_i and $\mathsf{a} = \mathsf{c}_\mathsf{i}, \mathsf{c} = \mathsf{d}_\mathsf{i}$) we have $(\mathsf{a}^{\mathcal{I}_\mathcal{C}}, \mathsf{b}^{\mathcal{I}_\mathcal{C}}) \in \mathsf{R}^{\mathcal{I}_\mathcal{C}}$.

If $\mathsf{C} = \exists_{\geq n}\, \mathsf{R}$, we prove the hypothesis by contradiction. We assume that $\mathsf{a}^{\mathcal{I}_\mathcal{C}} \notin (\exists_{\geq n}\, \mathsf{R})^{\mathcal{I}_\mathcal{C}}$. Then there exist at most m $(0 \leq m < n)$ distinct S-successors of a with $\mathsf{S} \in \mathsf{R}^\downarrow$. Two cases can occur: (1) the individual a is not blocked in $\mathcal{I}_\mathcal{C}$. Then we have less than n S-successors of a in $\mathcal{A}$ and the R$\exists_{\geq n}$-rule is applicable to a. This contradicts the assumption that $\mathcal{A}$ is complete. (2) a is blocked by an individual c but the same argument as in case (1) holds and leads to the same contradiction.

For $\mathsf{C} = \exists_{\leq n}\, \mathsf{R}$ we show the goal by contradiction. Suppose that $\mathsf{a}^{\mathcal{I}_\mathcal{C}} \notin (\exists_{\leq n}\, \mathsf{R})^{\mathcal{I}_\mathcal{C}}$. Then there exist at least $n+1$ distinct individuals ${\mathsf{b}_1}^{\mathcal{I}_\mathcal{C}}, \ldots, {\mathsf{b}_{n+1}}^{\mathcal{I}_\mathcal{C}}$ such that $(\mathsf{a}^{\mathcal{I}_\mathcal{C}}, {\mathsf{b}_i}^{\mathcal{I}_\mathcal{C}}) \in \mathsf{R}^{\mathcal{I}_\mathcal{C}}$, $i \in 1..n+1$. The following two cases can occur. (1) The individual a is not blocked: We have $n+1$ $(\mathsf{a}, \mathsf{b}_i):\mathsf{S}_i \in \mathcal{A}$ with $\mathsf{S}_i \in \mathsf{R}^\downarrow$ and $\mathsf{S}_i \notin T$, $i \in 1..n+1$.

The R$\exists_{\leq n}$ rule cannot be applicable since $\mathcal{A}$ is complete and the b_i are distinct, i.e. $\mathsf{b}_i \not\equiv \mathsf{b}_j \in \mathcal{A}$, $i, j \in 1..n+1$, $i \neq j$. This contradicts the assumption that $\mathcal{A}$ is clash-free. (2) There exists a witness c for a with $(\mathsf{c}, \mathsf{b}_i) : \mathsf{S}_i \in \mathcal{A}$, $\mathsf{S}_i \in \mathsf{R}^{\downarrow}$, and $\mathsf{S}_i \notin T$, $i \in 1..n+1$. This leads to an analogous contradiction. Due to the construction of the canonical interpretation in case of a blocking condition (with c being the witness) and a non-transitive role R (R is required to be a simple role, see the syntactic restrictions for number restrictions and role boxes), there is no $(\mathsf{a}^{\mathcal{I}_C}, {\mathsf{b}_k}^{\mathcal{I}_C}) \in \mathsf{R}^{\mathcal{I}_C}$ if there is no $(\mathsf{c}^{\mathcal{I}_C}, {\mathsf{b}_k}^{\mathcal{I}_C}) \in \mathsf{R}^{\mathcal{I}_C}$ $(\mathsf{k} \in 1..\mathsf{n}+1)$.

If $\mathsf{C} = \exists\, \mathsf{f}_1, \ldots, \mathsf{f}_\mathsf{n} \,.\, \mathsf{P}$ we show that there exist concrete objects $y_1, \ldots, y_n \in \Delta^{\mathcal{D}}$ such that $(\mathsf{a}^{\mathcal{I}_C}, y_1) \in {\mathsf{f}_1}^{\mathcal{I}_C}, \ldots, (\mathsf{a}^{\mathcal{I}_C}, y_n) \in {\mathsf{f}_\mathsf{n}}^{\mathcal{I}_C}$ and $(y_1, \ldots, y_n) \in \mathsf{P}^{\mathcal{I}_C}$. The R$\exists$P rule generates assertions $(\mathsf{a}, \mathsf{x}_1) : \mathsf{f}_1, \ldots, (\mathsf{a}, \mathsf{x}_\mathsf{n}) : \mathsf{f}_\mathsf{n}, (\mathsf{x}_1, \ldots, \mathsf{x}_\mathsf{n}) : \mathsf{P}$. Since $\mathcal{A}$ is clash-free there is no concrete domain predicate clash. Hence there exists a variable assignment α that maps $\mathsf{x}_1, \ldots, \mathsf{x}_\mathsf{n}$ to elements of $\Delta^{\mathcal{D}}$. The conjunction of concrete domain predicates is satisfiable and $({\mathsf{x}_1}^{\mathcal{I}_C}, \ldots, {\mathsf{x}_\mathsf{n}}^{\mathcal{I}_C}) \in \mathsf{P}^{\mathcal{I}_C}$. By definition of $\mathcal{I}_\mathcal{C}$ it holds that $(\mathsf{a}^{\mathcal{I}_C}, {\mathsf{x}_1}^{\mathcal{I}_C}) \in {\mathsf{f}_1}^{\mathcal{I}_C}, \ldots, (\mathsf{a}^{\mathcal{I}_C}, {\mathsf{x}_\mathsf{n}}^{\mathcal{I}_C}) \in {\mathsf{f}_\mathsf{n}}^{\mathcal{I}_C}$. Thus, there exist $y_1, \ldots, y_n$ such that the above-mentioned requirements are fulfilled and therefore $\mathsf{a}^{\mathcal{I}_C} \in (\exists\, \mathsf{f}_1, \ldots, \mathsf{f}_\mathsf{n} \,.\, \mathsf{P})^{\mathcal{I}_C}$

If $\mathsf{C} = \forall\, \mathsf{f} \,.\, \bot_{\mathcal{D}}$ then we show that $\mathsf{a}^{\mathcal{I}_C} \in (\forall\, \mathsf{f} \,.\, \bot_{\mathcal{D}})^{\mathcal{I}_C}$. Because $\mathcal{A}$ is clash-free, there cannot be an assertion $(\mathsf{a}, \mathsf{x}) : \mathsf{f} \in \mathcal{A}$ for some x in O_c and an $\mathsf{f} \in F$. Thus, it does not hold that there exists $(\mathsf{a}^{\mathcal{I}_C}, y) \in \mathsf{f}^{\mathcal{I}_C}$ and hence $\mathsf{a}^{\mathcal{I}_C} \in (\forall\, \mathsf{f} \,.\, \bot_{\mathcal{D}})^{\mathcal{I}_C}$.

If $\forall\, x \,.\, x : \mathsf{D} \in \mathcal{A}$, then –due to the completeness of $\mathcal{A}$– for each individual a in $\mathcal{A}$ we have $\mathsf{a} : \mathsf{D} \in \mathcal{A}$ and, by the previous cases, $\mathsf{a}^{\mathcal{I}_C} \in \mathsf{D}^{\mathcal{I}_C}$. Thus, $\mathcal{I}_\mathcal{C}$ satisfies $\forall\, x \,.\, x : \mathsf{D}$. Finally, since $\mathcal{I}_\mathcal{C}$ satisfies all assertions in $\mathcal{A}$, $\mathcal{I}_\mathcal{C}$ satisfies $\mathcal{A}$.

□

Theorem 46 (Completeness) Let $\mathcal{A}_\mathcal{T}$ be an augmented ABox and $\mathcal{R}$ be a role box. If $\mathcal{A}_\mathcal{T}$ is consistent w.r.t. $\mathcal{R}$, then there exists at least one completion $\mathcal{A}'$ being computed by applying the completion rules w.r.t. the role box $\mathcal{R}$.

Proof. By contraposition: Obviously, an ABox containing a clash is inconsistent. If there does not exists a completion of $\mathcal{A}_\mathcal{T}$, then it follows from Proposition 39 that the ABox $\mathcal{A}_\mathcal{T}$ is inconsistent w.r.t. the role box $\mathcal{R}$. □

Definition 47 (Maximum Number of Concepts) Let $\mathcal{A}$ be a completion of an augmented ABox. Then,
$n_\mathcal{A} := ||\{\mathsf{C} | \exists\, (\mathsf{a} : \mathsf{D}) \in \mathcal{A} : \mathsf{C} \in \mathit{subs}(\mathsf{D}), \text{or } \exists (\forall\, x \,.\, x : \mathsf{D}) \in \mathcal{A} : \mathsf{C} \in \mathit{subs}(\mathsf{D})\}||$ is called the maximum number of concepts in $\mathcal{A}$. The function *subs* applied to a concept D returns the set of all concepts appearing as substrings in D (incl. D).

Note that $n_\mathcal{A}$ is bounded by the length of the string of the augmented ABox $\mathcal{A}$. In the following we assume that $||\cdot||$ returns the cardinality of a set plus 1.

Lemma 48 Let $\mathcal{A}$ be a completion of an augmented ABox $\mathcal{A}_\mathcal{T}$. Furthermore, let $T_\mathcal{R}$ be the finite set of transitive roles mentioned in a role box $\mathcal{R}$. In any set X consisting of individuals occurring in $\mathcal{A}$ with a cardinality greater than $2^{||T_\mathcal{R}||\times n_\mathcal{A}}$ there exist at least two individuals $\mathsf{a}, \mathsf{b} \in X$ whose concept sets are equal.

Proof. The only rule that generates assertions with concepts not already mentioned in $\mathcal{A}$ is the R$\forall_+$C rule. New concepts of the form $\forall\,\mathsf{T}\,.\,\mathsf{C}$ may be generated. The number of these concepts is bounded by $||T_\mathcal{R}|| \times n_\mathcal{A}$ because there are only $||T_\mathcal{R}||$ transitive roles mentioned in the role box $\mathcal{R}$ and only $n_\mathcal{A}$ different concepts in $\mathcal{A}$. There cannot exist more than $2^{||T_\mathcal{R}||\times n_\mathcal{A}}$ different concept sets for the individuals in $\mathcal{A}'$. If we have $2^{||T_\mathcal{R}||\times n_\mathcal{A}}$ individuals with different concept sets, then there can be no additional individual with a new concept set. □

Lemma 49 Let $\mathcal{A}_\mathcal{T}$ be an augmented ABox and let $\mathcal{A}'$ be a completion of $\mathcal{A}_\mathcal{T}$ w.r.t. $\mathcal{R}$. Furthermore, let $T_\mathcal{R}$ be the finite set of transitive roles mentioned in the role box $\mathcal{R}$. Then, there occur at most $2^{||T_\mathcal{R}||\times n_\mathcal{A}}$ non-blocked new individuals in $\mathcal{A}'$.

Proof. Suppose we have $2^{||T_\mathcal{R}||\times n_\mathcal{A}} + 1$ non-blocked new individuals in $\mathcal{A}'$. From Lemma 48 we know that there exist at least two individuals a, b in $\mathcal{A}'$ such that $\sigma(\mathcal{A}', \mathsf{a}) = \sigma(\mathcal{A}', \mathsf{b})$. By Definition 21 we have either $\mathsf{a} \prec \mathsf{b}$ or $\mathsf{b} \prec \mathsf{a}$. Assume without loss of generality that $\mathsf{a} \prec \mathsf{b}$ holds. The condition $\sigma(\mathcal{A}', \mathsf{a}) = \sigma(\mathcal{A}', \mathsf{b})$ implies $\sigma(\mathcal{A}', \mathsf{a}) \supseteq \sigma(\mathcal{A}', \mathsf{b})$ and therefore a is a blocking individual for b (see Definition 42). This contradicts the hypothesis that b is not blocked. □

Theorem 50 (Termination) Let $\mathcal{A}_\mathcal{T}$ be an augmented $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$ ABox (with numbers occurring in number restrictions expressed in binary). Every completion of $\mathcal{A}_\mathcal{T}$ w.r.t. a role box $\mathcal{R}$ is finite and its size is $O(2^{4n})$ where $n = ||T_\mathcal{R}|| \times n_0$.

Proof. Let $\mathcal{A}'$ be a completion of $\mathcal{A}_\mathcal{T}$. From Lemma 49 we know that $\mathcal{A}'$ has at most $2^{||T||\times n_{\mathcal{A}'}} \leq 2^n$ non-blocked new individuals. Therefore, a total of at most $m \times 2^n$ new individuals can exist in $\mathcal{A}'$, where m is the maximum number of direct successors for any individual in $\mathcal{A}'$.

Note that m is bounded by the number of $\exists\,\mathsf{R}\,.\,\mathsf{C}$ concepts ($\leq n$) plus the total sum of numbers occurring in $\exists_{\geq n}\,\mathsf{R}$. Since numbers are expressed in binary, their sum is bounded by $2^{n_0} (\leq 2^n)$. Hence, we have $m \leq 2^n + n$. Since the number of individuals in the initial ABox is also bounded by n, the total number of individuals in $\mathcal{A}'$ is at most $m \times (2^n + n) \leq (2^n + n) \times (2^n + n)$, i.e. $O(2^{2n})$.

The number of different assertions of the form $\mathsf{a}\!:\!\mathsf{C}$ or $\forall\, x\,.\,x\!:\!\mathsf{C}$ in which each individual in $\mathcal{A}'$ can be involved, is bounded by n and each assertion has a size linear in n. Hence, the total size of these assertions is bounded $n \times n \times 2^{2n}$, i.e. $O(2^{3n})$.

The number of different assertions of the form $(\mathsf{a},\mathsf{b}):\mathsf{R}$ or $\mathsf{a} \neq \mathsf{b}$ is bounded by $(2^{2n})^2$, i.e. $O(2^{4n})$.

The number of different assertions of the form $(\mathsf{a},\mathsf{x}):\mathsf{f}$ is bounded by $O(2^{2n})$ due to fork elimination.

The number of different assertions of the form $(\mathsf{x}_1,\ldots,\mathsf{x}_\mathsf{n}):\mathsf{P}$ is bounded by $n+(n\times 2^{2n})$, i.e. $O(2^{3n})$. The initial set of concrete domain predicate assertions is bounded by n. In addition, for each individual there may be n concept assertions yielding additional predicate assertions.

In conclusion, we have a size of $O(2^{4n})$ for $\mathcal{A}'$. □

Theorem 51 (Decidability) Checking whether a knowledge base $(\mathcal{T},\mathcal{R},\mathcal{A})$ is consistent is a decidable problem.

Proof. Given a knowledge base $(\mathcal{T},\mathcal{R},\mathcal{A})$, an augmented ABox $\mathcal{A}_\mathcal{T}$ can be constructed in linear time. Thus, the claim follows immediately from Lemma 19 and Theorems 45, 46, and 50. □

5.4 Applying $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$: Configuration Revisited

In the previous section the decidability of the ABox consistency problem for the language $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$ has been shown. Thus, in principle all configuration problems formalized as consistency queries for $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$ knowledge bases as indicated in Section 5.2 can be solved. If the input knowledge base is consistent, the configuration will be represented by a model represented by the canonical interpretation which can be derived from a completion. However, due to the fact that the algorithm is nondeterministic, some problems might remain.

5.4.1 Unintended Blocking

In the context of configuration, blocking might lead to an "undesirable" model. Let us consider the ABox $\{\mathsf{a}:\mathsf{A}\sqcup\mathsf{C}\}$, the TBox $\{\mathsf{C}\sqsubseteq\exists\mathsf{R}.\mathsf{A}\sqcup\mathsf{C}\}$ and an empty role box. One possible completion that might be derived by a concrete implementation of the knowledge base consistency algorithm is the following:

$\{\forall\, x\,.\, x:(\neg\mathsf{C}\sqcup\exists\mathsf{R}.\mathsf{A}\sqcup\mathsf{C}),$
$\mathsf{a}:(\mathsf{A}\sqcup\mathsf{C}),\mathsf{a}:(\neg\mathsf{C}\sqcup\exists\mathsf{R}.\mathsf{A}\sqcup\mathsf{C}),\mathsf{a}:\mathsf{C},\mathsf{a}:\exists\mathsf{R}.\mathsf{A}\sqcup\mathsf{C},$
$(\mathsf{a},\mathsf{b}):\mathsf{R},\mathsf{b}:(\mathsf{A}\sqcup\mathsf{C}),\mathsf{b}:(\neg\mathsf{C}\sqcup\exists\mathsf{R}.\mathsf{A}\sqcup\mathsf{C}),\mathsf{b}:\mathsf{C},\mathsf{b}:\exists\mathsf{R}.\mathsf{A}\sqcup\mathsf{C},$
$(\mathsf{b},\mathsf{c}):\mathsf{R},\mathsf{c}:(\mathsf{A}\sqcup\mathsf{C}),\mathsf{c}:(\neg\mathsf{C}\sqcup\exists\mathsf{R}.\mathsf{A}\sqcup\mathsf{C}),\mathsf{c}:\mathsf{C},\mathsf{c}:\exists\mathsf{R}.\mathsf{A}\sqcup\mathsf{C}\}$

This is a completion with c being blocked by b. Hence, the canonical interpretation contains a loop w.r.t. to the role R. Whether this is acceptable or not might depend

on the application context of the configuration solution. However, it should be noted that in this specific case there also exists a completion without a blocked individual. The configuration example presented in Section 5.2 is solved without blocking.

5.4.2 Limited Expressivity

For engine configuration, a set of cylinders all of which have equal piston displacements might be required. However, with $\mathcal{ALCN}\mathcal{H}_{R^+}(\mathcal{D})^-$ concrete domains predicates can only be established for a single individual, i.e. a single cylinder, rather than between different cylinders. A whole set of cylinders being part of an engine can only be constrained using an ABox and respective concrete domain assertions. Thus, only a fixed set of individuals can be considered during the configuration process. If it is not clear in beforehand whether a 4-, 6- or 8-cylinder engine will be required, a more expressive description logic is needed.

5.4.3 Analysis of an Extension of $\mathcal{ALCN}\mathcal{H}_{R^+}(\mathcal{D})^-$

A possibility for extending the expressivity of $\mathcal{ALCN}\mathcal{H}_{R^+}(\mathcal{D})^-$ might be to employ the predicate exists restriction of $\mathcal{ALC}(\mathcal{D})$ which offers feature chains [Baader & Hanschke, 1991a]. We call the language $\mathcal{ALCN}\mathcal{H}_{R^+}(\mathcal{D})$. Unfortunately, it holds that $\mathcal{ALCN}\mathcal{H}_{R^+}$ augmented with a predicate exists restriction supporting feature chains as in $\mathcal{ALC}(\mathcal{D})$ is undecidable because in [Lutz, 1999a] it is shown that $\mathcal{ALC}(\mathcal{D})$ with generalized inclusion axioms (GCIs) is undecidable. $\mathcal{ALCN}\mathcal{H}_{R^+}$ offers role hierarchies and transitive roles which provide the same expressivity as GCIs.

An undecidability proof may lead to insights about how to come up with new operators or syntactic restrictions of existing operators in order to develop a representation language that can cope with specific application requirements not covered by less expressive (decidable) languages. Since the GCI-based undecidability proof with Turing machines presented in [Lutz, 1999a] is rather involved, we give a more direct proof based on transitive roles and role hierarchies and demonstrate that even if TBoxes are discarded, $\mathcal{ALCN}\mathcal{H}_{R^+}$ with concrete domains is undecidable in general.

The syntax and semantics of $\mathcal{ALCN}\mathcal{H}_{R^+}(\mathcal{D})$ is a slightly modified variant of the syntax and semantics of $\mathcal{ALCN}\mathcal{H}_{R^+}(\mathcal{D})^-$.

Definition 52 (New Predicate Exists Restriction) In $\mathcal{ALCN}\mathcal{H}_{R^+}(\mathcal{D})$ a specific subset $A \subseteq S$ of simple roles called *attributes* is distinguished (see Definition 1 for the definition of the set S of simple roles). If $\mathsf{a}_1\mathsf{a}_2 \cdots \mathsf{a}_{n-1}$ are attributes and f_n is a feature, then a composition of attributes and features (written $\mathsf{a}_1\mathsf{a}_2 \cdots \mathsf{a}_{n-1}\mathsf{f}_n$) is called a chain (with length n). A single feature (i.e. a chain of length 1) is also called a chain. If $\mathsf{P} \in \Phi_{\mathcal{D}}$ is a predicate of the concrete domain $\mathcal{D}$ and $\mathsf{u}_1, \ldots, \mathsf{u}_k$ are chains, then the following expression is a concept term: $\exists\, \mathsf{u}_1, \ldots, \mathsf{u}_k \,.\, \mathsf{P}$ *(predicate*

exists restriction). In addition to $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$, attributes can be used instead of roles in value and exists restrictions.

Each attribute a from A is mapped to a partial function $\mathsf{a}^{\mathcal{I}}$ from $\Delta^{\mathcal{I}}$ to $\Delta^{\mathcal{I}}$. If $\mathsf{u} = \mathsf{a}_1 \cdots \mathsf{a}_{n-1}\mathsf{f}_n$ is a chain, then $\mathsf{u}^{\mathcal{I}}$ denotes the composition $\mathsf{a}_1 \circ \ldots \circ \mathsf{a}_{n-1} \circ \mathsf{f}_n$ of partial functions ${\mathsf{a}_1}^{\mathcal{I}}, \ldots, {\mathsf{a}_{n-1}}^{\mathcal{I}}{\mathsf{f}_n}^{\mathcal{I}}$. The interpretation function is modified as follows:

$$(\exists\, \mathsf{u}_1, \ldots, \mathsf{u}_n \,.\, \mathsf{P})^{\mathcal{I}} := \{a \in \Delta^{\mathcal{I}} \mid \exists x_1, \ldots, x_n \in \Delta^{\mathcal{D}} : (a, x_1) \in {\mathsf{u}_1}^{\mathcal{I}}, \ldots, (a, x_n) \in {\mathsf{u}_n}^{\mathcal{I}}, (x_1, \ldots, x_n) \in \mathsf{P}^{\mathcal{I}}\}$$

Proposition 53 (Undecidability of $\mathcal{ALCNH}_{R^+}(\mathcal{D})$) The concept consistency problem for $\mathcal{ALCNH}_{R^+}(\mathcal{D})$ is not decidable.

The proposition can be proven by a reduction from the Post Correspondence Problem (PCP). The general idea of the proof is a slight variation of the undecidability proofs for the description logics $\mathcal{ALC}(\mathcal{D})$ with a transitivity operator [Baader & Hanschke, 1992] and $\mathcal{ALCRP}(\mathcal{D})$ [Lutz, 1998; Lutz & Möller, 1997].

Proof. A Post Correspondence Problem S is defined as follows. Given a nonempty finite set $S = \{(l_i, r_i) \mid i = 1, \ldots m\}$, where l_i and r_i are words over an alphabet Σ, a solution of S is a sequence of indices $i_1, \ldots, i_k$ with $k \geq 1$ such that the concatenations $wl = l_{i_1} \ldots l_{i_k}$ and $wr = r_{i_1} \ldots r_{i_k}$ denote the same word. The PCP is known to be undecidable if Σ contains at least two symbols.

For the reduction, the elements of Σ are viewed as digits from $\{1, \ldots, \text{B-1}\}$ at base B, where $B := |\Sigma| + 1$. $\overline{w}$ denotes the nonnegative integer at base 10 which the (nonempty) word w represents at base B (see also [Baader & Hanschke, 1992]). If vw is the concatenation of two words $v, w \in \Sigma^*$, then $\overline{vw} = \overline{v} * B^{|w|} + \overline{w}$, where $|w|$ is the length of the word w. The function $w \mapsto \overline{w}$ is a 1–1-mapping from Σ^* into the set of nonnegative integers. Let w_l, w_r be features and $f_1, \ldots, f_m$ be attributes. Furthermore, let R be a transitive superrole of the attributes f_i $(i \in 1, \ldots, m)$. Then, for a given instance S of the PCP we define a concept $C(S)$:

$$\begin{aligned} C(S) \;\doteq\; & \exists w_l.\mathit{null\text{-}p} \sqcap \exists w_r.\mathit{null\text{-}p} \\ & \sqcap_{i=1}^{m}(\exists w_l, f_i w_l.\mathit{constr\text{-}p}_l^i \sqcap \exists w_r, f_i w_r.\mathit{constr\text{-}p}_r^i) \sqcap \\ & \forall R.\, \sqcap_{i=1}^{m} (\exists w_l, f_i w_l.\mathit{constr\text{-}p}_l^i \sqcap \exists w_r, f_i w_r.\mathit{constr\text{-}p}_r^i) \sqcap \\ & \forall R. \exists w_l, w_r.\mathit{notequal\text{-}p} \end{aligned}$$

The predicates used are defined as follows:

$$\begin{aligned} \mathit{null\text{-}p}(a) &:= a = 0 \\ \mathit{constr\text{-}p}_l^i(a, b) &:= b = \overline{l_i} + a * B^{|l_i|} \\ \mathit{constr\text{-}p}_r^i(a, b) &:= b = \overline{r_i} + a * B^{|r_i|} \\ \mathit{notequal\text{-}p}(a, b) &:= a \neq b \end{aligned}$$

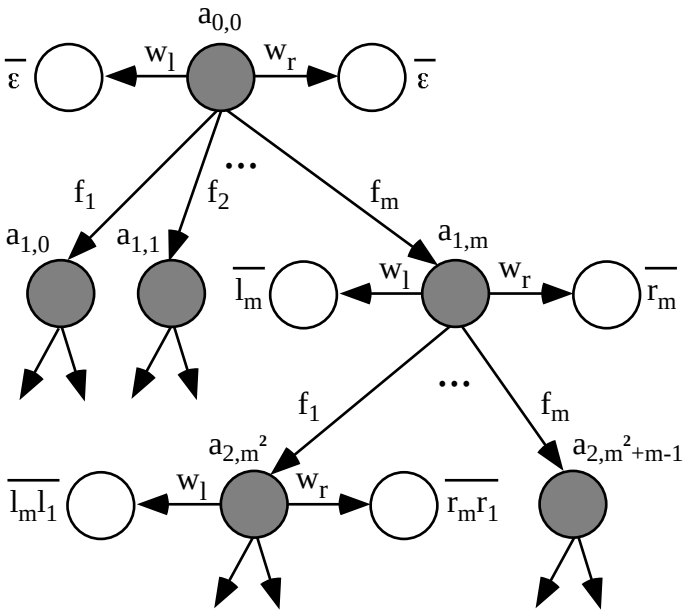


Figure 5.2: Search space of the Post Correspondence Problem encoded as a model of a concept $C(S)$.

The undecidability of $\mathcal{ALCNH}_{R^+}(\mathcal{D})$ is proven by showing that $C(S)$ is consistent iff the PCP S has no solution. Therefore, if the consistency of $C(S)$ could be decided, the algorithm could also be used to decide if a PCP S has a solution.

We first show that S has no solution if $C(S)$ is consistent. This can be easily seen by considering the definition of $C(S)$. If $C(S)$ is consistent there must exist an interpretation $\mathcal{I}$ with $C(S)^{\mathcal{I}} \neq \emptyset$. Figure 5.2 demonstrates that the interpretation encodes the (infinite) search space for a solution of S. However, since $C(S)$ is assumed to be consistent, $\forall R.\exists w_l, w_r.\mathit{notequal\text{-}p}$ holds. Therefore, none of the paths in the search space leads to a solution.

Now we prove that $C(S)$ is consistent if S has no solution. This direction is proven by defining an interpretation with $C(S)^{\mathcal{I}} \neq \emptyset$ for a PCP S for which it is known that no solution exists.

$$\Delta^{\mathcal{I}} = \{a_{ij} \mid i \geq 0, 1 \leq j < m^i\};$$
$$\forall i \geq 0, 0 \leq j < m^i :$$
$$f_1{}^{\mathcal{I}}(a_{ij}) = a_{i+1\ j*m},\ \ldots,\ f_m{}^{\mathcal{I}}(a_{ij}) = a_{i+1\ j*m+m-1},$$
$$w_l{}^{\mathcal{I}}(a_{ij}) = \overline{\phi_l(i,j)}, \quad w_r{}^{\mathcal{I}}(a_{ij}) = \overline{\phi_r(i,j)}$$

where ϕ_l and ϕ_r are two recursively defined concatenation functions (*concat* concate-

nates words and $\lfloor\rfloor$ denotes the $floor$ function):

$$\begin{aligned}
\phi_l(0,0) &= \epsilon \\
\phi_r(0,0) &= \epsilon \\
\phi_l(i,j) &= concat(\phi_l(i-1, \lfloor j/m \rfloor), l_{j+1-(m*\lfloor j/m \rfloor)}) \\
\phi_r(i,j) &= concat(\phi_r(i-1, \lfloor j/m \rfloor), r_{j+1-(m*\lfloor j/m \rfloor)}).
\end{aligned}$$

□

As we have discussed before, the undecidability proof for $\mathcal{ALCNH}_{R^+}(\mathcal{D})$ presented here follows the approach for showing the undecidability of $\mathcal{ALC}(\mathcal{D})$-trans presented in [Baader & Hanschke, 1992]. Furthermore, the idea to construct a concept $C(S)$ in such a way that it is satisfiable iff the PCP S has *no* solution has been taken from [Lutz & Möller, 1997; Lutz, 1998; Haarslev et al., 1998]. The basic idea of the undecidability proofs is to construct a transitive role in order to propagate a concept constraint to all individuals in the tree which encodes the search space of a PCP. In the undecidability proof for $\mathcal{ALCNH}_{R^+}(\mathcal{D})$ presented here, a similar effect is achieved by exploiting role hierarchies and transitive roles.[3]

Analyzing the model of the PCP it becomes clear that the undecidability is caused by the possibility to establish predicates for concrete domain objects that are referred to via features with different individuals on the left-hand side of the corresponding ABox assertions. The finite model property is lost in $\mathcal{ALCNH}_{R^+}(\mathcal{D})$. However, as long as a finite model is actually found by a calculus, this is no problem. So there might be some hope that "conditions" under which non-termination is "likely to occur" can be established. If these conditions are encountered, then the answer to the inference problem could be "unknown".

Rather than considering the quite complex PCP concept in detail, we discuss a simpler $\mathcal{ALCNH}_{R^+}(\mathcal{D})$ concept intended for describing lists of numbers.[4] As in the previous subsection, there are predicates established for concrete objects that are referred to by different individuals.

Let us assume, car is a feature cdr is an attribute and Rest is a transitive superrole of cdr. We also use the name cadr for the chain $\mathsf{cdr}\,\mathsf{car}$. Let P, $\mathsf{Q1}$ and $\mathsf{Q2}$ be elements of $\Phi_{\Re}$ (see Section 5.2) such that $\mathsf{P}(\mathsf{x},\mathsf{y}) := \mathsf{y} - \mathsf{x} = 1$, $\mathsf{Q1}(\mathsf{x}) := \mathsf{x} > 7$ and $\mathsf{Q2}(\mathsf{x}) := \mathsf{x} = 100$.

Example 1: $\exists\,\mathsf{car},\mathsf{cadr}\,.\,\mathsf{P} \sqcap \forall\,\mathsf{Rest}\,.\,(\exists\,\mathsf{car},\mathsf{cadr}\,.\,\mathsf{P})$

[3] Decidability problems with concrete domains and cyclic axioms are also discussed in [Buchheit et al., 1995].

[4] In a configuration context, for instance, a list of cylinders might be described.

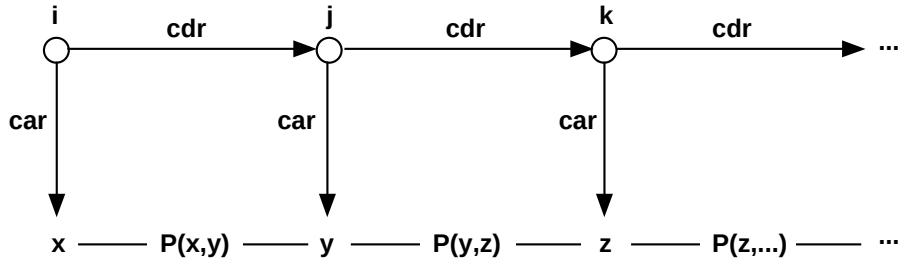


Figure 5.3: List of numbers greater than 7 decreasing by 1.

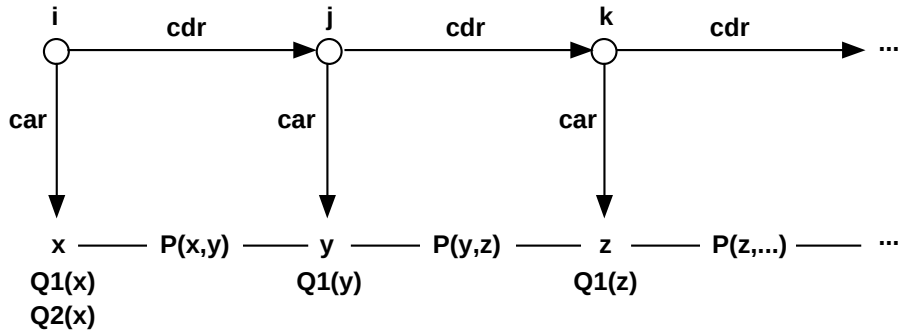


Figure 5.4: List of numbers greater than 7 decreasing by 1 starting at 100.

Figure 5.3 sketches a model for this concept (i, j and k are individuals and $\mathsf{x}, \mathsf{y}, \mathsf{z}$ are concrete objects). Since P is based on a total strict ordering, the model for the concept in Example 1 must be infinite.

Example 2: $\exists\,\mathsf{car}\,.\,\mathsf{Q1} \sqcap \exists\,\mathsf{car}\,.\,\mathsf{Q2}$
$\sqcap\, \exists\,\mathsf{car}, \mathsf{cadr}\,.\,\mathsf{P} \sqcap \forall\,\mathsf{Rest}\,.\,(\exists\,\mathsf{car}\,.\,\mathsf{Q1} \sqcap \exists\,\mathsf{car}, \mathsf{cadr}\,.\,\mathsf{P})$

Figure 5.4 shows an interpretation which is to be continued to the right in the expected way. Since x is equal to 100 it can easily be seen that this interpretation cannot be a model because it must be extended to the right until some 'successor' (filler of the role Rest) will be less than 7.

Example 3: $\forall\,\mathsf{cdr}\,.\,\bot \sqcup (\exists\,\mathsf{car}, \mathsf{cadr}\,.\,\mathsf{P} \sqcap \forall\,\mathsf{Rest}\,.\,(\forall\,\mathsf{cdr}\,.\,\bot \sqcup \exists\,\mathsf{car}, \mathsf{cadr}\,.\,\mathsf{P}))$

A model for this concept has the structure of the interpretation shown in Figure 5.3 but can be finite because there is no role filler for cdr required. Even the interpretation consisting only of one individual without fillers for cdr and car is a model. This interpretation represents an empty list.

From an application-oriented point of view, it is often not necessary to describe *infinite* lists. The concept in Example 3 captures that lists can be of *arbitrary but*

finite length.[5] Since there exists a finite model it might be possible to devise a calculus to compute a configuration based on an initial input ABox (cf. Figure 5.4). However, since the language is undecidable in general, a sound and complete (and terminating) calculus for deciding knowledge base consistency inevitably must return 'unknown' in some situations. We conjecture that it might be possible to detect these situations (i.e. guarantee termination) while preserving that both 'yes' and 'no' answers can be trusted. In the case of "linear" structures as discussed with the examples above it might be possible to integrate additional proof techniques involving the induction principle. In Example 1 and Example 2, "unknown" might be returned. Details of a calculus still have to be worked out.

5.5 Discussion

In this chapter the description logic $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$ has been introduced as an extension of $\mathcal{ALCNH}_{R^+}$. A tableaux calculus deciding the knowledge base consistency problem for $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$ has been presented. Soundness, completeness and termination has been shown. In addition, applications of the logic in the context of configuration problems have been sketched. The Cylinder example demonstrates that some requirements of a model-based configuration system are fulfilled by $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$. The calculus presented in this chapter can be used to solve "simple" configuration problems in which the configuration space can be described by an $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$ knowledge base (see [Cunis et al., 1991; Buchheit et al., 1995; Günter, 1995] for additional representation structures for solving configuration problems).

The DL system RACE will be extended with support for reasoning with concrete domains in the near future. The adaptation of important optimization techniques such as dependency-directed backtracking and model merging in the context of concrete domains is discussed in [Turhan & Haarslev, 2000; Turhan, 2000].

[5] It should be emphasized that, obviously, the concept of Example 3 is by no means equisatisfiable compared to the concept of Example 2.

Chapter 6

Spatiotemporal Terminological Reasoning

In the previous chapters it has been shown that expressive description logics can be developed such that many representation requirements in different application scenarios can be adequately fulfilled with formal inference systems. Besides the facilities offered by the base language $\mathcal{ALC}$, with $\mathcal{ALCNH}_{R^+}$ we have considered number restrictions, transitive roles and role hierarchies. With RACE a powerful DL inference system for this logic has been introduced. Furthermore, the language $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$ introduced in the previous chapter, extends $\mathcal{ALCNH}_{R^+}$ with facilities for reasoning about other domains such as the reals.

Although $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$ is an expressive description logic, some inferences required for dealing with natural phenomena cannot be adequately captured in $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$ (and other logics as well). For instance, spatial and temporal knowledge cannot be represented in an appropriate way in the context of terminological reasoning with description logics (see also Section 3.4). One approach might be to use role names to represent spatial relations such as 'inside' etc. Then, quantification over the objects which are inside a certain object are possible with value or exist restrictions. Although with $\mathcal{ALCNH}_{R^+}$ roles can be declared as transitive, this is not sufficient to capture, for instance, qualitative spatial inferences.

Let us consider an example where the well-known topological spatial relations from the RCC-8 theory [Cohn et al., 1997] are represented by role names (see Figure 6.1). As an ontological commitment, we assume that each abstract domain object is associated with its spatial representation via a feature has_area. Then, we consider terminological axioms for two different kinds of cities: city_1 and city_2.

$$\textbf{city\_1} \doteq \mathsf{city} \sqcap (\exists\, \mathsf{ntppi}\,.\,\mathsf{center}) \sqcap (\exists\, \mathsf{dc}\,.\,\mathsf{suburb})$$
$$\textbf{city\_2} \doteq \mathsf{city} \sqcap \exists\, \mathsf{ntppi}\,.\,(\mathsf{center} \sqcap \exists\, \mathsf{dc}\,.\,\mathsf{suburb})$$

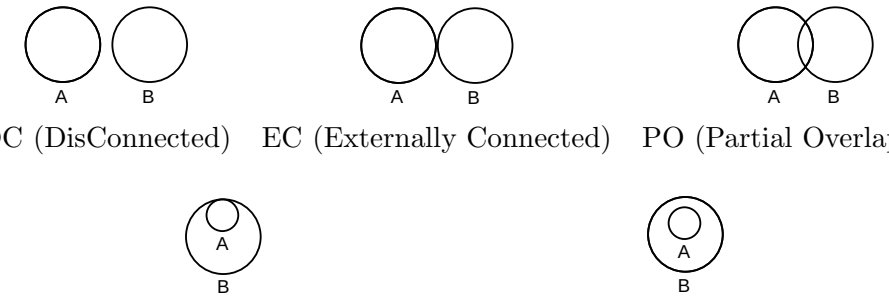


Figure 6.1: Elementary relations between two regions A and B. The inverses of TPP (TPPI) and NTPP (NTPPI) as well as the relation EQ (EQual) are not shown.

Both kinds of cities contain a center (topological relation ntppi, non-tangential proper part). In addition, in the former concept definition there exists a suburb which is disconnected (topological relation dc) from the city. In the second definition the existing suburb is disconnected only from the center (but not from the city as a whole). In Figure 6.2, models for the two concepts are given using a graph notation. Edges drawn with full lines correspond to existential quantification over roles in the concept definitions. Dashed lines between abstract objects represent predicates holding between the associated has_area features of the objects. If we require the suburb to be disconnected from the city, it is certainly disconnected from a region inside the city (in this case the center region). However, if a concept requires the existence of a suburb which is only disconnected from the center, then the suburb can be in any of the indicated relations to the city as a whole (see the lower parts in Figure 6.2). This set of possible topological relations in the example city_2 can easily be verified using an RCC-8 relation composition table (see e.g. [Cohn et al., 1997]). By considering Figure 6.2, we see that the concept city_2 should subsume the concept city_1 (the predicate dc_ec_po_tppi_ntppi represents a disjunction of spatial relations including dc).

However, since the semantics of topological relations is not represented when only role names (and defined roles) are used for modeling, the subsumption relation will not be detected. Even declaring roles as transitive does not suffice to achieve the necessary inferences.

We define another city concept as a specialization of city_2 with the additional restriction that all individuals which are spatially connected are no suburbs.

$$\textbf{city\_3} \doteq \mathsf{city\_2} \sqcap \forall\, \mathsf{spatially\_connected}\,.\,\neg\mathsf{suburb}$$

The role spatially_connected is to be interpreted as the disjunction of all topological

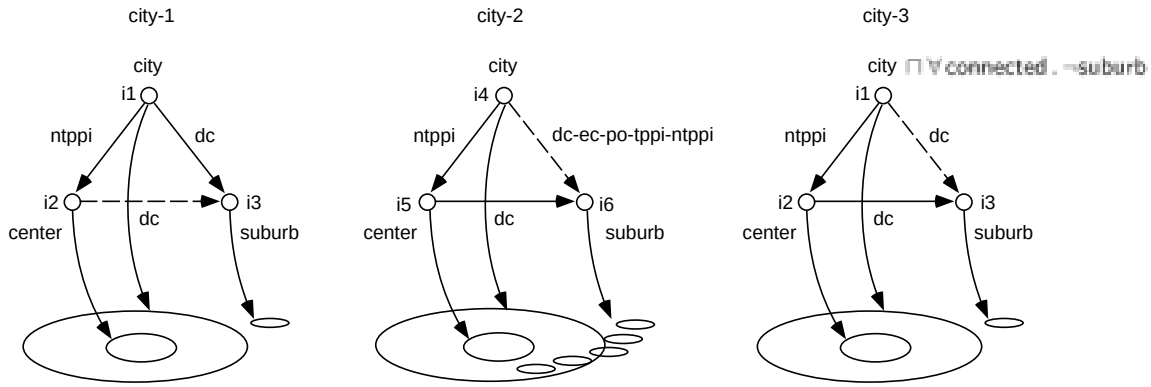


Figure 6.2: Models for concepts city_1 and city_2 with inferred role filler relationships (see text). In the lower part, examples for the spatial regions of the corresponding objects in the upper part are given. The regions are assumed to be associated with the corresponding abstract object with the feature has_area.

base relations except dc (disconnected). The concept city_3 is defined as a city_2 with an additional value restriction for the role spatially_connected. Although not directly apparent, careful thinking reveals that city_3 is more specific than city_1. Due to the value restriction for the role spatially_connected (see the concept definition of city_3) we can infer that only dc can hold between the city and the suburb. This and the additional restriction for the city (see Figure 6.2) are the reasons that city_3 is more specific than city_1. Again, the semantics of topological relations is not modeled by the representation formalism.

In order to formally integrate spatial and terminological reasoning, the description logic $\mathcal{ALCRP}(\mathcal{D})$ has been developed (see also [Haarslev et al., 1998; Lutz, 1998; Haarslev et al., 1999b]).

6.1 The Description Logic $\mathcal{ALCRP}(\mathcal{D})$

$\mathcal{ALCRP}(\mathcal{D})$ is a description logic with less expressive power than $\mathcal{ALCNH}_{R^+}$ regarding some features such as number restrictions, but is at the same time more powerful because of the incorporation of roles defined w.r.t. predicates of a concrete domain. Note that some notions (e.g. features are defined in a slightly different way as in the context of $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$).

6.1.1 The Concept Language of $\mathcal{ALCRP}(\mathcal{D})$

We first present the syntax and semantics of the language for specifying concept and role inclusions.

Definition 54 (Role Terms) Let R and F be disjoint sets of role and feature names, respectively. Any element of $R \cup F$ is an *atomic* role term. A composition of features (written $\mathsf{f_1 f_2 \ldots f_n}$) is called a feature chain. A simple feature can be considered as a feature chain of length 1. If $\mathsf{P} \in \Phi_{\mathcal{D}}$ is a predicate name with arity $n+m$ and $\mathsf{u_1}, \mathsf{u_2}, \ldots, \mathsf{u_n}$ as well as $\mathsf{v_1}, \mathsf{v_2}, \ldots, \mathsf{v_m}$ are feature chains, then the expression $\exists(\mathsf{u_1}, \ldots, \mathsf{u_n})(\mathsf{v_1}, \ldots, \mathsf{v_m}).\mathsf{P}$ (*role-forming predicate operator*) is a *complex* role term. Let S be a role name and let R be a role term. Then $\mathsf{S} \doteq \mathsf{R}$ is a *terminological axiom* or *role axiom*. This type of terminological axiom is also called *role introduction*. The role S is also called a *defined role*.

Using the definitions from above, we define the syntax of concept terms in $\mathcal{ALCRP}(\mathcal{D})$.

Definition 55 (Concept Terms) Let C be a set of concept names which is disjoint from R and F. Any element of C is a *concept term*. If C and D are concept terms, $\mathsf{R} \in R$ is an arbitrary role, $\mathsf{P} \in \Delta_{\mathcal{D}}$ is a predicate of the concrete domain, $\mathsf{u_i}$ is a feature chain, $n > 1$, then the following expressions are also concept terms:

- $\mathsf{C} \sqcap \mathsf{D}$ *(conjunction)*
- $\mathsf{C} \sqcup \mathsf{D}$ *(disjunction)*
- $\neg\mathsf{C}$ *(negation)*
- $\forall\, \mathsf{R}\,.\,\mathsf{C}$ *(concept value restriction)*
- $\exists\, \mathsf{R}\,.\,\mathsf{C}$ *(concept exists restriction)*
- $\exists\, \mathsf{u_1}, \ldots, \mathsf{u_n}\,.\,\mathsf{P}$ *(predicate exists restriction).*

A concept term may be put in parentheses. $\top$ ($\bot$) is considered as an abbreviation for $\mathsf{C} \sqcup \neg\mathsf{C}$ ($\mathsf{C} \sqcap \neg\mathsf{C}$).

For $\mathcal{ALCRP}(\mathcal{D})$ it would have been possible to also introduce a role box as a set of role axioms. However, because there is no need to impose special conditions on role boxes (in contrast to $\mathcal{ALCNH}_{R^+}$, see Section 2.1), in $\mathcal{ALCRP}(\mathcal{D})$ the role axioms are just part of the TBox.

Definition 56 (TBox, Introduction Axioms) Let A be a concept name and let D be a concept term. Then $\mathsf{A} \doteq \mathsf{D}$ and $\mathsf{A} \sqsubseteq \mathsf{D}$ are terminological axioms as well. A finite set of terminological axioms $\mathcal{T}$ is called a *terminology* or *TBox* if the left-hand sides of all terminological axioms in $\mathcal{T}$ are unique and, furthermore, all concept definitions are acyclic. The axioms $\mathsf{A} \sqsubseteq \mathsf{D}$ in a TBox are also called *concept introduction axioms*.

The next definition provides a model-theoretic semantics for the language introduced above. Let $\mathcal{D} = (\Delta_{\mathcal{D}}, \Phi_{\mathcal{D}})$ be a concrete domain.

Definition 57 (Semantics) An *interpretation* $\mathcal{I}_\mathcal{D} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})_\mathcal{D}$ consists of a set $\Delta^\mathcal{I}$ (the abstract domain), a set $\Delta^\mathcal{D}$ (the domain of the 'concrete domain' $\mathcal{D}$) and an interpretation function $\cdot^\mathcal{I}$. The interpretation function $\cdot^\mathcal{I}$ maps each concept name C to a subset $\mathsf{C}^\mathcal{I}$ of $\Delta^\mathcal{I}$, each role name R from R to a subset $\mathsf{R}^\mathcal{I}$ of $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$, each feature f from F to a partial function $\mathsf{f}^\mathcal{I}$ from $\Delta^\mathcal{I}$ to $\Delta^\mathcal{I} \cup \Delta^\mathcal{D}$,[1] and each predicate name P from $\Phi_\mathcal{D}$ with arity n to a subset $\mathsf{P}^\mathcal{I}$ of $\Delta^n_\mathcal{D}$. If $\mathsf{u} = \mathsf{f_1} \cdots \mathsf{f_n}$ is a feature chain, then $\mathsf{u}^\mathcal{I}$ denotes the composition $\mathsf{f_1}^\mathcal{I} \circ \ldots \circ \mathsf{f_n}^\mathcal{I}$ of partial functions $\mathsf{f_1}^\mathcal{I}, \ldots, \mathsf{f_n}^\mathcal{I}$. Let the symbols C, D be concept expressions, R, S be role names, $\mathsf{u_1}, \ldots, \mathsf{u_n}$ be feature chains and let P be a predicate name. Then, the interpretation function can be extended to arbitrary concept and role terms as follows:

$$
\begin{aligned}
(\mathsf{C} \sqcap \mathsf{D})^\mathcal{I} &:= \mathsf{C}^\mathcal{I} \cap \mathsf{D}^\mathcal{I} \\
(\mathsf{C} \sqcup \mathsf{D})^\mathcal{I} &:= \mathsf{C}^\mathcal{I} \cup \mathsf{D}^\mathcal{I} \\
(\neg \mathsf{C})^\mathcal{I} &:= \Delta^\mathcal{I} \setminus \mathsf{C}^\mathcal{I} \\
(\exists\, \mathsf{R}\,.\,\mathsf{C})^\mathcal{I} &:= \{a \in \Delta^\mathcal{I} \mid \exists\, b \in \Delta^\mathcal{I} : (a,b) \in \mathsf{R}^\mathcal{I}, b \in \mathsf{C}^\mathcal{I}\} \\
(\forall\, \mathsf{R}\,.\,\mathsf{C})^\mathcal{I} &:= \{a \in \Delta^\mathcal{I} \mid \forall\, b : (a,b) \in \mathsf{R}^\mathcal{I} \Rightarrow b \in \mathsf{C}^\mathcal{I}\} \\
(\exists\, \mathsf{u_1}, \ldots, \mathsf{u_n}\,.\,\mathsf{P})^\mathcal{I} &:= \{a \in \Delta^\mathcal{I} \mid \exists x_1, \ldots, x_n \in \Delta^\mathcal{D} : \\
&\qquad (a, x_1) \in \mathsf{u_1}^\mathcal{I}, \ldots, (a, x_n) \in \mathsf{u_n}^\mathcal{I}, \\
&\qquad (x_1, \ldots, x_n) \in \mathsf{P}^\mathcal{I}\} \\
(\exists\, (\mathsf{u_1}, \ldots, \mathsf{u_n})(\mathsf{v_1}, \ldots, \mathsf{v_m})\,.\,\mathsf{P})^\mathcal{I} &:= \{(a,b) \in \Delta^\mathcal{I} \times \Delta^\mathcal{I} \mid \\
&\qquad \exists x_1, \ldots, x_n, y_1, \ldots, y_m \in \Delta^\mathcal{D} : \\
&\qquad\quad (a, x_1) \in \mathsf{u_1}^\mathcal{I}, \ldots, (a, x_n) \in \mathsf{u_n}^\mathcal{I}, \\
&\qquad\quad (b, y_1) \in \mathsf{v_1}^\mathcal{I}, \ldots, (b, y_m) \in \mathsf{v_m}^\mathcal{I}, \\
&\qquad\quad (x_1, \ldots, x_n, y_1, \ldots, y_m) \in \mathsf{P}^\mathcal{I}\}
\end{aligned}
$$

An interpretation $\mathcal{I}$ is a *model* of a TBox $\mathcal{T}$ iff $\mathcal{I}$ satisfies $\mathsf{A}^\mathcal{I} \subseteq \mathsf{D}^\mathcal{I}$ ($\mathsf{A}^\mathcal{I} = \mathsf{D}^\mathcal{I}$) for all concept introduction axioms $\mathsf{A} \sqsubseteq \mathsf{D}$ ($\mathsf{A} \doteq \mathsf{D}$) in $\mathcal{T}$ and $\mathsf{S}^\mathcal{I} = \mathsf{R}^\mathcal{I}$ for all terminological axioms $\mathsf{S} \doteq \mathsf{R}$ (role introductions) in $\mathcal{T}$.

6.1.2 The Assertional Language of $\mathcal{ALCRP}(\mathcal{D})$

In the following, the language for representing knowledge about individuals is introduced. An *ABox* $\mathcal{A}$ is a finite set of assertional axioms which are defined as usual:

Definition 58 (ABox Assertions) Let O be a set of individual names. Furthermore, let X be a set of names for concrete objects ($X \cap O = \emptyset$). If C is a concept

[1] Note the difference between the semantics for features in $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$ and $\mathcal{ALCRP}(\mathcal{D})$.

term, R a role name, $\mathsf{a}, \mathsf{b} \in O$ are individual names and $\mathsf{x}, \mathsf{x}_1, \ldots, \mathsf{x}_n \in X$ are names for concrete objects, then the following expressions are *assertional axioms*:

- $\mathsf{a} : \mathsf{C}$ *(concept assertion)*,
- $(\mathsf{a}, \mathsf{b}) : \mathsf{R}$ *(role assertion)*,
- $(\mathsf{a}, \mathsf{x}) : \mathsf{f}$ *(concrete domain feature assertion)*,
- $(\mathsf{x}_1, \ldots, \mathsf{x}_n) : \mathsf{P}$ *(concrete domain predicate assertion)*.

The interpretation function $\cdot^{\mathcal{I}}$ of the interpretation $\mathcal{I}$ for the concept language can be extended to the assertional language by additionally mapping every individual name from O to a single element $\Delta^{\mathcal{I}}$ (the unique name assumption does not necessarily hold). Concrete objects from X are mapped to elements of $\Delta^{\mathcal{D}}$.

An interpretation satisfies an assertional axiom $\mathsf{a} : \mathsf{C}$ iff $\mathsf{a}^{\mathcal{I}} \in \mathsf{C}^{\mathcal{I}}$, $(\mathsf{a}, \mathsf{b}) : \mathsf{R}$ iff $(\mathsf{a}^{\mathcal{I}}, \mathsf{b}^{\mathcal{I}}) \in \mathsf{R}^{\mathcal{I}}$, $(\mathsf{a}, \mathsf{x}) : \mathsf{f}$ iff $(\mathsf{a}^{\mathcal{I}}, \mathsf{x}^{\mathcal{I}}) \in \mathsf{f}^{\mathcal{I}}$ and $(\mathsf{x}_1, \ldots, \mathsf{x}_n) : \mathsf{P}$ iff $({\mathsf{x}_1}^{\mathcal{I}}, \ldots, {\mathsf{x}_n}^{\mathcal{I}}) \in \mathsf{P}^{\mathcal{I}}$.

An interpretation $\mathcal{I}$ is a *model* of an ABox $\mathcal{A}$ w.r.t. a TBox $\mathcal{T}$ iff $\mathcal{I}$ is a model of $\mathcal{T}$ and furthermore satisfies all assertional axioms in $\mathcal{A}$.

6.2 Decidability and Undecidability Results

Based on the notion of a model, the inference problems for $\mathcal{ALCRP}(\mathcal{D})$ are defined in a similar way as for $\mathcal{ALCNH}_{R^+}$ (see Section 2.3). As mentioned above, for brevity, no role box is considered in the context of $\mathcal{ALCRP}(\mathcal{D})$.

In [Lutz & Möller, 1997] as well as in [Haarslev et al., 1998] it is shown that, unfortunately, the inference problem of checking the consistency of concepts (and ABoxes) in the "generic" language $\mathcal{ALCRP}(\mathcal{D})$ is undecidable in general. However, in [Haarslev et al., 1999b] a restricted variant of $\mathcal{ALCRP}(\mathcal{D})$ is described that is indeed decidable if only (syntactically) *restricted* concept terms are used. Thus, the above-mentioned $\mathcal{ALCRP}(\mathcal{D})$ inference problems can be decided if only restricted $\mathcal{ALCRP}(\mathcal{D})$ concept terms are admitted.

Definition 59 A concept term X is called *restricted* w.r.t. a TBox $\mathcal{T}$ iff its equivalent X' which is unfolded w.r.t. $\mathcal{T}$ and in negation normal form fulfills the following conditions:[2]

(1) For any subconcept term C of X' that is of the form $\forall\, \mathsf{R}_1 \,.\, \mathsf{D}$ $(\exists\, \mathsf{R}_1 \,.\, \mathsf{D})$ where R_1 is a complex role term, D does not contain any terms of the form $\exists\, \mathsf{R}_2 \,.\, \mathsf{E}$ $(\forall\, \mathsf{R}_2 \,.\, \mathsf{E})$ where R_2 is also a complex role term.

(2) For any subconcept term C of X' that is of the form $\forall\, \mathsf{R} \,.\, \mathsf{D}$ or $\exists\, \mathsf{R} \,.\, \mathsf{D}$ where R is a complex role term, D contains only predicate exists restrictions that (i) quantify over

[2]For technical reasons, we assume that a concept term is a subconcept term of itself.

attribute chains of length 1 and (ii) are not contained inside any value and exists restrictions that are also contained in D.

A terminology is called restricted iff all concept terms appearing on the right-hand side of terminological axioms in $\mathcal{T}$ are restricted w.r.t. $\mathcal{T}$. An ABox $\mathcal{A}$ is called restricted w.r.t. a TBox $\mathcal{T}$ iff $\mathcal{T}$ is restricted and all concept terms used in $\mathcal{A}$ are restricted w.r.t. the terminology $\mathcal{T}$.

Theorem 60 The ABox consistency problem for restricted $\mathcal{ALCRP}(\mathcal{D})$ concept terms is decidable if $\mathcal{D}$ is an admissible concrete domain.

The proof is given in [Haarslev et al., 1999b].

Given two restricted concept terms, their subsumption relationship can be tested. This follows from the following proposition.

Proposition 61 The set of restricted $\mathcal{ALCRP}(\mathcal{D})$ concept terms is closed under negation.

For the proof see [Haarslev et al., 1999b].

6.3 Spatioterminological Reasoning

The examples discussed in this section employ reasoning with a concrete domain which can represent spatial relations between domain objects. Due to its widespread use, we focus on topological relations known from the RCC-8 theory [Randell et al., 1992].

6.3.1 $\mathcal{ALCRP}(\mathcal{RCC})$

Before presenting examples we briefly introduce the concrete domain $\mathcal{RCC}$. We will consider specific spatial objects whose spatial representations are given as polygons. It is shown that $\mathcal{RCC}$ provides predicates which can be used to describe qualitative spatial RCC-8 relations as roles between spatial objects. The relations are depicted in Figure 6.1.

Definition 62 The concrete domain $\mathcal{RCC}$ is defined w.r.t. the topological space $\langle \mathbb{R}^2, 2^{\mathbb{R}^2} \rangle$. The domain $\Delta^{\mathcal{RCC}}$ contains all non-empty, regular closed subsets of $\mathbb{R}^2$, which are called *regions* for short. The set of predicate names is defined as follows:

- A unary *concrete_domain_top* predicate $\mathsf{is\text{-}region}$ with $\mathsf{is\text{-}region}^{\mathcal{RCC}} = \Delta^{\mathcal{RCC}}$ and its negation $\mathsf{is\text{-}no\text{-}region}$ with $\mathsf{is\text{-}no\text{-}region}^{\mathcal{RCC}} = \emptyset$.

- The 8 basic predicates dc, ec, po, tpp, ntpp, tppi, ntppi and eq correspond to the RCC-8 relations (see Figure 6.1). I would like to refer to [Haarslev et al., 1999b] for a formal definition of the semantics.

- In order to name disjunctions of base relations, we need additional predicates. Unique names for these "disjunction predicates" are enforced by imposing the following canonical order on the basic predicate names: dc, ec, po, tpp, ntpp, tppi, ntppi, eq. For each sequence $p_1, \ldots, p_n$ of basic predicates in canonical order ($n \geq 2$), an additional predicate of arity 2 is defined. The predicate has the name $\mathsf{p_1}\text{-}\cdots\text{-}\mathsf{p_n}$ and we have $(r_1, r_2) \in \mathsf{p_1}\text{-}\cdots\text{-}\mathsf{p_n}^{\mathcal{RCC}}$ iff $(r_1, r_2) \in \mathsf{p_1}^{\mathcal{RCC}}$ or ... or $(r_1, r_2) \in \mathsf{p_n}^{\mathcal{RCC}}$. The predicate dc-ec-po-tpp-ntpp-tppi-ntppi-eq is also called spatially-related.

- A binary predicate inconsistent-relation with $\mathsf{inconsistent\text{-}relation}^{\mathcal{RCC}} = \emptyset$ is the negation of spatially-related.

Proposition 63 The concrete domain $\mathcal{RCC}$ is admissible.

This is proven in [Haarslev et al., 1999b].

Based on the results presented in [Renz, 1998] we can conclude that there exists always a model whose individuals are polygons which are not necessarily internally connected. Although in some specific contexts this might be disturbing, for many applications, internal connectedness is not required. For instance, countries can have islands which are not connected to the main area of the country etc.

6.3.2 Reasoning with $\mathcal{ALCRP}(\mathcal{RCC})$: A GIS Application

The city examples from the introduction of this chapter demonstrate the importance of spatioterminological TBox reasoning. With the following role axioms the above-mentioned implicit subsumption relationships between the different city concepts do indeed follow.

$$\textbf{ntppi} \doteq \exists\,(\mathsf{has\_area})(\mathsf{has\_area})\,.\,\mathsf{ntppi}$$
$$\textbf{dc} \doteq \exists\,(\mathsf{has\_area})(\mathsf{has\_area})\,.\,\mathsf{dc}$$
$$\textbf{spatially\_connected} \doteq \exists(\mathsf{has\_area})(\mathsf{has\_area}).\mathsf{ec\text{-}po\text{-}tpp\text{-}ntpp\text{-}tppi\text{-}ntppi\text{-}eq}$$

With the concrete domain $\mathcal{RCC}$ introduced above, we can now turn to another example where the expressive power of $\mathcal{ALCRP}(\mathcal{D})$ is demonstrated in the context of spatioterminological ABox reasoning. First, we consider a set of additional role

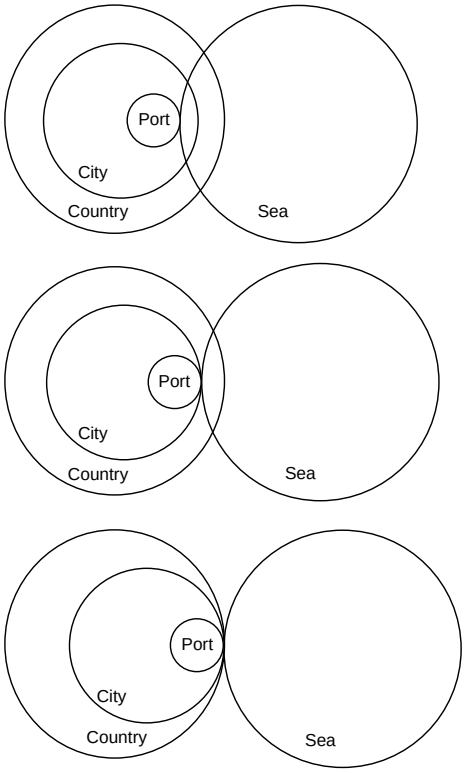


Figure 6.3: Candidate configurations for ABox individuals (see text).

introduction axioms.

$$\textbf{related} \doteq \exists(\mathsf{has\_area})(\mathsf{has\_area}).\mathsf{spatially\text{-}related}$$
$$\textbf{inside} \doteq \exists(\mathsf{has\_area})(\mathsf{has\_area}).\mathsf{tpp\text{-}ntpp}$$
$$\textbf{inside\_i} \doteq \exists(\mathsf{has\_area})(\mathsf{has\_area}).\mathsf{tppi\text{-}ntppi}$$
$$\textbf{touching} \doteq \exists(\mathsf{has\_area})(\mathsf{has\_area}).\mathsf{ec}$$
$$\textbf{overlapping} \doteq \exists(\mathsf{has\_area})(\mathsf{has\_area}).\mathsf{po\text{-}tpp\text{-}ntpp\text{-}tppi\text{-}ntppi\text{-}eq}$$

The following concept introduction axioms constitute the TBox of our example for reasoning in $\mathcal{ALCRP}(\mathcal{RCC})$.

$$\textbf{coastal\_city} \doteq \mathsf{city} \sqcap \exists\, \mathsf{touching}\,.\,\mathsf{sea}$$
$$\textbf{country} \sqsubseteq \forall\, \mathsf{overlapping}\,.\,\neg\mathsf{sea}$$
$$\textbf{sea} \sqsubseteq \mathsf{ocean}$$

A coastal_city is defined as a city which is touched by a sea (in the sense of ocean). Furthermore, the second axiom enforces that countries do not overlap with seas.

The inferential power of $\mathcal{ALCRP}(\mathcal{RCC})$ is further explained with an instance problem concerning the following ABox.

$$\begin{aligned} \mathsf{loc\_1} &: \mathsf{city} \\ \mathsf{country\_1} &: \mathsf{country} \\ \mathsf{port\_1} &: \mathsf{port} \\ \mathsf{sea\_1} &: \mathsf{sea} \\ (\mathsf{loc\_1}, \mathsf{country\_1}) &: \mathsf{inside} \\ (\mathsf{loc\_1}, \mathsf{port\_1}) &: \mathsf{inside\_i} \\ (\mathsf{port\_1}, \mathsf{sea\_1}) &: \mathsf{touching} \end{aligned}$$

The TBox and the ABox of the example contain only restricted concept terms. Therefore the inference problems are decidable. We consider the instance problem $\mathsf{instance?}(\mathsf{loc\_1}, \mathsf{coastal\_city})$ posed as a query to the description logic system. As indicated above (see Section 2.3), the answer of the query is determined by the test whether the ABox becomes inconsistent if the assertion $\mathsf{loc\_1} : \neg\mathsf{coastal\_city}$ is added).

Possible spatial configurations based on ABox information are shown in Figure 6.3. Considering the definition of inside, the topological relation between the city loc_1 and the country country_1 is either tpp (tangential proper part) or ntpp (non-tangential proper part). The basic relations between the city and the port are tppi or ntppi (i for inverse), i.e. the port port_1 is a tangential or a non-tangential proper part of the city. Since the port touches the sea sea_1 (relation ec, externally connected) and, due to the second terminological axiom, a country and a sea cannot overlap (base relations po, tppi, ntppi or eq) only the third configuration in Figure 6.3 leads to a consistent scenario. However, since, due to the query, it is claimed that $\mathsf{loc\_1} : \neg\mathsf{coastal\_city}$ holds, there must not be a sea touching the city (see the terminological axiom for coastal_city). Hence, the ABox is inconsistent and the answer to the query $\mathsf{instance?}(\mathsf{loc\_1}, \mathsf{coastal\_city})$ is 'yes'.

6.4 Spatiotemporal Terminological Reasoning

Considering the general mechanism for integrating concrete domains, it becomes clear that another instance of $\mathcal{ALCRP}(\mathcal{D})$ can deal with qualitative temporal relations between time intervals according to [Allen, 1983]. The idea is to define a concrete domain $\mathcal{ALLEN}$ for modeling constraints for time intervals with binary predicates representing Allen's Interval Algebra (before, after, meets, met_by, overlaps, overlapped_by, during, contains, starts, started_by, finishes, finished_by, equal and disjunctions of these basic predicates). Constraint satisfaction algorithms known from the literature (see e.g. [Gerevini, 1997] for an overview) can then be employed to check

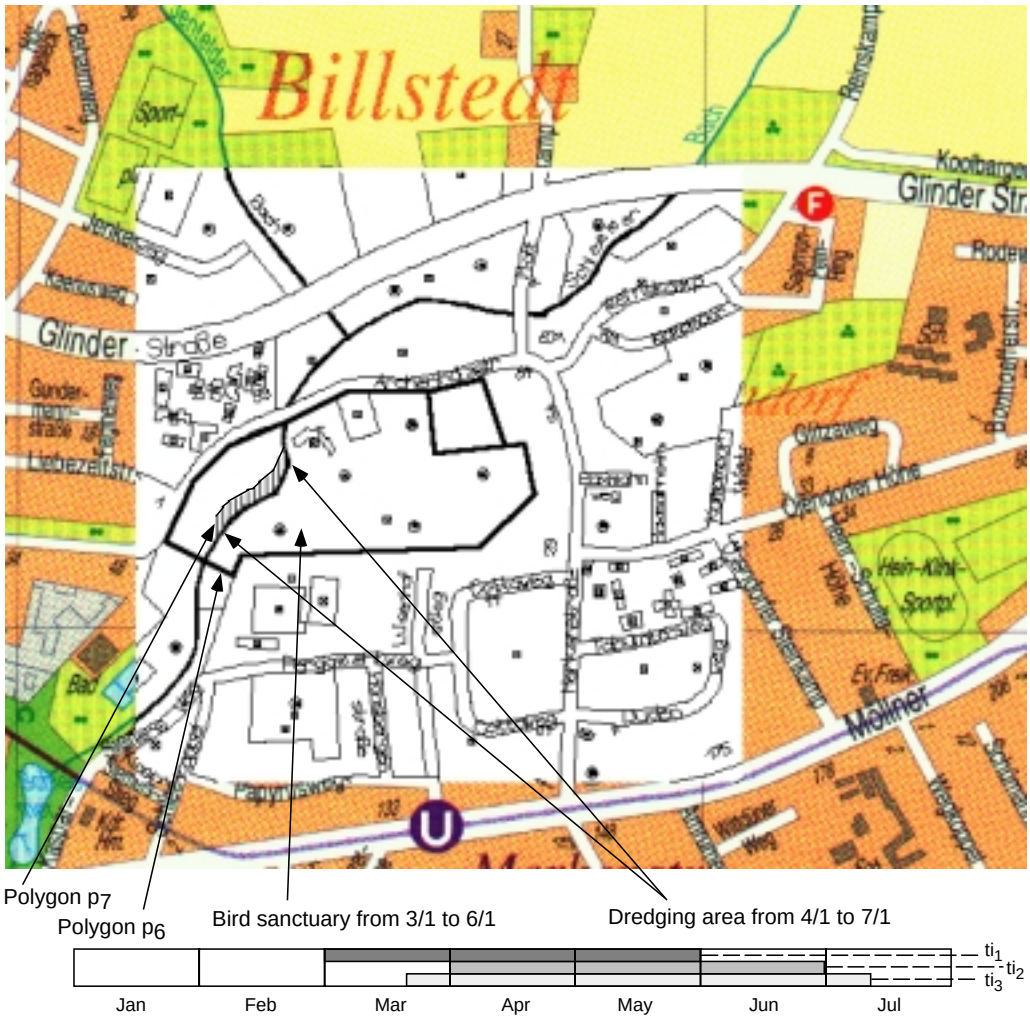


Figure 6.4: A clip from a city map (see text).

the satisfiability of conjunctions of predicates. Furthermore, in [Baader & Hanschke, 1991b] it is shown that, from a technical perspective, any two disjoint admissible concrete domains can be combined to form a single admissible concrete domain (the combination operator is called $\oplus$). An extension $\oplus'$ of $\oplus$ concerning roles based on predicates as in $\mathcal{ALCRP}(\mathcal{D})$ is discussed in [Haarslev et al., 1999b]. The combination $\mathcal{RCC} \oplus' \mathcal{ALLEN}$ also defines predicates as conjunctions of predicates from the component concrete domains, i.e. the spatial and the temporal concrete domain.

The use of the temporal domain in combination with the spatial domain will be illustrated with an example from the GIS application introduced in Figure 6.4. We assume that, in the GIS database, a certain area is defined to be a bird sanctuary from the beginning of March to the end of May (see the dark-gray bar in Figure 6.4). Now,

as part of a planning scenario, let us assume a hypothetical dredging operation is to be scheduled from April to June inclusive (middle-gray bar). In Figure 6.4 the affected area of the creek "Schleemer Bach" is indicated with a hatched region. In addition, we assume that dredging a creek involves handling trucks and dredging machinery. Thus, we assume the existence of a so-called "support area" touching the real dredging area in the creek (the topological relation is ec). Furthermore, background knowledge should indicate that the "support process" starts earlier and lasts longer than the proper dredging operation because the dredging machinery has to be installed and deinstalled (see the light-gray bar in Figure 6.4 which is used as an example here). Obviously, as the reader might expect, the spatiotemporal constraints, the conceptual knowledge and the knowledge about individuals in this example suggest that dredging in an "active" bird sanctuary should not be sanctioned by the planning module of the GIS. The question is: How can this planning problem using GIS facilities be solved by knowledge representation techniques and logical inferences? In the following we describe a solution with the description logic $\mathcal{ALCRP}(\mathcal{RCC}\oplus'\mathcal{ALLEN})$. The planning problem is represented as a knowledge base consistency problem. The TBox contains the background knowledge of the domain and the ABox represents supposed spatial and temporal constraints for the bird sanctuary and for a certain dredging operation.

Both domain objects, the bird sanctuary and the dredging operation, are represented as ABox objects. The idea is to show that the corresponding ABox can be proven to be inconsistent given the constraints modeling the knowledge informally introduced in the previous section.

As an ontological decision, let us assume that temporal intervals are associated with individuals via the feature has_duration. We can define a spatiotemporal_process as a process for which an interval and a region exist as fillers for the corresponding features. The predicate is_interval is assumed to check membership in the corresponding concrete domain (see the admissibility criterion in Definition 32).

spatiotemporal_process $\doteq$ process $\sqcap \exists$ has_duration . is_interval $\sqcap \exists$ has_area . is_region

noisy_process $\sqsubseteq$ process

dredging_support_process $\sqsubseteq$ spatiotemporal_process $\sqcap$ noisy_process

In addition, several auxiliary concepts are introduced to capture the noisiness of a dredging support process. These concepts are only partially defined with inclusion axioms. The concept spatiotemporal_process is used in subsequent terminological inclusion axioms for more specific spatiotemporal processes. The interaction of space and time in the example can be represented by introducing two roles ec_during and ntppi_overlaps which relate fillers of corresponding has_area and has_duration fea-

tures.

$$\textbf{ec-during} \doteq \exists\,(\mathsf{has\_area}, \mathsf{has\_duration})(\mathsf{has\_area}, \mathsf{has\_duration})\,.\,\mathsf{ec\text{-}during}$$
$$\textbf{ntppi-overlaps} \doteq \exists\,(\mathsf{has\_area}, \mathsf{has\_duration})(\mathsf{has\_area}, \mathsf{has\_duration})\,.\,\mathsf{ntppi\_overlaps}$$
$$\textbf{connected-ends-during-overlaps} \doteq$$
$$\exists\,(\mathsf{has\_area}, \mathsf{has\_duration})(\mathsf{has\_area}, \mathsf{has\_duration})\,.\,\mathsf{connected\text{-}ends\text{-}during\text{-}overlaps}$$

The predicate ec-during ensures that (i) the constraint ec is imposed on the fillers of the has_area features and (ii) the constraint during describes a relation between the fillers of the has_duration features. With ntppi-overlaps the corresponding constraints are established between the has_area and has_duration features, respectively. We also define a role connected-ends-during-overlaps combining spatial connectedness and generalized temporal overlapping. Thus, the predicates from both domains simultaneously hold for a pair of abstract individuals. Now, we can (partially) define two concepts dredging_process and bird_sanctuary.

$$\textbf{dredging\_process} \sqsubseteq$$
$$\mathsf{spatiotemporal\_process} \sqcap \exists\,\mathsf{ec\text{-}during}\,.\,\mathsf{dredging\_support\_process}$$
$$\textbf{bird\_sanctuary} \sqsubseteq$$
$$\mathsf{spatiotemporal\_process} \sqcap \neg\mathsf{noisy\_process} \sqcap$$
$$\forall\,\mathsf{connected\text{-}ends\text{-}during\text{-}overlaps}\,.\,\forall\,\mathsf{connected}\,.\,\neg\mathsf{noisy\_process}$$

The spatiotemporal role ec-during is used to relate the dredging_support_process in the appropriate way. A bird sanctuary is modeled as a non-noisy spatiotemporal_process with all fillers of the defined spatiotemporal role connected-ends-during-overlaps being objects which, in turn, are connected to non-noisy processes. The idea behind this definition is that temporally overlapping connected objects must not be spatially connected to noisy processes because this would cause an inconsistency. The $\mathcal{ALCRP}(\mathcal{D})$ restrictedness criterion is fulfilled for this terminology.

In order to represent GIS information concerning the bird sanctuary we simply add the following ABox axiom for an individual i1 which is assumed here to represent the available information about a particular bird sanctuary stored in the GIS. As a hypothesis for the dredging process, additional ABox axioms for the individual i2 and its spatiotemporal relationship with i1 (see also Figure 6.4) are used.[3]

$$\mathcal{A}_0 := \big\{\ \mathsf{i1} : \mathsf{bird\_sanctuary},\ \mathsf{i2} : \mathsf{dredging\_process},\ (\mathsf{i1}, \mathsf{i2}) : \mathsf{ntppi\text{-}overlaps}\big\}$$

[3]We neglect the technical details about the GIS implementation and its relation to the ABox of the DL system.

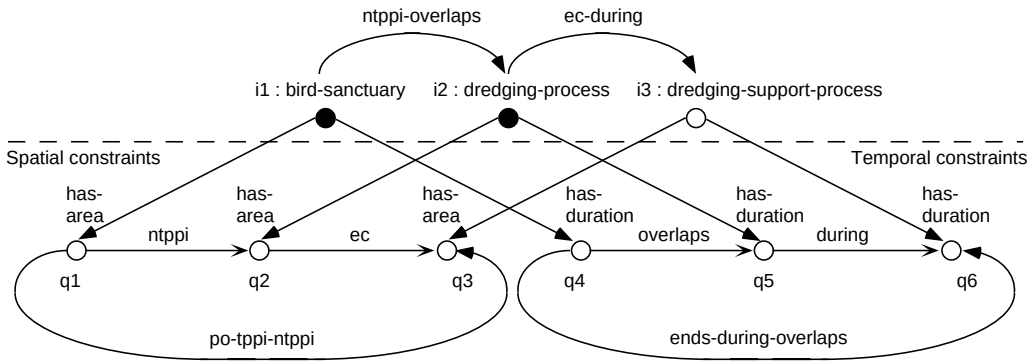


Figure 6.5: ABox for testing the satisfiability of the dredging process query with implicit spatial and temporal information shown in two constraint nets (see text).

The ABox $\mathcal{A}_0$ is checked for consistency. In Figure 6.5 a graphical representation of the ABox $\mathcal{A}_0$ is shown. Due to the constraints given in ABox $\mathcal{A}_0$, an individual i3 is generated. For this individual, the constraint dredging_support_process is asserted. Furthermore, additional constraints resulting from the given spatiotemporal relationship between i1 and i2 are derived (see the role ec-during and the spatial and temporal constraints in Figure 6.4). The filler of i1 for connected-ends-during-verlaps is i2 in this ABox (see Figure 6.4 and 6.5). Due to the value restriction for this role in the concept bird_sanctuary, new constraints are added to i2. The value restriction over connected is asserted. The fillers of the connected role with respect to i2 are i1, i3 and i2 itself.[4] Let us consider the individual i3. The constraint $\neg$noisy_process is added to i3. However, this causes a clash because i3 is a dredging_support_process which is a noisy_process by definition. Thus, due to the spatiotemporal constraints, there is no way to find a consistent tableau and, therefore, ABox $\mathcal{A}_0$ is inconsistent. The reader can easily verify that the existence of i3 (the dredging_support_process) is indeed required to make the ABox inconsistent because, in our example, the dredging process itself is not assumed to be a noisy process. If this were the case, there would be another inconsistency concerning i2, of course.

The example shows that the interaction between spatiotemporal and conceptual knowledge is very important for GIS systems. The inconsistency of the ABox $\mathcal{A}_0$ can be interpreted by the application system in such a way that the dredging operation should better be planned in another period of the year.[5] Note that ABox reasoning cannot easily be replaced by model checking because the implicit processes (see the exists constraint in the definition of dredging_process) are not given as part of the input to the GIS system. For the application system, the knowledge about a

[4]The predicate of the role connected subsumes the predicates ntppi, ec and po-tppi-ntppi.

[5]It would be attractive to compute alternatives for the has_duration intervals but this is beyond the scope of this example.

necessary dredging_support_process need not be directly available. Furthermore, in $\mathcal{ALCRP}(\mathcal{D})$ we have seen that with defined roles, the interaction between different concrete domains provides a powerful modeling technique which is neither available in $\mathcal{ALC}(\mathcal{D})$ nor in $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$.

In the example ABox $\mathcal{A}_0$ we have explicitly added the spatiotemporal constraint ntppi-overlaps between i1, the bird_sanctuary, and the dredging_process i2. This is required because the corresponding ABox assertion cannot be inferred. In principle, we have to add similar constraints for every pair of spatiotemporal processes. So, could we find a way to avoid this? In Figure 6.4 "explicit" regions and time intervals are indicated. We could use the concept-forming predicate operator together with additional predicates from an extended concrete domain in order to represent quantitative knowledge about regions and time intervals with explicit coordinates and time points, respectively. However, the problem of combining metric and topological constraints has to be investigated in future research (but see [Haarslev et al., 1999b]).

6.5 Spatioterminological Default Reasoning

In the following we investigate a Reiter-based approach [Reiter, 1980] to terminological default reasoning about spatial information. Originally, a default rule has the form

$$\frac{\alpha : \beta_1, \beta_2, \ldots, \beta_n}{\gamma}$$

(also written $\alpha : \beta_1, \beta_2, \ldots, \beta_n \ / \ \gamma$), where α, β_i and γ are FOPL formulae. α is called the *precondition* of the rule, the β_i terms are called *justifications,* and γ is the *consequent*. Intuitively the idea behind default reasoning is the following: starting with a world description A of what is known to be true, default rules can be applied such that they augment A by default rule conclusions γ to yield a *set of beliefs*. A default can be applied, i.e. its conclusion γ can be added to the set of current beliefs iff α is entailed by this set, each formula β_i is consistent with the current set of beliefs and γ is not already entailed.

Defaults may interact and depending on the set of default rules being applied, different "possible worlds" or hypotheses can be computed. These possible worlds are referred to as *extensions* (see below for a formal definition). Depending on the reasoning mode the *consequence problem* for terminological default theories is to decide whether a given assertional axiom is member of all extensions (skeptical mode) or of at least one extension (credulous mode) [Reiter, 1980].

Using description logic *concept terms* in default rules instead of first-order or propositional logic formulae has been extensively considered in [Baader & Hollunder, 1995a].

A *terminological default theory* is a pair (A, D) where A is an ABox, and D is a finite set of *terminological* default rules whose preconditions, justifications and consequents are concept terms. Because concept terms correspond to unary predicates ranging over a free variable, these defaults are called *open* defaults. In contrast, *closed* defaults do not contain any free variables. Unlike Reiter's original proposal, the approach of [Baader & Hollunder, 1995a] applies defaults only to those individuals that are explicitly mentioned in the world description (ABox). Default rules are never applied to implicit individuals introduced by $\exists$-restrictions. With this kind of semantics the consequence problem for (A, D) is decidable (see [Baader & Hollunder, 1995a] for details). Closed default rules can be obtained by instantiating the free variable in the concept expressions with all explicitly mentioned ABox individuals (see [Baader & Hollunder, 1995a] for a formal definition). Thus, for closed defaults, α, β_i and γ are *concept membership assertions* (ABox concept axioms).

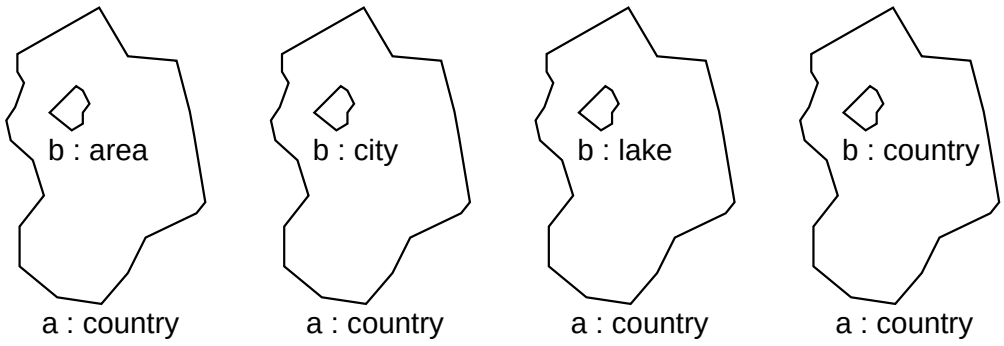
Once we have a closed default theory, a set of consequences of such a theory is referred to as an *extension* which is a set of deductively closed formulae defined by a fixed point construction. In the case of terminological default reasoning about spatial information it is also interesting to conclude spatial relations by default. Therefore, we extended the approach presented in [Baader & Hollunder, 1995a] to be able to deal with role assertions in default rules. This can be achieved by allowing $\mathcal{ALCRP}(\mathcal{RCC})$ *ABoxes* inside the default rules as α, β_i and γ. Before discussing the computation of extensions of such closed default theories, we first consider some examples of using defaults in the context of terminological reasoning about spatial information.

6.6 An Image Understanding Application

We will now illustrate the use of a DL with integrated topological reasoning for an example which could be part of an aerial image interpretation task. The idea is to use defaults for hypothesis generation regarding the classification of areas in an image. The default reasoning component of the DL will generate extensions of the ABox representing hypothesized classifications which are consistent with the rest of the knowledge base. The consistency check involves spatial reasoning. Additionally, also spatial relationships between areas could be hypothesized, for example, in case of partial object occlusions (see below).

Example 1

Suppose we have incomplete knowledge regarding the classification of the object b in Figure 6.6(a). We already know that a is a country, but area b is only known to be an area. The image interpretation system may want to generate possible hypotheses for b, which either could be a city (Figure 6.6(b)) or a lake (Figure 6.6(c)). Both



(a) (b) (c) (d)

Figure 6.6: Generation of hypotheses for object b.

concepts are plausible hypotheses w.r.t. the size of area b. Obviously, these different hypotheses are disjoint, since b cannot be both a city and a lake. Other hypotheses are not generated although these might be plausible at first sight. In particular, since we require that countries are always disjoint (relation dc) or touching (relation ec), the system deduces that the hypothesis shown in Figure 6.6(d) should not be generated.

Formalizing the Example: Using $\mathcal{ALCRP}(\mathcal{RCC})$'s role-forming predicate-based operator, we declare a set of role axioms according to the mentioned $\mathcal{RCC}$ predicates:

$$\begin{aligned} inside &\doteq \exists(has\_area)(has\_area).\mathsf{tpp\text{-}ntpp} \\ contains &\doteq \exists(has\_area)(has\_area).\mathsf{tppi\text{-}ntppi} \\ overlaps &\doteq \exists(has\_area)(has\_area).\mathsf{po} \\ touches &\doteq \exists(has\_area)(has\_area).\mathsf{ec} \\ disjoint &\doteq \exists(has\_area)(has\_area).\mathsf{dc} \end{aligned}$$

The following definitions of concepts are required to model domain objects representing different kinds of regions in a TBox which satisfies the $\mathcal{ALCRP}(\mathcal{D})$ restrictedness criteria. This conceptual background knowledge also applies to the subsequent examples.

$$
\begin{aligned}
area &\doteq \exists(has\_area).\textsf{is-region} \\
natural\_region &\doteq \neg administrative\_region \\
country\_region &\dot{\sqsubseteq} administrative\_region \sqcap \\
&\quad large\_scale \sqcap area \\
city\_region &\dot{\sqsubseteq} administrative\_region \sqcap \\
&\quad \neg large\_scale \sqcap area \\
lake\_region &\dot{\sqsubseteq} natural\_region \sqcap area \\
river\_region &\dot{\sqsubseteq} natural\_region \sqcap area
\end{aligned}
$$

An *area* is a two-dimensional region with some extent. Furthermore, we distinguish between *administrative_regions* and *natural_regions* which are disjoint concepts. The difference between a *country_region* and a *city_region* is that the former is *large_scale*, but the latter is not. Thus, these two concepts are disjoint as well. The intention behind the other concept definitions should be obvious.

$$
\begin{aligned}
country &\doteq country\_region \sqcap \\
&\quad \forall contains.\neg country\_region \sqcap \\
&\quad \forall overlaps.\neg country\_region \sqcap \\
&\quad \forall inside.\neg country\_region \\
city &\doteq city\_region \sqcap \\
&\quad \exists inside.country\_region \\
lake &\dot{\sqsubseteq} lake\_region \\
river &\doteq river\_region \sqcap \\
&\quad \forall overlaps.\neg lake\_region \sqcap \\
&\quad \forall contains.\bot \sqcap \\
&\quad \forall inside.\neg lake\_region
\end{aligned}
$$

A *country* is a *country_region* that can never contain or be contained within other *country_regions*. Also, *countries* never overlap other *country_regions*. Each *city* must belong to a specific *country*, i.e. must lie within a *country*. Unfortunately, we cannot write this directly as $\exists inside.country$ because the unfolded resulting term is no longer restricted. So, we have to use the somewhat weaker version with the base concept *country_region*. In our world model a *city* must be inside a *country*. For a *river* we require that it never *overlaps* or is *inside* with a *lake_region*.

$$
river\_flowing\_into\_a\_lake \doteq river \sqcap \exists touches.lake\_region
$$

A *river_flowing_into_a_lake* is a specific *river* that *touches* a *lake_region* (recall that the RCC-8 relations ec and po and also ec and ntpp-tpp are disjoint). It would

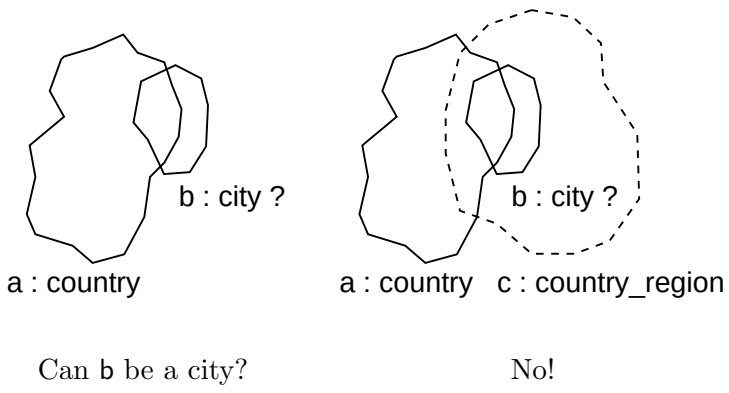


Figure 6.7: Subtle inferences due to topological constraints.

be reasonable to also state that cities do not overlap other cities etc., but this is ignored here for the sake of brevity.

Formalizing hypothesis generation in the way we already discussed informally, we now consider the following spatioterminological *default rules* d_1, d_2 and d_3:

$$d_1 = \frac{area \ : \ city}{city} \qquad d_2 = \frac{area \ : \ lake}{lake} \qquad d_3 = \frac{area \ : \ country}{country}$$

Suppose we have an ABox according to our world description as shown in Figure 6.6(a):

$$\{a : country, b : area, (a, b) : contains, (b, a) : inside\}$$

Closing defaults, i.e. instantiating the defaults d_1, d_2, d_3 over the ABox individuals a and b yields 6 different closed defaults. Now, let us assume α, β and γ have been replaced by the corresponding *assertional axioms* (e.g. instantiating the default $area \ : \ city \ / \ city$ with the individual a yields the closed default rule $\{a : area\} \ : \ \{a : city\} \ / \ \{a : city\}$ – expressions like $a : city$ are called assertional axioms or ABox axioms). We use the notation $d_i(ind)$ to refer to a default that is instantiated with the individual ind. Given our 6 closed default rules let us examine the status of each:

- Default $d_1(a)$ cannot be applied because adding $a : city$ to the ABox yields a contradiction with $a : country$. The concepts $country\_region$ and $city\_region$ are disjoint (due to $large\_scale$ and $\neg large\_scale$).
- Default $d_1(b)$ can be applied. We get an augmented ABox or *Extension 1*, see Figure 6.6(b):

$$\{a : country, b : area, b : city, (a, b) : contains, (b, a) : inside\}$$

- Default $d_2(a)$ cannot be applied because adding $a : lake$ to the ABox yields a contradiction with $a : country$. A *country* is an *administrative_region* and a *lake* is defined as a *natural_region*, and both are disjoint concepts.

- Default $d_2(b)$ can be applied. Thus, we can get an augmented ABox or *Extension 2*, see Figure 6.6(c):

  $$\{a : country, b : area, b : lake, (a, b) : contains, (b, a) : inside\}$$

  However, if we have an ABox already augmented by the conclusion of default $d_1(b)$, $b : city$, we cannot apply $d_2(b)$. So, only one of $d_1(b)$ or $d_2(b)$ can be applied, resulting in two different *extensions.*

- Default $d_3(a)$ cannot be applied, because its conclusion is already entailed by the ABox.

- Default $d_3(b)$ cannot be applied even if no other default has been applied before. Adding the default's consequent $b : country$ would yield an inconsistent ABox because a is already known to be a *country* and so, among others, $a : \forall contains.\neg country\_region$ holds. Because $(a, b) : contains$ holds and $b : country$ would imply $b : country\_region$, the default cannot be applied. Thus, we cannot get an extension corresponding to the wrong interpretation in Figure 6.6(d).

Example 2

Another subtle inference can be demonstrated by showing that the default $d_1(b)$ (as defined above) cannot be applied to conclude that object b in Figure 6.7 is a *city*. Figure 6.7 corresponds to the ABox or world description

$$\{a : country, b : area, (a, b) : overlaps, (b, a) : overlaps\}$$

Trying to assert $b : city$ would result in a constraint $b : city\_region \sqcap \exists inside.country\_region$. Therefore, polygon a cannot be the appropriate *country_region* because $(b, a) : overlaps$ holds. Due to the exists restriction there exists an implicit individual c which is a *country_ region* such that $(b, c) : inside$ holds. As can be seen in Figure 6.7, there is no way to find a spatial arrangement such that b is inside c and c does not overlap with a or does not contain a. Because a is a *country* and, therefore, may not overlap or may not be contained in another *country_region*, there is no way to conclude that b could possibly be a *city*.

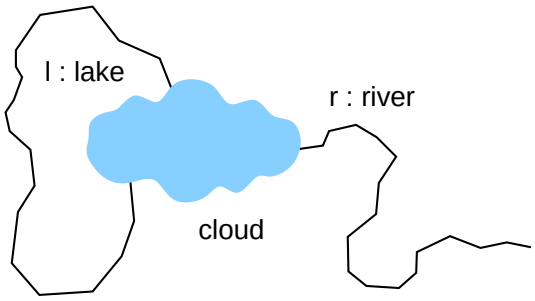


Figure 6.8: Incomplete spatial information.

Example 3

Let us consider Figure 6.8. In this case, we only have *incomplete spatial information* w.r.t. to the topological relationship between r and l, because a cloud occludes relevant parts of the two objects. The corresponding ABox is

$$\{l : lake, r : river\}$$

Since we already know that l is a lake and r is a river (perhaps this is also a hypothesis generated by previous default rule applications), we can conclude from our conceptual background knowledge that the spatial relationship between the river and the lake must be either *ec* (touches) or *dc* (disconnected or disjoint). There are no other possibilities, e.g., a river never overlaps a lake and is never contained within a lake. We can therefore hypothesize these two possible spatial relationships by default rule applications. This shows that not only concept or class memberships can be deduced by defaults. The important insight is the following duality: We can either use spatial relations between object pairs to conclude their concept memberships, or we can use already known concept memberships to conclude particular spatial relations between objects.

Unfortunately, conclusions about relations between individuals cannot be expressed with the terminological default rules introduced so far, because α, β_i and γ are limited to *concept expressions*. We now extend the terminological default rules introduced in [Baader & Hollunder, 1995a] by permitting so-called *ABox patterns* instead of concept expressions for α, β_i and γ ([Möller & Wessel, 1999]). ABox patterns are basically ABoxes with placeholders for individuals (written with capital letters). Closing the default rules instantiates the patterns with all possible combinations of individuals yielding closed defaults whose α, β_i and γ are $\mathcal{ALCRP}(\mathcal{RCC})$ ABoxes:

$$d_4 = \frac{\{X : lake, Y : river\} \ : \ \{(X, Y) : disjoint\}}{\{(X, Y) : disjoint\}}$$

$$d_5 = \frac{\{X : lake, Y : river\} \ : \ \{(X,Y) : touches\}}{\{(X,Y) : touches\}}$$

Closing the patterns, i.e. instantiating X, Y over the ABox $A = \{l : lake, r : river\}$ would yield eight different closed defaults whose α, β_i and γ are $\mathcal{ALCRP}(\mathcal{RCC})$ ABoxes, e.g. instantiating d_4 with $X \leftarrow l, Y \leftarrow r$ yields the closed default rule

$$\frac{\{l : lake, r : river\} \ : \ \{(l,r) : disjoint\}}{\{(l,r) : disjoint\}}$$

Additionally, as well as allowing variables such as X and Y, one might also be able to refer to specific ABox individuals in the ABox patterns (for instance, the individual "Bodensee").

6.7 Default Reasoning with Specificity

Let us consider the world description

$$A = \{r : river\_flowing\_into\_a\_lake, l : lake\}$$

Since it is already known that r is actually a $river\_flowing\_into\_a\_lake$ and not only a $river$, we would like to conclude that the lake l in A should be *the* lake. That is, the complex role assertion $(l,r) : touches$ should be added:

$$d_6 = \frac{\{X : lake, Y : river\_flowing\_into\_a\_lake\} \ : \ \{(X,Y) : touches\}}{\{(X,Y) : touches\}}$$

In the case of d_6, we would like to render the application of d_4 and d_5 *invalid*, because they are "less specific" than d_6 (even if d_5 yields the same conclusion, $touches$).

A default d_a is said to be more specific than d_b, $d_a \prec d_b$ iff $(\alpha(d_a) \models \alpha(d_b)) \wedge (\alpha(d_b) \not\models \alpha(d_a))$ where $\alpha(d)$ denotes the precondition of the default d. Algorithms for computing the so-called *S-extensions* (S for specificity) have already been developed by Baader and Hollunder [Baader & Hollunder, 1995b]. There is a strong conjecture that these algorithms can be applied in our $\mathcal{ALCRP}(\mathcal{RCC})$ context as well. In contrast, the ordinary extensions are called *R-extensions* (R for Reiter). In our example, we would get two different R-extensions, but only one S-extension containing the ABox axiom $(r,l) : touches$. The other *R-extension* containing $(r,l) : disjoint$ could not be derived, since only the most specific active defaults are applied when computing S-extensions. This would render the application of d_4 and d_5 impossible because d_6 is also active and more specific than both d_4 and d_5.

This concludes the illustrating examples. We have shown that standardized reasoning services of a DL can be used to generate hypotheses consistent with the available

knowledge. This is, of course, only one of several building blocks required for image interpretation. Questions regarding the ordering of extensions, the verification of possible extensions with additional evidence, the incorporation of metric information a.o. have not been treated and in many cases cannot be answered. Below the line, however, we hope that the value of inference services has been demonstrated.

In the next section we will show that the *consequence problem* is decidable for terminological default theories with default rules containing $\mathcal{ALCRP}(\mathcal{RCC})$ ABoxes. Since we can always obtain ordinary ABoxes from our ABox patterns by closing them, the consequence problem is decidable for defaults with ABox patterns as well. We believe this is an important result which extends the applicability of default reasoning to interpretation tasks as they arise in high-level computer vision.

6.8 Computing Extensions

Intuitively, given a closed terminological default theory (A, D) a deductively closed set of consequences of such a theory is referred to as an *extension*. As usual, the exact definition is given by a fixpoint construction. We cite a formal definition taken from [Baader & Hollunder, 1995a]. $Th(\Gamma)$ stands for the deductive closure of a set of formulae Γ. In a description logic context Γ is an ABox.

Definition 64 Let E be a set of closed formulae and (A, D) be a closed default theory. We define $E_0 := A$ and for all $i \geq 0$

$$E_{i+1} := E_i \cup \{\gamma \mid \alpha : \beta_1, \ldots, \beta_n/\gamma \in D, \alpha \in Th(E_i), \neg\beta_1, \ldots, \neg\beta_n \notin Th(E)\}$$

Then, $Th(E)$ is an extension of (A, D) iff

$$Th(E) = \bigcup_{i=0}^{\infty} Th(E_i)$$

Note that, in principle, this definition for an extension $Th(E)$ has a non-constructive nature because in the definition the deductive closure $Th(E)$ is already used in each iteration step. Nevertheless, as we will see below, the definition induces an algorithm for actually computing extensions if the implicit entailment subproblems in the definition are decidable (see also [Baader & Hollunder, 1995a]).

In order to be able to infer spatial relations between domain objects, the basic terminological default reasoning approach described in [Baader & Hollunder, 1995a] is adapted. The basic idea is that the precondition, the justifications and the consequent of a default can be ABoxes.

Definition 65 A *spatioterminological default rule* d (or spatioterminological default for short) has the form $d = \alpha : \beta_1 \ldots \beta_n \ / \ \gamma$ where α, β_i and γ are consistent and restricted $\mathcal{ALCRP}(\mathcal{RCC})$ ABoxes which may, among others, contain predicate-based role axioms of the form $(a, b) : \exists(\mathsf{has\_area})(\mathsf{has\_area}).\mathsf{P}$ with P being an $\mathcal{RCC}$ predicate of arity two. A *spatioterminological default theory* is a tuple (A, D) where D is a set of spatioterminological default rules and A is a consistent and restricted $\mathcal{ALCRP}(\mathcal{RCC})$ ABox.

Lemma 66 A restricted $\mathcal{ALCRP}(\mathcal{RCC})$ ABox axiom δ is logically entailed by a restricted $\mathcal{ALCRP}(\mathcal{RCC})$ ABox A,

$$A \models \delta, \qquad \text{iff} \qquad \begin{cases} \delta = a : C \longrightarrow \\ \qquad \neg SAT(A \cup \{a : \neg C\}) \\ \delta = (a, b) : R \longrightarrow \\ \qquad \neg SAT(A \cup \{a : \forall R.X_{new}, b : \neg X_{new}\}), \\ \delta = (a, b) : f \longrightarrow \\ \qquad \neg SAT(A \cup \ \{a : \forall f.X_{new} \sqcup \\ \qquad\qquad \exists(f).\mathsf{is\text{-}region}, b : \neg X_{new}\}), \\ \delta = (a, x) : f \longrightarrow \\ \qquad \neg SAT(A \cup \ \{a : \exists(f).\Psi \sqcup \exists f.\top \sqcup \forall f.\bot, x : \overline{\Psi}\}), \\ \delta = (x_1, x_2) : P \longrightarrow \\ \qquad \neg SAT(A \cup \{(x_1, x_2) : \overline{P}\}) \\ \delta = (a, b) : \exists(u)(v).P \longrightarrow \\ \qquad \neg SAT(A \cup \{(a, b) : \exists(u)(v).\overline{P}\}) \wedge \\ \qquad \neg SAT(A \cup \{a : \forall u.\top\}) \wedge \neg SAT(A \cup \{b : \forall v.\top\}), \\ \qquad \text{where } u = v = has\_area, \end{cases}$$

where X_{new} is a new atomic concept that does not appear elsewhere in the ABox A. X_{new} is used as a "marker" concept. Analogously, Ψ (resp. $\overline{\Psi}$) is a new (otherwise unused) concrete domain "marker" predicate. These two predicates have the property that they do not interact with the other concrete domain predicates P_i. Therefore, the two arbitrary conjunctions of concrete domain predicates $\bigwedge_{i=1}^{k} P_i \wedge \Psi$ and $\bigwedge_{i=1}^{k} P_i \wedge \overline{\Psi}$ are satisfiable iff $\bigwedge_{i=1}^{k} P_i$ is satisfiable. However, $\bigwedge_{i=1}^{k} P_i \wedge \Psi \wedge \overline{\Psi}$ is always unsatisfiable, regardless of the satisfiability of $\bigwedge_{i=1}^{k} P_i$. Additionally, R is a primitive role and f is a feature. $SAT(A)$ decides the ABox consistency problem for an ABox A. Please note that a, b are interpreted as abstract domain objects, unlike x, x_1, x_2 which are interpreted as concrete domain objects. The concrete domain S_2 and the abstract domain are disjoint.

Proof. (Sketch) The first case is the instance checking problem, which is decidable because C is a restricted concept term. The second case deals with primitive role assertions. In case that b is an R successor of a, the assertion $a : \forall R.X_{new}$ would entail

$b : X_{new}$, where X_{new} is a new (otherwise unused) atomic "marker" concept. This would obviously contradict the assertion $b : \neg X_{new}$. The same trick can be applied to check whether $(a, b) : f$ holds. Unlike primitive roles, the f successor of a might be a concrete domain object, which would also contradict the assertion $a : \forall R.X_{new}$. However, we can check for the presence of a concrete domain filler f of a by asserting $\exists(f).\mathsf{is\text{-}region}$. To check if $(a, x) : f$ holds, we cannot propagate a X_{new} marker, since $x : X_{new}$ yields an immediate contradiction (recall that the concrete domain and the abstract domain are disjoint). We therefore have to propagate a new, otherwise unused concrete domain "marker" predicate Ψ. As stated above, Ψ (resp. $\overline{\Psi}$) does not affect the satisfiability of the other concrete domain predicates P_i, and therefore the only possibility to get a contradiction with respect to Ψ ($\overline{\Psi}$) is to have asserted both $\Psi(x)$ and $\overline{\Psi(x)}$ for a concrete domain object x. However, we do not want to infer $(a, x) : f$ if a has an f successor in the abstract domain or can not have an f successor in the concrete or abstract domain. We therefore check for the presence of an abstract domain filler f of a by asserting $\exists f.\top$ and additionally check whether it is known that a can't have an f successor by asserting $a : \forall f.\bot$. In the fifth case we must decide whether the binary concrete domain predicate P holds for the concrete domain objects x_1, x_2. There exists a concrete domain predicate $\overline{P}$, the negation of P. The last case is more problematic, because the $\mathcal{ALCRP}(\mathcal{RCC})$ language does not provide a negation operator for predicate-based role axioms. However, we can check whether $(a, b) : \exists(has\_area)(has\_area).\overline{P} \lor a : \neg\exists(has\_area).\mathsf{is\text{-}region} \lor b : \neg\exists(has\_area).\mathsf{is\text{-}region}$ holds. The NNF of $\neg\exists(has\_area).\mathsf{is\text{-}region}$ is $\exists(has\_area).\mathsf{is\text{-}no\text{-}region} \sqcup \forall(has\_area).\top$. Since $\exists(has\_area).\mathsf{is\text{-}no\text{-}region}$ is inconsistent, the resulting term is $(a, b) : \exists(has\_area)(has\_area).\overline{P} \lor a : \forall has\_area.\top \lor b : \forall has\_area.\top$. Obviously, this is not an $\mathcal{ALCRP}(\mathcal{RCC})$ ABox. However, $A \cup \{a_1 \lor a_2 \lor \cdots \lor a_n\}$ is inconsistent iff $\forall a_i : A \cup \{a_i\}$ is inconsistent. Note that the predicate name $\overline{P}$ exists because the concrete domain is required to be admissible. □

Theorem 67 The consequence problem for a spatioterminological default theory (A, D) is decidable.

Proof. Considering the sound and complete tableaux calculus for deciding the consistency of restricted $\mathcal{ALCRP}(\mathcal{RCC})$ ABoxes, $\delta \in Th(\Gamma)$ iff $\Gamma \models \delta$. Thus, instead of taking $Th(E)$ we can view the ABox E as a representative for an extension. The fixpoint construction in Definition 64 can be used as a tester for determining whether a given ABox E really *is* an extension of a default theory (A, D). Since each extension E is an ABox having the form $A \cup \{\gamma \,|\, \alpha : \beta_1 \ldots \beta_n/\gamma \in D'\}$ for a set of so-called *generating defaults* $D' \subseteq D$, we can simply check for each element E of $\{A \cup X \mid X \in 2^{\{\gamma \,|\, \alpha:\beta_1 \ldots \beta_n/\gamma \in D\}}\}$ whether it is an extension or not.

The following inference problems need to be decided:

1. $\alpha \in Th(E_i)$: This can be easily tested by checking whether $E_i \models \alpha$ where $\alpha = \{a_1, a_2, \ldots, a_n\}$. We can decide this *ABox entailment problem* iff we can decide whether each assertional axiom a_i is logically entailed by E_i, i.e. $\forall a_i \in \alpha : E_i \models a_i$. This can be decided according to Lemma 66.

2. $\neg\beta_i \notin Th(E)$: This can be checked by testing whether $E \not\models \neg\beta_i$. However, $E \not\models \neg\beta_i$, where $\beta_i = \{b_1, b_2, \ldots, b_n\}$ iff $A \cup \beta_i$ is consistent. The ABox consistency problem for restricted $\mathcal{ALCRP}(\mathcal{RCC})$ ABoxes is decidable.

3. $Th(E) = \bigcup_{i=0}^{\infty} Th(E_i)$: The fixpoint can be constructed in a finite number of steps because we consider only a finite number of defaults. In principle, we have to decide the *ABox equivalence problem*. An ABox A_1 is equivalent to an ABox A_2, $A_1 \equiv A_2$ iff $A_1 \models A_2$ and $A_2 \models A_1$, i.e. the ABox equivalence problem can be reduced to two ABox entailment problems.

□

In [Baader & Hollunder, 1995a] another algorithm is discussed for computing extensions. Empirical tests with prototype implementations indicate that this algorithm seems to be more efficient in the average case. There is a strong conjecture that the algorithm is also applicable in the $\mathcal{ALCRP}(\mathcal{RCC})$ context. Furthermore, it can easily be seen that the results for spatioterminological default theories wrt. $\mathcal{ALCRP}(\mathcal{RCC})$ can be extended to $\mathcal{ALCRP}(\mathcal{D})$ as well.

6.9 Discussion

In the context of computer vision a first theory for applying spatioterminological default reasoning has been investigated. The work presented in this chapter extends previous work on $\mathcal{ALCRP}(\mathcal{RCC})$ [Haarslev et al., 1998; Haarslev et al., 1999b] by integrating default reasoning into spatioterminological reasoning. New contributions to previous work on terminological default reasoning [Baader & Hollunder, 1995a] are: As a base language, the expressive spatioterminological description logic $\mathcal{ALCRP}(\mathcal{RCC})$ is used. Allowing not only concept terms as formulae in default rules but also restricted $\mathcal{ALCRP}(\mathcal{RCC})$ ABoxes with complex role assertions is necessary from an application-oriented point of view but imposes a number of theoretical problems. We have shown that the possible extensions of a closed $\mathcal{ALCRP}(\mathcal{RCC})$ spatioterminological default theory can be effectively computed.

An implementation of $\mathcal{ALCRP}(\mathcal{D})$ is described in [Turhan, 1998]. With the implementation of the $\mathcal{ALCRP}(\mathcal{D})$ default reasoning substrate, an implementation of an $\mathcal{ALCRP}(\mathcal{D})$ TBox and ABox management system as well as an RCC-8 relation network consistency checker is also available for research purposes. Qualitatively

speaking, tests with the current implementation indicate that for small problems with few ABox assertions, results can be expected in a reasonable time but, currently, runtimes dramatically increase when more than only a few individuals are involved.

As pointed out before, spatioterminological default reasoning can provide an important service for constrained hypothesis generation in vision systems. The development of the underlying foundations is a necessary step towards knowledge-based vision system architectures, where powerful inference services can be employed instead of costly and error-prone application-specific programming. As pointed out before, the selection of RCC-8 predicates is motivated by its widespread use. Other calculi for qualitative and quantitative spatial representations might be considered as well. See e.g. [Eschenbach, 1999] for a careful theoretical analysis of spatial structures.

Chapter 7

Deductive Information Systems

As more and more information sources of various kinds become available for an increasing number of users, one major challenge for computer science is to provide adequate access and retrieval mechanisms. This is not only true for web-based information which by its nature tends to be highly unorganized and heterogeneous, but also for dedicated databases which are designed to provide a particular service.

7.1 Example-Based Instance Retrieval

The instance retrieval inference problem is usually defined with respect to a concept, an ABox and a TBox and possibly an RBox. The problem can be slightly extended. We can characterize the set of individuals which are to be retrieved not only by a concept term but also – or alternatively – by a set of example individuals.

An example-based instance retrieval problem is given by a filter concept F and a set of individuals $\{i_1, \ldots, i_n\}$. The filter concept F is used to preselect a set of individuals. The example individuals $\{i_1, \ldots, i_n\}$ have the purpose to select "similar" individuals from the set of preselected individuals. Note that the individuals $\{i_1, \ldots, i_n\}$ need not be instances of F. We will show that the example individuals can be used to derive a second concept term C describing individuals which are "related" to the example individuals. The returned individuals are instances of $F \sqcap C$.

In order to retrieve individuals "related" to example individuals it is necessary to compute an abstraction representing the commonalities of the example individuals. As "relatedness" of individuals is usually hard to define, we pursue a terminology-based approach. Instead of the individuals, we consider the direct types of the individuals with respect to a TBox and an RBox. More specifically, we describe each individual of the example-based instance retrieval query by the conjunction of its direct types. Informally speaking, the idea is to compute an abstraction for all

direct type conjunctions. The abstraction will represent the commonalities of the example individuals as described by the direct type conjunctions.

We now give a formalization of the commonalities of a set of concepts in terms of the Least Common Subsumer (LCS).

Definition 68 (Least Common Subsumer) A concept C is a least common subsumer of two consistent concepts $\mathsf{D1}$ and $\mathsf{D2}$ iff C subsumes both $\mathsf{D1}$ and $\mathsf{D2}$ and there is no other common subsumer of $\mathsf{D1}$ and $\mathsf{D2}$ that is subsumed by C (see [Cohen et al., 1992]).

If the representation language contains an OR operator, the LCS of two concepts is the disjunction of the input concepts. As we have seen above, some description logic languages such as CLASSIC do not offer an OR operator. Furthermore, for many applications, the expressivity of CLASSIC is appropriate. Even a more restricted language such as $\mathcal{ALN}$ can successfully be used for representing domain models in many cases. The advantage is that the consistency problem is in P in this case. Thus, there have been investigations to develop algorithms for computing the LCS in these less expressive languages.

For the language $\mathcal{ALN}$ the LCS of two consistent concepts can be computed as follows [Cohen et al., 1992]. If the LCS operation is applied w.r.t. a TBox with concept introduction axioms, the arguments must be unfolded before applying the following LCS function. Any concept term can be transformed into an unfolded form by iteratively replacing (or inserting) concept names by their defining terms.

- $\mathsf{LCS}(\mathsf{C}_{1_1} \sqcap \ldots \sqcap \mathsf{C}_{1_k}, \mathsf{C}_{2_1} \sqcap \ldots \sqcap \mathsf{C}_{2_l}) := \mathsf{LCS}(\mathsf{C}_{1_1}, \mathsf{C}_{2_1}) \sqcap \ldots \sqcap \mathsf{LCS}(\mathsf{C}_{1_k}, \mathsf{C}_{2_l})$
- $\mathsf{LCS}(\forall\, \mathsf{r1}\,.\,\mathsf{C}, \forall\, \mathsf{r2}\,.\,\mathsf{D}) := \mathsf{if}\ \mathsf{r1} = \mathsf{r2}\ \mathsf{then}\ \forall\, \mathsf{r1}\,.\,\mathsf{LCS}(\mathsf{C}, \mathsf{D})\ \mathsf{else}\ \top$
- $\mathsf{LCS}(\exists_{\geq n}\, \mathsf{r1}, \exists_{\geq m}\, \mathsf{r2}) := \mathsf{if}\ \mathsf{r1} = \mathsf{r2}\ \mathsf{then}\ (\exists_{\geq min(n,m)}\, \mathsf{r1})\ \mathsf{else}\ \top$
- $\mathsf{LCS}(\exists_{\leq n}\, \mathsf{r1}, \exists_{\leq m}\, \mathsf{r2}) := \mathsf{if}\ \mathsf{r1} = \mathsf{r2}\ \mathsf{then}\ (\exists_{\leq max(n,m)}\, \mathsf{r1})\ \mathsf{else}\ \top$

It can be easily verified that the LCS is an associative operation. Using the LCS operation we define a sequence Q_i of query concepts. Let T_i be the conjunction of the direct types of the individual i_k. Further, let $Q_0 := \{T_1, \ldots, T_n\}$ be the set of direct type conjunctions for each example individual i_k given as parameter to the example-based instance retrieval operation ($k \in 1..n$). Furthermore, $LCS_i := lfold(\text{LCS}, \top, (Q_{i_1}, \ldots, Q_{i_{n_i}}))$. The function $lfold$ successively applies the first argument, a left-associative function f, to the result of the previous application of f (initially $\top$ is used) and the components of the third argument (a sequence). The index n_i is the cardinality of the set Q_i. Q_{i+1} is defined as the set of parents of LCS_i. Note that there always exists an LCS_l which is equal to $\top$.

The *example-based instance retrieval problem* is to find a minimal index i for the sequence of concepts Q_i such that the set of example individuals $\{i_1, \ldots, i_n\}$ is a proper subset of the set of individuals S_Q_i retrieved by $instance\_retrieval(F \sqcap Q_{i_1} \sqcap \ldots \sqcap Q_{i_{n_i}})$ or S_LCS_i retrieved by $instance\_retrieval(F \sqcap LCS_i)$. Once the minimal index i is obtained, the result of the example-based instance retrieval operation is S_Q_i if $\{i_1, \ldots, i_n\}$ is a proper subset of S_Q_i, or S_LCS_i otherwise. The concept F is the filter concept introduced above.

In addition to example-based instance retrieval, the LCS operator is used as a subtask for the "bottom-up" construction of knowledge bases based on the DLs $\mathcal{ALN}$ with cyclic concept definitions [Baader & Küsters, 1998]. See also [Kietz & Morik, 1994] for a similar application concerning the constructive induction of a CLASSIC TBox from data.

7.2 Cooperative Information Systems: Agent-Based Computing

In this section a cooperative information system scenario with different agent, each with specific reasoning capabilities, is discussed. The guiding example is a "TV-Assistant" with a database containing TV-program information. The task of the TV-Assistant is to guide TV watchers in selecting "their" favorite program item from a potentially large set of candidates. For example, the TV-Assistant should be able to identify "a pirate movie with sailing ships" among the 300 movies which a new German digital TV channel broadcasts every 30 minutes. Furthermore, based on the preferences of a user w.r.t. the TV program, the TV-assistant also has to determine suitable commercials. Thus, it is demonstrated that description logics can play an important role in the important area of e-business applications.

There is obviously a large variety of criteria by which TV watchers would like to express their preferences. They may want to refer to the contents of the program item in terms of its genre type, main characters, location, historical events, plot etc. They may also want to refer to production information, e.g. producer, cast, recording technique, date of origin etc., maybe also to their particular viewer situation, e.g. language and age requirements. While some of these criteria can already be used in existing TV-program services (e.g. genre, cast, age recommendations), inference-based or deductive information retrieval is in its infancy.

The prevailing approaches for inference-based access and retrieval utilize textual information in terms of keywords and word statistics. Surface-based textual information retrieval, typically based on string-indexing, offers several advantages, in particular the use of queries involving natural language terms, and the availability of text documents. On the other hand, string-indexing is unreliable in several

respects. First, documents may not be produced with the aim to support textual retrieval. Hence it is a matter of chance whether or not a desirable keyword really appears in the document. Second, as examples of TV-program selections show, naturally expressed queries may involve terms which are less specific than the textual descriptor of the data (or only conceptually close to it), e.g. "sailing ship" instead of "frigate". Similarly, one may be interested in a movie about one's home town, say Hamburg. Inference-based retrieval should not only index into descriptors involving the string "Hamburg" but possibly also into locations spatially related to Hamburg, e.g. "Northern Germany" or "Reeperbahn" (Hamburg's famous red-light district). It is also apparent that additional conceptual information must be exploited to avoid unwanted retrieval hits involving certain popular food items.

In this chapter we investigate the use of conceptual descriptions based on description logics for deductive information retrieval with an agent-based scenario. The chapter deals with two main aspects. First, strengths and limitations of DLs for information retrieval are investigated. Second, cooperation strategies for agents whose reasoning is based on different DLs are analyzed.

When investigating DLs for information retrieval, one can consider a wide variety of languages which have been analyzed in research and partly implemented so far. The most important differentiating aspect of DLs is expressiveness, i.e. what concept expressions may be formulated within a particular DL, and the resulting complexity of inference procedures such as consistency checking or subsumption computations. Hence there are DLs of fairly limited expressiveness but attractive runtime properties, and there are DLs with enriched expressiveness for which one has to pay with doubly exponential inference complexity. Furthermore, if certain restrictions on expressiveness are not observed, a DL may become undecidable in the sense that terminating inference procedures which are sound and complete do not exist any longer. This may be acceptable in some applications, but must be considered a severe disadvantage where reliability is at stake.

From what we know about DLs today, it seems reasonable to expect that there will not be a single DL optimally suited for all knowledge-based applications. Rather we have to consider heterogeneous special-purpose knowledge bases using DLs of different expressiveness. Each of the knowledge bases may be designed to meet different design goals. One knowledge base may provide quick but crude inferences, another may allow more sophisticated inferences at the cost of longer response times, a third one may be a specialist for temporal reasoning, a forth one for spatial reasoning, and so on. This has led us to investigate DL systems in an agent-based scenario, where different DLs are organized to cooperate in an information retrieval task.

The general structure of an agent-based scenario is shown in Figure 7.1. The user (which may be a software system) submits information retrieval tasks to a broker which is the central node in the agent network. The broker may invoke services of

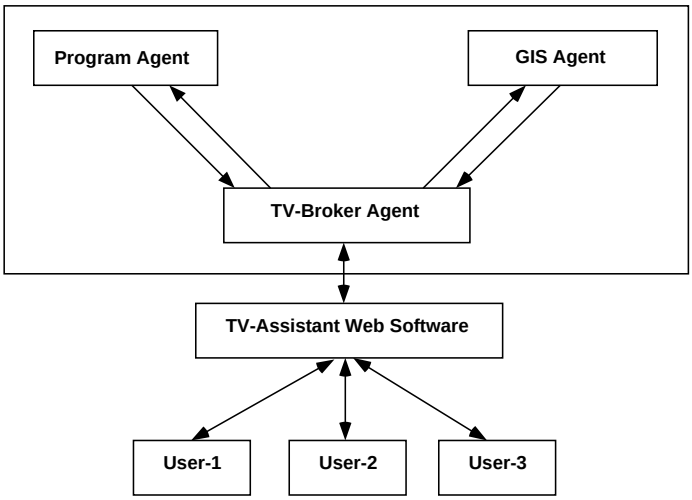


Figure 7.1: Overview of the System Architecture. A broker agent communicates with an agent which responsible for the tv program and an agent which is a specialist for geographical information system (GIS) requests. The broker agent provides the interface for the software of the TV-Assistant web server.

other agent nodes which are his subagents. The knowledge bases of the agents may be based on different DLs and may have been developed independently from each other. In order to communicate with his subagents, the broker must have certain interschema knowledge to formulate an information retrieval task for a subagent and make use of his result.

We will consider instance retrieval and instance checking as basic information retrieval tasks. As defined above, instance retrieval is an inference service of a DL system where a concept term is submitted as a query, and the task is to retrieve all individuals which are instances of the concept term. For instance checking, a query comprises a concept term together with a set of individuals which are to be checked against the concept term. Both tasks are identical from a logical perspective and amount to consistency checking of ABoxes (if full negation is supported in the logic). We will also consider the application an example-based extension of instance retrieval, where examples are used to compute a second concept term of which the retrieved individuals have to be instances (see above).

Since all reasoning processes of an agent are based on the agent's terminology, it is obvious that instance checking can only be performed by an agent if the task is formulated in terms of the agent's own terminology. In our scenario, it is the task of the broker to transform queries or subtasks thereof so that subagents can be employed. For this purpose, the broker is equipped with interschema knowledge which relates different terminologies to each other. Thus, the broker can approximate

assertional descriptions and concept terms by expressions which only use the concepts and roles of a particular subagent (for details see below). In exceptional cases, these expressions may be equivalent, but in general they will be approximations. Hence it is an interesting question whether subagent reasoning can be employed at all for instance checking without jeopardizing the correctness of the overall result.

In this section it is shown that this is indeed possible - albeit only to some extent. The basic idea is to approximate queries in such a way that the concept term of a query (which constrains individuals) is specialized, and individual descriptions - if there are any - are generalized. Thus, if approximated individuals are found to be instances of the approximated concept term, the original individuals will also be instances of the original concept term. On the other hand, we have to be aware of the fact that the combined logics of two agents may be undecidable. If this is the case, inconsistency may not be detected reliably by any procedure which combines the knowledge of these agents. This is an inherent limitation with interesting consequences regarding the conclusions which one can draw in a multi-agent information retrieval scenario.

7.2.1 Multi-Agent Inference Problems

We now turn to the multi-agent information retrieval problem which has been sketched in the introduction of this section. Agents comprising different knowledge bases expressed in different description logics cooperate for information retrieval. The approach presented in this section presumes that one of the agents plays the part of a broker, the others, called specialists, supply information to the broker. The broker receives queries from a user, communicates with specialists, and delivers answers to the user. Communication with the user is performed using the broker's terminology. To be able to communicate with the other agents, the broker has knowledge about the concept and role names of each agent, and how they relate to the broker's terminology (interschema axioms [Catarci & Lenzerini, 1993]). The main task of the broker is to transform queries into the terminology of another agent and transform answers back into the broker's terminology. In the following, this is described more precisely.

Inference Problems w.r.t. Namespaces

In order to use a description logic in an application, a set of atomic concepts and roles tailored to the application must be specified. Relationships between the atomic concepts or roles can be defined with axioms. To distinguish between different knowledge bases, we define the notion of a namespace.

Definition 69 (Namespace) A namespace n is a set of atomic concepts or atomic roles with a common prefix n: where n is the name of the namespace.

Definition 70 (Concept and Role from a Namespace) A concept is called *a concept from a namespace* n iff all atomic component concepts have the prefix n:. A role is called *a role from a namespace* n iff it has a prefix n:.

Using the notion of a namespace, the special role of the broker agent can be specified. Its knowledge base contains atomic concepts and roles from multiple namespaces (with different prefixes). Thus the broker can perform inference services w.r.t. to different namespaces. For instance, the broker can compute the *parents* of a (not necessarily atomic) concept C w.r.t. a namespace n and a TBox $\mathcal{T}$. The result is the set of most-specific atomic concepts in the namespace n that subsume C. The set of *children* w.r.t. to a namespace and the *direct types* of an individual w.r.t. a certain namespace are defined analogously.

Namespace Transformations for Concepts, Roles and ABoxes

Let us consider an instance checking task now, and how the broker may invoke a specialist. The query consists of a concept term, an individual name and an ABox in the broker's terminology. The task is to check whether the individual is an instance of the concept term. If the broker can prove w.r.t. its own knowledge base either that the individual is an instance of the concept term or that it is not, the task is solved (but see the section on inconsistent queries below). If the broker's knowledge is inconclusive, a specialist may help. Hence the query must be transformed into an approximate query with concepts and roles only in the namespace of the specialist. The key idea is to transform the query such that the ABox is abstracted and the concept term is refined (or specialized). Furthermore, it might be necessary to ensure that abstractions and refinements fulfill certain restrictedness criteria (see Section 6.2).

Definition 71 (ABox Abstraction) An ABox $\mathcal{A}'$ is an *abstraction* of an ABox $\mathcal{A}$ iff $\mathcal{A} \models \mathcal{A}'$ holds.

We use the following heuristics to compute an ABox abstraction. For each role found in a role assertion of an ABox the most-specific superrole of the namespace of the destination agent is inserted. Note that the most-specific superrole in the namespace of the destination agent may also be a synonym of the role being transformed. If the most-specific superrole is not unique, in the namespace of the consulted agent a new role is dynamically added with appropriate inclusion axioms such that the new role is a subrole of the set of most-specific superroles. If no superrole exists in the TBox of a certain destination agent, then *agent_name*:$\mathsf{related}$ is used. The role $\mathsf{related}$ is assumed to be a superrole of all atomic roles of an agent. Similar transformations are employed for the atomic concepts used in concept assertions. Either synonyms or the conjunction of the parents w.r.t. the namespace of the destination agent are

inserted. Note that in the worst case $\top$ will be returned as an abstraction of a certain concept.

Definition 72 (Concept Refinement) A concept C' is a *refinement* of a concept C iff C subsumes C'.

There are different strategies for computing concept refinements. First, in order to refine a query concept C_0, the children w.r.t. the namespace of the destination agent are computed. Given the children $C_1, \ldots, C_n$ of a concept C_0, a refinement is the disjunction $C_1 \sqcup \ldots \sqcup C_n$. If only the bottom concept $\bot$ is returned as a child of C_0, a second strategy is employed. A refinement can be computed w.r.t. the form of the concept. In a similar way as in the abstraction process of the ABox, the roles and the concept terms in the query concepts are transformed. Let us consider an existential restriction $\exists\,\mathsf{R}\,.\,\mathsf{C}$ as an example. The transformation is a concept $\exists\,\mathsf{R}'\,.\,\mathsf{C}'$ where R' is the most-general subrole of R in the namespace of the destination agent and C' is the result of the (recursive) transformation of C. For atomic concepts, the transformation is the disjunction of the children w.r.t. the namespace of the destination agent (see the first strategy). Another strategy could be to use defaults in order to specialize queries (see e.g. [Wahlöf, 1996; Lambrix et al., 1998] for applications in the context of information retrieval). The refinement or rewriting of concepts using terminologies has also been investigated, for instance, in [Baader et al., 2000] and [Badea & Niehuys-Cheng, 2000].

Broker-Based Query Answering

Let us consider now how the answers of different agents can be combined from a logical perspective.

We will first take a principled view of this problem. Let us assume that we have two TBoxes, $\mathcal{T}_1$ and $\mathcal{T}_2$, representing the knowledge of the agents A and B, respectively. For instance, let $\mathcal{T}_1$ be an $\mathcal{ALCNH}_{R^+}$ TBox and $\mathcal{T}_2$ be an $\mathcal{ALCRP}(\mathcal{D})$ TBox. Considering the results of Section 5.4.3 we know that the combination of $\mathcal{ALCNH}_{R^+}$ and $\mathcal{ALCRP}(\mathcal{D})$ is undecidable. Hence, in general, we know that there is no sound and complete (and terminating) calculus for answering a query posed to two knowledge bases of these two DLs. Nevertheless, let us consider an instance checking problem $\mathit{instance}?(i, C)$ w.r.t. the knowledge base $(\mathcal{T}_1 \cup \mathcal{T}_2, \mathcal{A})$. Furthermore, let us assume that $\mathcal{A}$ is an ABox of $\mathcal{T}_1$ and that $\mathcal{A}$ is not inconsistent. The idea is to address the query – or a suitable transformation thereof – to each knowledge base separately. If the answer to the instance checking problem $\mathit{instance}?(i, C)$ w.r.t. $(\mathcal{T}_1, \mathcal{A})$ is 'yes', then it is obvious that the answer w.r.t. $(\mathcal{T}_1 \cup \mathcal{T}_2, \mathcal{A})$ is 'yes' as well. In order to address the query to $\mathcal{T}_2$ we may have to transform it since we assume that $\mathcal{A}$ is a $\mathcal{T}_1$ ABox. For instance, $\mathcal{A}$ may contain number restrictions which are no $\mathcal{T}_2$ concept

terms. If we transform the instance checking problem such that $instance?(i, C')$ w.r.t. $(\mathcal{T}_2, \mathcal{A}')$ is solved, with $\mathcal{A}'$ an abstraction and C' a refinement, it can be easily seen that we can exploit the answer for the original query: If the answer to the sub-problem $instance?(i, C')$ w.r.t. $(\mathcal{T}_2, \mathcal{A}')$ is 'yes' then the answer to $instance?(i, C)$ w.r.t. $(\mathcal{T}_1 \cup \mathcal{T}_2, \mathcal{A})$ is 'yes' as well.

If both terminologies return 'no', we cannot be sure of the overall result because of undecidability. But we can attempt to solve the inference problem $instance?(i, \neg C)$, again by addressing each knowledge base separately. If the answer to the complementary query is 'yes' then the answer to the original query $instance?(i, C)$ is 'no' for sure.

Although some limitations are inevitable, we now have the basis for a sound query answering schema in an arbitrary agent scenario. Each agent has a sound and complete (and terminating) sub-algorithm for the inference tasks in question. There is one specific agent, the broker, which poses inference problems to other agents. For this task the broker needs a TBox that is a close approximation of the TBoxes of the other agents.

We now put things together and describe how an instance checking query can be dealt with in a multiagent scenario. The broker first tries to answer a query himself using his own ABox. If the answer is negative, the broker consults an agent whose namespace contains the individual of the query. The concept of the query must be specialized and the broker's ABox fragment of the individual (if any) must be abstracted in order to preserve correctness. If the answer of the specialist is positive, this answer is returned by the broker. In the case of a negative answer, we assume that the specialist returns his ABox fragment in which the individual in question is mentioned. The broker transforms this information into his terminology and can now consult other specialist agents using the enriched ABox information about the individual. In each case, the atomic concepts and roles are translated into the specialist's namespace while preserving correctness by ABox abstraction and concept refinement. If one of the specialists returns 'yes' then the broker returns 'yes', otherwise the answer is undecided. In this case, the broker may try to answer the query with the negated concept. If he (or one of the specialists) succeeds, the answer to the original query is 'no', otherwise it remains undecided.

A broker delegates an example-based instance retrieval task to a specialist agent whose name is determined by the namespace of the individuals mentioned in the example-based instance retrieval query. As a restriction, all individuals must be from the same namespace. The filter concept is a concept from the namespace of the broker. Therefore, it has to be transformed to be "understandable" by the consulted specialist agent. The transformed filter concept is defined to be a concept refinement w.r.t. the namespace of the consulted agent.

7.2.2 Supporting E-Business: Inferences for Internet Technology

As we have discussed in the introduction, the motivating scenario for the agent-based architecture which we investigate in this section is an information retrieval application. In particular, we focus on a TV-Assistant whose main task is to provide personalized TV-programs in a web-based environment. The user can ask for a personalized TV-program by (i) providing a time window and (ii) examples of broadcasts which characterize his interests. The TV-assistant will then retrieve broadcasts of that time window (the so-called basic selection) and mark broadcasts which are of special interest to the user according to his examples (his personal selection). This can be done, for example, by highlighting broadcast in a program table.

As a second task, the TV-Assistant has to select advertisements to be inserted into the personalized program display. To accomplish this, advertisements are associated with conceptual descriptions, so-called trigger concepts, of the types of broadcasts for which the advertisements should be displayed. So if a broadcast contained in the personal selection of a user turns out to be an instance of a trigger concept, the associated advertisement will be shown.

In our application scenario we assume that the TV-Assistant makes use of the services of a multiagent information system as shown in Figure 7.1).

At first, the basic program selection is retrieved by posing an instance retrieval query to the TV-Broker Agent with the time window expressed as the conceptual constraint. The TV-Broker Agent forwards this query to the Program Agent and receives the basic program selection.

As a second step, the personal program selection is obtained by posing an example-based instance retrieval query to the TV-Broker Agent with the examples provided by the user and the time window as a filter concept. The answer is a set of broadcasts (the personal selection) associated with additional information, for example the actors or the main location in the case of a movie.

Now the advertisements have to be determined based upon the trigger concepts which are maintained by the TV-Assistant. For each broadcast of the personal selection the TV-Broker Agent is asked whether the broadcast is an instance of a trigger concept. In our scenario, there are two agents besides the TV-Broker Agent which can possibly solve the instance checking problem. We assume that, at first, the Program Agent is asked. When his answer is 'no', the broker turns to a specialist which in our case is a GIS-agent with spatial-reasoning power.

We will now present the agents in detail and then discuss several examples.

The Agents of the Application Example

The Program Agent uses the language $\mathcal{ALN}$ with concept introduction axioms only (i.e. non-cyclic terminological axioms with the additional condition that each terminological axiom is used only once on the left-hand side of the axioms). A prototype system for the Program Agent [Möller et al., 1998] has been implemented with the knowledge representation system CLASSIC [Brachman et al., 1991] which provides an optimized ABox implementation for the language $\mathcal{ALN}$ (actually, CLASSIC supports a slightly more expressive DL, see Chapter 3). In this chapter we present only a subset of the implemented knowledge base. We assume that the TBox of the Program Agent contains the terminological axioms explained in Section 2.4.

The Program Agent will perform the example-based instance retrieval procedure explained in Section 7.1. To this end, we will extend the domain model with definitions for movie and define new concepts sailing_ship and titanic as subconcepts of ship.[1] In addition, we assume that soldier and pirate are declared as subconcepts of person. Furthermore, the domain model is extended with concepts for specific movies. Important roles for movies are has_main_character and has_main_location. For the examples we use the following terminological axioms.

$$
\begin{aligned}
\textbf{soldier} &\sqsubseteq \mathsf{person}\\
\textbf{pirate} &\sqsubseteq \mathsf{person}\\
\textbf{sailing\_ship} &\sqsubseteq \mathsf{ship}\\
\textbf{titanic} &\sqsubseteq \mathsf{ship}\\
\textbf{pirate\_movie} &\sqsubseteq \mathsf{movie} \sqcap\\
&\quad \forall\, \mathsf{has\_main\_character} \,.\, (\mathsf{pirate} \sqcap \mathsf{captain}) \sqcap\\
&\quad \forall\, \mathsf{has\_main\_location} \,.\, \mathsf{sailing\_ship}\\
\textbf{titanic\_movie} &\sqsubseteq \mathsf{movie} \sqcap\\
&\quad \forall\, \mathsf{has\_main\_character} \,.\, \mathsf{captain} \sqcap\\
&\quad \forall\, \mathsf{has\_main\_location} \,.\, \mathsf{titanic}\\
\textbf{action\_movie} &\sqsubseteq \mathsf{movie} \sqcap\\
&\quad \forall\, \mathsf{has\_main\_character} \,.\, \mathsf{action\_hero}
\end{aligned}
$$

The structure shown in the ABox below represents a small excerpt of the domain model for objects (ABox) in our TV-Assistant application.

[1] The notion titanic is modeled as a concept rather than as an instance because different individual ships might carry this name.

$$
\begin{aligned}
\mathsf{movie\_1} &: \mathsf{movie} \sqcap \\
&\quad \exists_{\leq 1}\, \mathsf{has\_main\_character} \sqcap \\
&\quad \exists_{\leq 1}\, \mathsf{has\_main\_location} \\
\mathsf{hornblower} &: \mathsf{captain} \sqcap \mathsf{soldier} \\
\mathsf{lydia} &: \mathsf{sailing\_ship} \\
(\mathsf{movie\_1}, \mathsf{hornblower}) &: \mathsf{has\_main\_character} \\
(\mathsf{movie\_1}, \mathsf{lydia}) &: \mathsf{has\_main\_location} \\
\mathsf{movie\_2} &: \mathsf{pirate\_movie} \\
\mathsf{movie\_3} &: \mathsf{titanic\_movie}
\end{aligned}
$$

For our second example we assume that the Program Agent has detailed information about the James Bond movie $\mathsf{the\_world\_is\_not\_enough\_1}$:

$$
\begin{aligned}
\mathsf{the\_world\_is\_not\_enough\_1} &: \mathsf{action\_movie} \\
\mathsf{james\_bond\_1} &: \mathsf{action\_hero} \\
\mathsf{loc\_1} &: \mathsf{baku} \\
\mathsf{loc\_2} &: \mathsf{london} \\
\mathsf{country\_1} &: \mathsf{azerbaijan} \\
\mathsf{continent\_1} &: \mathsf{asia} \\
\mathsf{port\_1} &: \mathsf{port} \\
\mathsf{sea\_1} &: \mathsf{caspian\_sea} \\
(\mathsf{the\_world\_is\_not\_enough\_1}, \mathsf{james\_bond\_1}) &: \mathsf{has\_main\_character} \\
(\mathsf{the\_world\_is\_not\_enough\_1}, \mathsf{loc\_1}) &: \mathsf{has\_main\_location} \\
(\mathsf{the\_world\_is\_not\_enough\_1}, \mathsf{loc\_2}) &: \mathsf{has\_main\_location} \\
(\mathsf{loc\_1}, \mathsf{country\_1}) &: \mathsf{capital\_of} \\
(\mathsf{loc\_1}, \mathsf{port\_1}) &: \mathsf{has\_port} \\
(\mathsf{port\_1}, \mathsf{sea\_1}) &: \mathsf{located\_at} \\
(\mathsf{country\_1}, \mathsf{continent\_1}) &: \mathsf{located\_on\_continent}
\end{aligned}
$$

The GIS Agent uses the description logic $\mathcal{ALCRP}(\mathcal{RCC})$. We assume that the TBox of the GIS Agent contains the role and concept introduction axioms introduced in Section 6.3. Before the role of the GIS Agent in our scenario can be understood we first have to discuss the inferences of the central agent, the TV-Broker Agent.

The TV-Broker Agent uses the description logic $\mathcal{ALCNH}_{R^+}$. We assume that all atomic concepts and roles of the Program Agent and the GIS-Agent as well as the corresponding terminological axioms are also available in the TBox of the TV-Broker Agent. Atomic concepts and roles "imported" from other agents are indicated by a prefix. We use prefixes pa, ga and ba for the Program Agent, the GIS Agent and the TV-Broker Agent, respectively. Queries to the TV-Broker Agent and trigger concepts use atomic concepts and roles with prefix ba only.

The first part of the TBox of the TV-Broker Agent specifies the relationships between the roles used by the Program Agent and the roles used by the GIS Agent in terms of terminological axioms:

$$\begin{aligned}
&\textbf{pa:capital_of} \sqsubseteq \text{ga:inside}\\
&\textbf{pa:has_port} \sqsubseteq \text{ga:inside_i}\\
&\textbf{pa:located_at} \sqsubseteq \text{ga:touching}\\
&\textbf{pa:located_on_continent} \sqsubseteq \text{ga:inside}\\
&\textbf{pa:has_main_location} \sqsubseteq \text{ga:inside}\\
&\textbf{ga:spatially_connected} \sqsubseteq \text{ba:spatially_connected}
\end{aligned}$$

Within the description logic $\mathcal{ALCNH}_{R^+}$, the relationships are manifested using role inclusion axioms. Furthermore, in order to approximate the semantics of topological RCC-8 relations, ga:inside and ga:inside_i are declared to be transitive roles. Obviously ga:touching is not a transitive role.

For each agent acquaintance, axioms indicating the direct subsumption relationships (parents and children) of the atomic concepts which are imported from the agent are added automatically to the TBox of the TV-Broker Agent. For instance, the role axiom

$$\textbf{ga:inside} \sqsubseteq \text{ga:spatially\_connected}$$

is added to the TBox of the TV-Broker Agent. Additional concept inclusion axioms are employed to relate the concept terms of different TBoxes. These axioms cannot be set up automatically but have to be modeled by a system engineer.

$\mathbf{pa{:}baku} \sqsubseteq \mathsf{ga{:}city}$

$\mathbf{pa{:}london} \sqsubseteq \mathsf{ga{:}city}$

$\mathbf{pa{:}azerbaijan} \sqsubseteq \mathsf{ga{:}country}$

$\mathbf{pa{:}port} \doteq \mathsf{ga{:}port}$

$\mathbf{pa{:}caspian\_sea} \sqsubseteq \mathsf{ga{:}sea}$

$\mathbf{ga{:}coastal\_city} \sqsubseteq \mathsf{ba{:}coastal\_city}$

$\mathbf{ga{:}coastal\_city} \doteq \mathsf{ga{:}city} \sqcap \exists\, \mathsf{ga{:}touching} \,.\, \mathsf{ga{:}sea}$

$\mathbf{ga{:}country} \sqsubseteq \forall\, \mathsf{ga{:}overlapping} \,.\, \neg \mathsf{ga{:}sea}$

$\mathbf{ba{:}asian\_city} \doteq \mathsf{pa{:}city} \sqcap \exists\, \mathsf{ga{:}inside} \,.\, \mathsf{pa{:}asia}$

Note that generalized concept inclusion axioms are used as well. In addition, all axioms from Section 6.3 for representing the knowledge of the GIS Agent are included into the TBox of the TV-Broker Agent (with appropriate prefixes).

Reasoning Examples

As a first example, let us assume that movie_2 and movie_3 and the filter concept C are used in an example-based query posed to the Program Agent.

$\mathsf{example\_base\_instance\_retrieval}(\{\mathsf{movie\_2}, \mathsf{movie\_3}\}, \mathsf{C})$

For answering this query, the LCS operation is applied to the (unfolded) direct types of both movies (see Section 7.1) and returns the following concept:

$\mathsf{movie} \sqcap \forall\, \mathsf{has\_main\_character} \,.\, \mathsf{captain} \sqcap \forall\, \mathsf{has\_main\_location} \,.\, \mathsf{ship}$

We can see that an abstraction of the original movies has been computed. From pirate_movie the concept sailing_ship has been abstracted to ship and from titanic_movie the concept titanic has been abstracted to ship as well. The resulting LCS concept presented above is used as a query for retrieving instances from the ABox of the Program Agent. The results to this query are further restricted by the time window filter concept.

In our example the movie movie_1 is an instance of the LCS concept and, therefore, it is returned as an answer (possibly among others).

As a second example, we assume that advertisements for *cruises* and *trips to asian cities* are to be associated with appropriate broadcasts. To trigger the advertisements we consider the concept $\exists\, \mathsf{ba{:}spatially\_connected} \,.\, \mathsf{ba{:}coastal\_city}$ and the concept $\exists\, \mathsf{ba{:}spatially\_connected} \,.\, \mathsf{ba{:}asian\_city}$, which might be set up in the interest of a travel agency. If a retrieval result (e.g. the movie the_world_is_not_enough_1) is an instance of a trigger concept, the associated role fillers for ba:spatially_connected that

are instances of ba:coastal_city or ba:asian_city are examined to find cruises or trips offered by the travel agency. The idea is, of course, that after viewing the movie, people might be inclined to book a cruise to the main location of the movie.

The TV-Broker can prove that the_world_is_not_enough_1 is an instance of $\exists$ ba:spatially_connected . ba:asian_city because ga:inside is declared as a transitive role in the knowledge base of the TV-Broker Agent. Thus, there is no need to consult another agent.

Unfortunately, given the ABox associated with the_world_is_not_enough_1 (see above) the TV-Broker Agent cannot prove that the movie is an instance of $\exists$ ba:spatially_connected . ba:coastal_city. It cannot even prove that it is an instance of $\neg\exists$ ba:spatially_connected . ba:coastal_city.

In our scenario, the TV-Broker Agent therefore asks the GIS Agent to check whether the individual the_world_is_not_enough_1 is an instance of the concept $\exists$ ga:spatially_connected . ga:coastal_city. After transforming the ABox which the TV-Broker Agent has received from the Program Agent, the following ABox is delegated to the GIS Agent.

$$
\begin{aligned}
\mathsf{the\_world\_is\_not\_enough\_1} &: \top \\
\mathsf{james\_bond\_1} &: \top \\
\mathsf{loc\_1} &: \mathsf{ga{:}city} \\
\mathsf{loc\_2} &: \mathsf{ga{:}city} \\
\mathsf{country\_1} &: \mathsf{ga{:}country} \\
\mathsf{port\_1} &: \mathsf{ga{:}port} \\
\mathsf{sea\_1} &: \mathsf{ga{:}sea} \\
(\mathsf{the\_world\_is\_not\_enough\_1}, \mathsf{james\_bond\_1}) &: \mathsf{ga{:}related} \\
(\mathsf{the\_world\_is\_not\_enough\_1}, \mathsf{loc\_1}) &: \mathsf{ga{:}inside} \\
(\mathsf{the\_world\_is\_not\_enough\_1}, \mathsf{loc\_2}) &: \mathsf{ga{:}inside} \\
(\mathsf{loc\_1}, \mathsf{country\_1}) &: \mathsf{ga{:}inside} \\
(\mathsf{loc\_1}, \mathsf{port\_1}) &: \mathsf{ga{:}inside\_i} \\
(\mathsf{port\_1}, \mathsf{sea\_1}) &: \mathsf{ga{:}touching} \\
(\mathsf{country\_1}, \mathsf{continent\_1}) &: \mathsf{ga{:}inside}
\end{aligned}
$$

Considering the example presented in Section 6.3 it can be easily verified, that the answer of the GIS Agent to the query is 'yes' because ga:spatially_connected is a superrole of ga:inside. In fact, the third configuration shown in Figure 6.3 qualitatively describes exactly the relation of Baku and Azerbaijan (see Figure 7.2).

Figure 7.2: Map indicating the position of Baku, the capital of Azerbaijan.

Once the TV-Broker Agent knows the result of the query, it adds the result to the ABox originally sent to the GIS Agent, in our example the assertion $\mathsf{the\_world\_is\_not\_enough\_1} : \exists\, \mathsf{ga{:}spatially\_connected}\,.\,\mathsf{ga{:}coastal\_city}$. Now the software module of the TV-Broker can be instructed to insert specific travel agency commercials associated with the trigger concepts (in this case e.g. cruises).

Note that, in general, combining knowledge of independent knowledge bases may uncover inconsistencies. For example, the ABox returned by the GIS Agent could prove inconsistent with the ABox of the TV-Broker Agent. In this case no meaningful answer can be supplied.

The example in this section demonstrates the combined expressive power of different DLs. $\mathcal{ALCNH}_{R^+}$ provides generalized concept inclusions, role hierarchies and transitive roles which are needed to represent much of the knowledge required in the application domain. However, some of the ontological interdependencies cannot be captured due to undecidability results. Using the formalism $\mathcal{ALCRP}(\mathcal{RCC})$ it is possible to include ontological interdependencies concerning conceptual and spatial knowledge.

It would be possible to extend the language $\mathcal{ALCNH}_{R^+}$ with inverse roles and so-called qualified number restrictions (e.g. [Horrocks et al., 1999b]) in order to better approximate the knowledge of the specialists (in our case the GIS Agent) in the TV-Broker Agent. However, developing an optimized ABox reasoner implementation for the extended logic is a difficult task and subject to further research. In any case, due to the undecidability result, it would be only an approximation. The

agent architecture proposed in this chapter provides an organized way to cope with the undecidability and incompleteness problems resulting from the combination of different expressive representation languages.

7.2.3 Related Work

Information retrieval in a distributed context is a commercially very interesting research topic. There exists a vast amount of scientific contributions and it is hardly possible to cover at least a small subset. Each of the different approaches has its own pros and cons. Here, we focus on related work concerning information retrieval in the context of description logics rather than on work based on database theory and data structure conversion (e.g. the TSIMMIS system [Garcia-Molina et al., 1997]) or other knowledge representation approaches (e.g. [Fensel et al., 1998]).

An early work about the application of description logics for information retrieval purposes is [Meghini et al., 1993]. While some authors focus on multi-valued logic in order to capture the notion of "relevance" (e.g. [Meghini & Straccia, 1996]) most contributions rely on a standard semantics. For instance, the Information Manifold project [Levy et al., 1996] has extended the CLASSIC description logic [Brachman et al., 1991] with so-called conjunctive queries in order to provide a more expressive query language. Solutions for query refinement based on defaults have been developed in [Lambrix et al., 1998].

Subsumption of conjunctive queries for expressive description logics have also been investigated in [Horrocks et al., 1999a]. It would be interesting to also support instance retrieval based on conjunctive queries to expressive description logics such as $\mathcal{ALCNH}_{R^+}$ but the development of efficient algorithms for instance retrieval based on conjunctive queries are still an active research area. A first approach is described [Horrocks & Tessaris, 2000]. Conjunctive queries are related to Datalog-like specifications. However, in Datalog-like approaches reasoning is supported only about one specific model (the database) where for conjunctive queries of description logics all models are considered.

The FindUr approach [McGuinness, 1998] also relies on the CLASSIC system. FindUr uses so-called ontologies for supporting web browsing and search. FindUr focuses on the retrieval of web pages which are annotated with ontological notions. Agent communication in a description logic context has been considered by [Klusch, 1998]. In addition and complementary to our approach, game theory is used to control the "activity" of agents.

Schema integration and inter-schema knowledge modeling has been investigated by [Catarci & Lenzerini, 1993]. In contrast to the approach presented in this chapter, [Catarci & Lenzerini, 1993] does not rely on the assumption that the domains of different agents are identical. As a consequence, the notion of intensional inclusion

of different concepts is defined (rather than extension inclusions with GCIs). If desired, this could also be considered in our agent scenario. The decomposition of queries in a multidatabase scenario has been considered by [Cardiff et al., 1998]. Ideas taken from this context can also be applied in an agent scenario. Newest results on information integration with a description logic capturing the expressiveness of entity-relationship models are described in [Calvanese et al., 1998].

7.2.4 Summary

Agent-oriented problem solving has been analyzed from a formal knowledge representation point of view. Based on interschema knowledge a central agent, called broker, transforms inference problems such that they can be delegated to other agents preserving at least a sound overall inference algorithm. We have discussed examples involving instance retrieval and instance checking.

In the first example the motivation for delegation was to employ an agent which uses a less expressive description logic (the Program Agent) such that inferences can be computed more efficiently. In the second example an agent based on a description logic with different expressive power is employed for dealing with an instance checking problem that cannot be solved w.r.t. the knowledge represented by the broker. We have seen that the combined description logic is undecidable in general. Although the abstraction of the ABox and the refinement of the query concept as proposed in this section yields a sound inference algorithm based on delegation many inferences will be lost if the abstraction is too general and the refinement is too specific. Therefore, the knowledge of the broker must be a close approximation of the knowledge represented by the consulted specialist. The examples indicate that $\mathcal{ALCNH}_{R^+}$ is well-suited as a representation language for a broker. On the one hand, the logic is expressive enough to approximate the knowledge of other agents. On the other hand, with RACE there exists an implementation which guarantees quite encouraging average-case performance for practical reasoning.

The deficiencies of the approach have been indicated as well. As the combined language $\mathcal{ALCNH}_{R^+}$ and $\mathcal{ALCRP}(\mathcal{D})$ is not decidable in general, it is not possible to check whether any given input ABox is inconsistent w.r.t. the combined knowledge of the overall agent system. Thus, cases where query answering is not very useful due to an inconsistent input ABox might remain undetected. Although a specialist agent might conclude that the abstracted ABox is inconsistent, there are cases where the abstraction is consistent whereas the original is not. An idea to circumvent this problem in some cases might be to compute a refinement of the input ABox and to let the broker refuse query answering if one of the specialist acquaintances can prove that the refinement is inconsistent. Details of this approach have to be investigated in future work.

Implicit knowledge plays an important role in agent-based communication because, for expressive representation languages, there exists no "canonical form" that can be used as format for knowledge interchange. Comparisons between different representation structures have to be computed on a semantical basis. The work presented in this chapter discusses examples in the context of spatioterminological reasoning. Apparently, the key to adequate domain knowledge modeling is not only the definition of many ontological notions with class-subclass or part-whole relations. Instead, representation formalisms that capture the semantics of spatial object are required in order to avoid unintended models. The topological relations we have discussed in this chapter are part of the whole story. Obviously, reasoning facilities for spatial knowledge must be augmented with reasoning techniques for temporal knowledge (see section 6.3.2 for a first account in the context of spatiotemporal terminological reasoning). Whether the combination of spatial and temporal terminological reasoning can be adequately exploited in the agent-oriented scenario that we have investigated in the chapter is subject to future research.

7.3 Information Retrieval with Probabilistic Description Logics

In the previous section, we have seen that the task of similarity-based information retrieval can be split into three subtasks: First, the direct types of a finite set of individuals are computed yielding a finite set of concepts. Then, the LCS of these concepts is computed. Finally, by determining its instances the LCS concept is used as a retrieval concept. For the purpose of similarity-based information retrieval, the first task is fulfilled by the well-known realization inference service. The third subtask, determining the instances of the LCS concept, is accomplished by the instance retrieval inference service of the knowledge representation system.

In certain cases, computing the LCS of a set of concepts yields a very general concept. As a consequence, a large set of information items are retrieved resulting in an information flood if all items are displayed at once. Thus, at least a ranking is needed or we have to define a new operator for computing the commonalities between concepts. In this chapter, we pursue the second approach and define an LCS operator that takes additional domain knowledge into account.

The main contribution of this chapter is the proposal of a probabilistic LCS operation for a probabilistic extension of the DL $\mathcal{ALN}$ which has been introduced in [Koller et al., 1997] for the knowledge representation system P-Classic. The probabilistic LCS operator makes use of P-Classic's ability to model the degree of overlap between concepts. With the probabilistic LCS operator we investigate an example-based retrieval approach in which well known information retrieval techniques are integrated with formally well investigated inference services of DLs.

7.3.1 Preliminaries: P-CLASSIC

Despite its name P-Classic does not support all concept operators supported by the CLASSIC system. The description logic of P-CLASSIC is $\mathcal{ALN}$, i.e. negation only for concept names, value restrictions $\forall\, \mathsf{R}\,.\,\mathsf{C}$ and number restrictions $(\geq nR)$ and $(\leq nR)$. In the following we assume that $(=\ n\,R)$ as an abbreviation for $(\geq\ n\,R) \sqcap (\leq\ n\,R)$. The semantics of concepts is given in terms of an interpretation in the same way as specified above. Note that $\top$ and $\bot$ are expressible by $(\geq\ 0\,R)$ and $A \sqcap \neg A$, respectively.

A few definitions are required for the algorithms presented below.

Definition 73 (Depth) The *depth* of a concept is recursively defined as follows:

- If $C = A$, $C = \neg A$, $C = (\geq\ n\,R)$, or $(\leq\ n\,R)$, then $\mathit{depth}(C) := 0$.
- If $C = \forall\, R.C'$, then $\mathit{depth}(C) := 1 + \mathit{depth}(C')$.

Note that, in contrast to usual definitions of the concept depth, we define the depth of number restrictions as 0.

Definition 74 (Canonical form) Let $C_1, \ldots, C_m$ be concepts and $\{R_1, \ldots, R_M\}$ the set of all roles occurring at toplevel (nesting depth 0) in $C_1, \ldots, C_m$. Then C_i is in *canonical form* iff

$$C_i = \alpha_{i1} \sqcap \cdots \sqcap \alpha_{in_i} \sqcap \beta_{iR_1} \sqcap \cdots \sqcap \beta_{iR_{j_i}}$$

where $j_i \in \{0, \ldots, M\}$, α_{ik} is an atomic concept or negated atomic concept with no atomic concept appearing more than once and $\beta_{iR_j} = (\geq\ l_{iR_j}\ R_j) \sqcap (\leq\ m_{iR_j}\ R_j) \sqcap \forall\, R_j.C'_{iR_j}$ with C'_{iR_j} also being in canonical form.

It is easy to see that any concept can be transformed into an equivalent concept in canonical form in linear time.

The following example shows that the concept computed by the LCS is sometimes too general and, thus, might not always be a useful retrieval concept. Let sports-tool (ST), sports-broadcast (SB), team-sports-broadcast (TSB), individual-sports-broadcast(ISB), basketball (B), football (FB), and tennis-racket (TR) be atomic concepts, has-sports-tool an atomic role, and

$$\begin{aligned}
\text{basketball-broadcast (BB)} &:= \text{team-sports-broadcast} \sqcap (=\ 1\ \text{has-sports-tool}) \sqcap \\
&\quad \forall\, \text{has-sports-tool.basketball}, \\
\text{football-broadcast (FB)} &:= \text{team-sports-broadcast} \sqcap (=\ 1\ \text{has-sports-tool}) \sqcap \\
&\quad \forall\, \text{has-sports-tool.football, and} \\
\text{tennis-broadcast (TB)} &:= \text{individual-sports-broadcast} \sqcap (=\ 1\ \text{has-sports-tool}) \sqcap \\
&\quad \forall\, \text{has-sports-tool.tennis-racket}
\end{aligned}$$

be concepts. Subsequently, the concept abbreviations given in brackets are used. Let us consider a user interested in TV broadcasts similar to FB and BB. Then, computing the LCS of FB and BB would result in a useful retrieval concept: TSB$\sqcap$(= 1 has-sports-tool)$\sqcap\forall$ has-sports-tool.ST. However, there might be another user whose interests are expressed by FB and TB. The LCS computation then yields the retrieval concept $A :=$ SB $\sqcap$ (= 1 has-sports-tool) $\sqcap\forall$ has-sports-tool.ST denoting the set of *all* sports broadcasts with a sports tool. Since A is a very general concept, using A as a retrieval concept would result in a large amount of TV broadcasts, which might not be acceptable on the part of the user. A more suitable result would be to allow for $B :=$ TSB $\sqcap$ (= 1 has-sports-tool) $\sqcap\forall$ has-sports-tool.ST and $C :=$ ISB $\sqcap$ (= 1 has-sports-tool) $\sqcap\forall$ has-sports-tool.ST as alternative retrieval concepts. This is plausible because in Davis Cup matches, for instance, teams of tennis players compete against each other. Hence, in our intuition, there is a non-empty overlap between the concepts TSB and ISB which cannot be adequately quantified in $\mathcal{ALN}$. This is where P-Classic comes into play. In order to model the degree of overlap between concepts by probabilities, the knowledge representation system P-Classic was introduced in [Koller et al., 1997]. The DL underlying P-Classic is a probabilistic extension of $\mathcal{ALN}$ augmented by functional roles (attributes).

One of the goals of P-Classic is to compute probabilistic subsumption relationships of the form $P(D|C)$ denoting the probability of an individual to be an instance of D given that it is an instance of C. In case $C \equiv \top$, we write $P(D)$. In order to fully describe a concept, its atomic concept components and the properties of number restrictions and universal role quantifications need to be described. Therefore, a set $\mathcal{P}$ of probabilistic classes (p-classes) is introduced describing a probability distribution over the properties of individuals conditioned on the knowledge that the individuals occur on the right-hand side of a role. Each p-class is represented by a Bayesian network and one of the p-classes $P^* \in \mathcal{P}$ is the root p-class. The root p-class describes the distribution over all individuals and all other p-classes describe the distribution over role successors assuming independence between distinct individuals. The Bayesian networks are modeled as directed acyclic graphs whose nodes represent atomic concepts, number restrictions $[Number(R)]$, and the p-class from which role successors are drawn $[PC(R)]$. In addition to P-Classic, we introduce extra nodes for negations of atomic concepts. Dependencies are used to model conditional probabilities and are modeled by edges in the network. For instance, for an individual, we can state the probability of this individual to be an instance of ISB under the condition that it is an instance of SB. The range of the variables of a node representing an atomic or negated atomic concept can be either *true* or *false* and for $Number(R)$ it is a subset of $\mathbb{N}$. In order to guarantee termination of the inference algorithm for computing $P(D|C)$, this subset must be finite. Thus, the number of

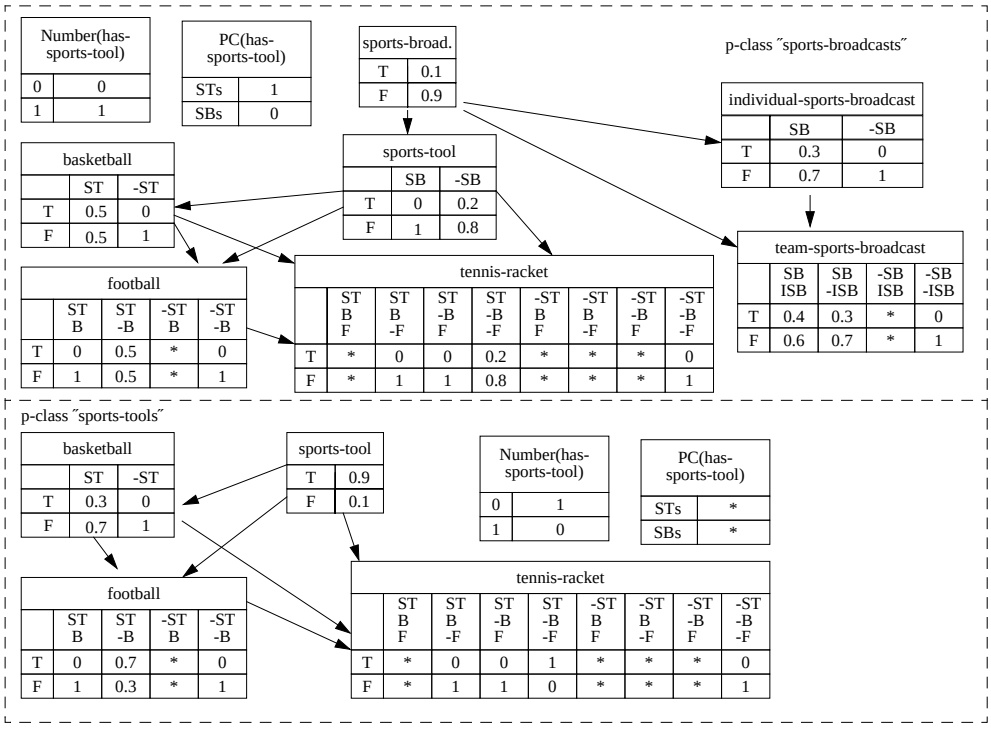


Figure 7.3: P-Classic TBox about sports broadcasts.

role successors for a role is bounded. The function $bound(R)$ indicates the maximum number of role successors for R. The range of a $PC(R)$ node is the set of p-classes $\mathcal{P}$ indicating the p-classes the R-successors are drawn from. The reason for introducing special nodes for negations of atomic concepts is that this extension enables us to evaluate expressions of the form $P(A \sqcap \neg A)$ as 0 which will be a necessary property subsequently. In order to demonstrate the advantages of the probabilistic LCS operator, we will now create a P-Classic TBox with overlapping concepts. Figure 7.3.1 shows a knowledge base about sports broadcasts enriched by probability information. For instance, it is stated that a broadcast is considered to be about team-sports (TSB) with probability 0.3 given that it is a broadcast about sports (SB) but no individual-sports (-ISB). Two p-classes are represented. The concept sports-broadcasts is the root p-class and the role successors for the role has-sports-tool are drawn from the p-class sports-tools. For each concept C, the probability $P_{P^*}(C)$ with which an individual is an instance of C can then be computed by a standard inference algorithm for Bayesian networks. For example, the probability of $P_{P*}(\mathsf{TSB} \sqcap (= 1\,\mathsf{has\text{-}sports\text{-}tool}) \sqcap \forall \mathsf{has\text{-}sports\text{-}tool}.\mathsf{B})$ is computed by setting the nodes for TSB and B to *true*, $Number(\mathsf{has\text{-}sports\text{-}tool}) = 1$, and $PC(\mathsf{has\text{-}sports\text{-}tool}) = \mathsf{STs}$. By Bayesian network propagation we yield a value of 0.015. With the formalism for

computing expressions of the form $P(D|C)$ it is possible to express the degree of overlap between C and D by a probability.

Based on the probabilistic description logic summarized in this section, it is possible to define a probabilistic LCS operator which takes into account the degree of overlap between concepts.

7.3.2 A Probabilistic Extension of Example-Based Retrieval

Intuitively, given concepts $C_1, \ldots, C_m$, the key idea is to allow those concepts for candidates of a probabilistic least common subsumer (PLCS) of $C_1, \ldots, C_m$ which have a non-empty overlap with $C_1, \ldots, C_m$. In order to keep the set of candidates finite, we consider only concepts whose depth is not larger than $\max\{depth(C_i)|i \in \{1, \ldots, m\}\}$. From the viewpoint of information retrieval this is no severe restriction, since in practical applications deeply nested concepts usually do not have any relevant individuals as instances (e.g., the concept $\mathsf{FB} \sqcap \forall \mathsf{has\text{-}sports\text{-}tool}.\forall \mathsf{has\text{-}sports\text{-}tool}.\mathsf{F}$ in our example).

Definition 75 Let $C_1, \ldots, C_m$ be $\mathcal{ALN}$ concepts and P^* the root p-class of a P-Classic TBox. Then we define the set of *PLCS concept candidates* of $C_1, \ldots, C_m$ as

$$\begin{aligned} Can(C_1, \ldots, C_m) \;:=\; & \{E | P_{P^*}(E \sqcap C_1) > \theta \wedge \cdots \wedge P_{P^*}(E \sqcap C_m) > \theta \wedge \\ & depth(E) \leq \max\{depth(C_i)|i = 1, \ldots, m\}\}. \end{aligned}$$

Definition 75 induces the following observation.

Proposition 76 Let $C_1, \ldots, C_m$ be $\mathcal{ALN}$ concepts. Then, in the worst case, the cardinality of $Can(C_1, \ldots, C_m)$ is exponential in m.

Proof. Given a P-Classic TBox in which $C_1, \ldots, C_m$ are all atomic concepts with $\forall i, j \in \{1, \ldots, m\} : P(C_i \sqcap C_j) > \theta$, we can bound $\sharp Can(C_1, \ldots, C_m)$ by the exponential function 2^m. □

The threshold θ can be equal to 0 but, since the number of candidates is exponential in the worst case, in practice a value should be chosen such that runtimes are acceptable.

In the next step, we want to measure the effectiveness of using a certain PLCS candidate for retrieval. It will be helpful to be able to express the probability of an individual to be an instance of a concept disjunction. Since this language operator is not contained in $\mathcal{ALN}$, we use the following definition which is essentially taken from [Rohatgi, 1976].

Definition 77 Let $C_1, \ldots, C_m$ be $\mathcal{ALN}$ concepts. Then we define

$$\begin{aligned} P(C_1 \sqcup \cdots \sqcup C_m) \;:=\; & (-1)^2 \sum_{k=1,\ldots,m} P(C_k) \;+\; (-1)^3 \sum_{k_1<k_2} P(C_{k_1} \sqcap C_{k_2}) \;+ \\ & (-1)^4 \sum_{k_1<k_2<k_3} P(C_{k_1} \sqcap C_{k_2} \sqcap C_{k_3}) + \;\ldots \\ & +(-1)^{m+1} P(C_1 \sqcap \cdots \sqcap C_m). \end{aligned}$$

It should be noted that by Definition 77 we do not extend the syntax of the underlying DL.

Proposition 78 Let $C_1, \ldots, C_m$ be concepts. Then computing $P(C_1 \sqcup \cdots \sqcup C_m)$ is exponential in m.

The proof is obvious and is omitted here.

In many retrieval environments, it is customary to use two real numbers: recall and precision. Both values indicate the quality of a concept E to function as an appropriate PLCS. By these measures the qualities of potential PLCSs can be compared to one another. The comparison will be formalized by the notion of dominance between triples $(E, r_{E,C_1,\ldots,C_m}, p_{E,C_1,\ldots,C_m})$ and $(E', r_{E',C_1,\ldots,C_m}, p_{E',C_1,\ldots,C_m})$.

Definition 79 (Recall) Let E and $C_1, \ldots, C_m$ be $\mathcal{ALN}$ concepts. Then we define the *recall* of E's w.r.t. $C_1, \ldots, C_m$ as

$$r_{E,C_1,\ldots,C_m} := P(C_1 \sqcup \cdots \sqcup C_m | E) = \frac{P(E \sqcap (C_1 \sqcup \cdots \sqcup C_m))}{P(E)}.$$

According to this definition, the larger the recall measure of a concept E, the more specific it is w.r.t. probabilistic subsumption of $C_1, \ldots, C_m$. For a concept E, a perfect recall is yielded iff $r_{E,C_1,\ldots,C_m} = 1$. For example, if E is a PLCS candidate and A an atomic concept such that $A \sqsubseteq E$, then $r_{E,A,\neg A} = 1$. Unlike in the definition of the (crisp) LCS, a concept expression does not necessarily need to subsume $C_1, \ldots, C_m$ (completely) in order to be a PLCS candidate. This motivates the introduction of the precision measure.

Definition 80 (Precision) Let E and $C_1, \ldots, C_m$ be $\mathcal{ALN}$ concepts. Then we define E's *precision* of $C_1, \ldots, C_m$ as

$$p_{E,C_1,\ldots,C_m} := P(E | C_1 \sqcup \cdots \sqcup C_m) = \frac{P(E \sqcap (C_1 \sqcup \cdots \sqcup C_m))}{P(C_1 \sqcup \cdots \sqcup C_m)}.$$

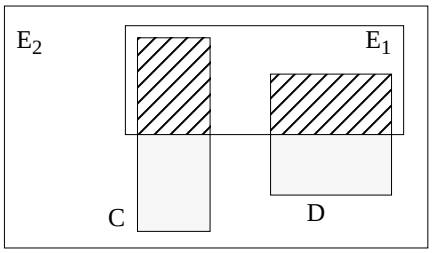


Figure 7.4: Scenario of four concepts illustrating the meaning of "recall" and "precision".

The precision measures the probability with which a randomly chosen individual, which is an instance of any of the C_i, $i \in \{1, \ldots, m\}$, is also an instance of the PLCS candidate E. As a consequence of Definition 80, if $E = lcs(C_1, \ldots, C_m)$, we have $p_{E,C_1,\ldots,C_m} = 1$.

Figure 7.3.2 illustrates the meaning of both measures given four concepts represented as areas in the 2D space. The recall of E_1, $r_{E_1,C,D}$, corresponds to the ratio of the size of the hatched area and the size of E_1. E_1's precision, $p_{E_1,C,D}$, is the ratio of the size of E_1 and the size of the union of E_1, C, and D. Given the appropriate values for E_2 we see that $r_{E_2,C,D}$ is smaller than $r_{E_1,C,D}$ but $p_{E_2,C,D}$ is larger than $p_{E_1,C,D}$.

Proposition 81 Let E and $C_1, \ldots, C_m$ be $\mathcal{ALN}$ concepts. Then, computing $r_{E,C_1,\ldots,C_m}$ and $p_{E,C_1,\ldots,C_m}$ takes time exponential in the length of $E, C_1, \ldots, C_m$.

Proof. Since $P(E \sqcap (C_1 \sqcup \cdots \sqcup C_m)) = P(E) - (P(E \sqcup C_1 \sqcup \cdots \sqcup C_m) - P(C_1 \sqcup \cdots \sqcup C_m))$, the claim follows from Proposition 78. □

With the above considerations, we will define the set of PLCSs of concepts $C_1, \ldots, C_m$ as a set of triples where the first component is a concept $E \in Can(C_1, \ldots, C_m)$ and the other components are E's recall and precision. In a concrete application, a user should be able to specify minimum values for at least one of the measures that he is willing to accept. For example, he could specify a recall of 0.8 preventing him from obtaining too general PLCS concepts and, thus, restricting the amount of retrieved data.

With the notion of dominance between candidates we can define the set of probabilistic least common subsumers.

Definition 82 (Dominance) Let E, E' and $C_1, \ldots, C_m$ be $\mathcal{ALN}$ concepts. Then $(E, r_{E,C_1,\ldots,C_m}, p_{E,C_1,\ldots,C_m})$ *dominates* $(E', r_{E',C_1,\ldots,C_m}, p_{E',C_1,\ldots,C_m})$ iff $(r_{E,C_1,\ldots,C_m} > r_{E',C_1,\ldots,C_m}) \wedge (p_{E,C_1,\ldots,C_m} > p_{E',C_1,\ldots,C_m})$.

Definition 83 (Probabilistic Least Common Subsumers) Let $C_1, \ldots, C_m$ be $\mathcal{ALN}$ concepts. Then we define the set of *probabilistic least common subsumers* of $C_1, \ldots, C_m$ as

$$\begin{aligned} p\text{-}lcs(C_1, \ldots, C_m) \quad := \quad & \{(E, r_{E,C_1,\ldots,C_m}, p_{E,C_1,\ldots,C_m}) \in Can(C_1, \ldots, C_m) \times \mathbb{R} \times \mathbb{R} | \\ & \neg\exists(E', r_{E',C_1,\ldots,C_m}, p_{E',C_1,\ldots,C_m}) : \\ & (E', r_{E',C_1,\ldots,C_m}, p_{E',C_1,\ldots,C_m}) \text{ dominates} \\ & (E, r_{E,C_1,\ldots,C_m}, p_{E,C_1,\ldots,C_m})\}. \end{aligned}$$

$p\text{-}lcs(C_1, \ldots, C_m)$ is called *minimal* iff
$\forall(E, r_{E,C_1,\ldots,C_m}, p_{E,C_1,\ldots,C_m}), (E', r_{E',C_1,\ldots,C_m}, p_{E',C_1,\ldots,C_m}) \in p\text{-}lcs(C_1, \ldots, C_m) : E \not\equiv E'$.

In Definition 83 we formalize the ideas of Figure 7.3.2 conditioned on the general case of m concepts. When defining $p\text{-}lcs(C_1, \ldots, C_m)$ we consider only concepts with a non-empty overlap with each of the $C_1, \ldots, C_m$. We only accept triples with the best quality measures and, therefore, accept only dominating triples in $p\text{-}lcs(C_1, \ldots, C_m)$. From this definition we can derive the following statement.

Proposition 84 The set $p\text{-}lcs(C_1, \ldots, C_m)$ has the following properties:

(i) $p\text{-}lcs(C_1, \ldots, C_m)$ is finite.

(ii) Minimality: $(E, r_{E,C_1,\ldots,C_m}, p_{E,C_1,\ldots,C_m}) \in p\text{-}lcs(C_1, \ldots, C_m) \Longrightarrow$
$\forall i \in \{1, \ldots, m\} : P(E \sqcap C_i) > 0 \wedge \neg\exists(E', r_{E',C_1,\ldots,C_m}, p_{E',C_1,\ldots,C_m}) :$
$(r_{E',C_1,\ldots,C_m} > r_{E,C_1,\ldots,C_m}) \wedge (p_{E',C_1,\ldots,C_m} > p_{E',C_1,\ldots,C_m}) \wedge$
$(depth(E') \leq depth(E))$.

Proof. (i) is obvious since the maximum depth of the concepts in $p\text{-}lcs(C_1, \ldots, C_m)$ is limited by the maximum depth of the $C_1, \ldots, C_m$ and the number of concept components of $C_1, \ldots, C_m$ is finite ensuring the number of PLCS candidates to be finite. Hence, $p\text{-}lcs(C_1, \ldots, C_m)$ is finite as well. For $(E, r_{E,C_1,\ldots,C_m}, p_{E,C_1,\ldots,C_m}) \in p\text{-}lcs(C_1, \ldots, C_m)$, the fact that $P(E \sqcap C_i) > 0$, for all $i \in \{1, \ldots, m\}$, follows immediately by the definition of $Can(C_1, \ldots, C_m)$. $\neg\exists(E', r_{E',C_1,\ldots,C_m}, p_{E',C_1,\ldots,C_m}) : r_{E',C_1,\ldots,C_m} > r_{E,C_1,\ldots,C_m} \wedge p_{E',C_1,\ldots,C_m} > p_{E,C_1,\ldots,C_m} \wedge depth(E') \leq depth(E)$ also follows since $p\text{-}lcs(C_1, \ldots, C_m)$ contains only dominating triples $(E, r_{E,C_1,\ldots,C_m}, p_{E,C_1,\ldots,C_m})$ with $depth(E) \leq \max\{depth(C_i) | i \in \{1, \ldots, m\}\}$. □

The minimal set $p\text{-}lcs(C_1, \ldots, C_m)$ can be computed in three steps: First, the set of concepts which have a non-empty overlap with each of the $C_1, \ldots, C_m$ must be computed. Proposition 84 (i) states a necessary criterion for a corresponding algorithm to terminate since the set of concepts E which have a non-empty

overlap with each of the C_i is finite. Then, for each concept E in this set, we have to compute the parameters $r_{E,C_1,\ldots,C_m}$ and $p_{E,C_1,\ldots,C_m}$ and then build the set of dominant triples $p\text{-}lcs(C_1,\ldots,C_m)$. Proposition 84 (ii) guarantees that there is no relevant retrieval concept with better recall *and* precision than the corresponding measures of the triples in $p\text{-}lcs(C_1,\ldots,C_m)$. Finally, the minimal set $p\text{-}lcs(C_1,\ldots,C_m)$ must be determined. This can be done by successively eliminating a triple $(E, r_{E,C_1,\ldots,C_m}, p_{E,C_1,\ldots,C_m})$ from $p\text{-}lcs(C_1,\ldots,C_m)$ as long as the following condition holds:

$$\forall(E, r_{E,C_1,\ldots,C_m}, p_{E,C_1,\ldots,C_m}) \in p\text{-}lcs(C_1,\ldots,C_m):$$
$$\neg\exists(E', r_{E',C_1,\ldots,C_m}, p_{E',C_1,\ldots,C_m}) \in p\text{-}lcs(C_1,\ldots,C_m) \text{ with } E \equiv E'.$$

The necessary equivalence test can be performed by structural comparisons since the involved concepts are in canonical form. In general, a minimal $p\text{-}lcs(C_1,\ldots,C_m)$ is not unique since there is no rule stating which triple to eliminate in case two triples with equivalent concepts are present. However, in our similarity-based information retrieval application this is no problem because the sets of instances of equivalent concepts are equal.

Procedure 2 computes the set of PLCS candidates given concepts $C_1,\ldots,C_m$ and the TBox as a Bayesian network BN. In the first step, all atomic concepts and negated atomic concepts in the Bayesian network are collected in the set X_1 if there is a non-empty overlap with each of the $C_1,\ldots,C_m$. Computing the concept candidates for our example, *compute-concept-candidate*(FB, TB), we get $X_1 = \{\mathsf{SB}, \mathsf{TSB}, \mathsf{ISB}\}$. Secondly, we build the set of all conjunctions of concepts of X_1 which have a non-empty overlapping with each of the $C_1,\ldots,C_m$ including the ones consisting of only one conjunct. In our case, we yield $X_2 = \{\mathsf{SB}, \mathsf{TSB}, \mathsf{ISB}, \mathsf{SB} \sqcap \mathsf{TSB}, \mathsf{SB} \sqcap \mathsf{ISB}, \mathsf{ISB} \sqcap \mathsf{TSB}, \mathsf{SB} \sqcap \mathsf{TSB} \sqcap \mathsf{ISB}\}$. In the next part of the algorithm, we collect all number restrictions having a non-empty overlap with each of the $C_1,\ldots,C_m$ in the set X_3. Since the maximum number of role successors is bounded, we can guarantee finiteness of X_3. Let X be an abbreviation for $(=\ 1\ \mathsf{has\text{-}sports\text{-}tool})$ and Y an abbreviation for $(\geq\ 0\ \mathsf{has\text{-}sports\text{-}tool}) \sqcap (\leq\ 1\ \mathsf{has\text{-}sports\text{-}tool})$. Then, in our example, we have $X_3 = \{X, Y\}$. Subsequently, for all roles R_i and all $\forall$-quantifications occurring in $C_1,\ldots,C_m$ and involving R_i, we add those concepts to X_4 which have a non-empty overlap with each of the R_i quantifiers ($C'_1,\ldots,C'_m$ in the algorithm). In our example, we compute $X_4 := \{\mathsf{ST}\}$. Now, in X_5 we collect all conjunctions of number restrictions from X_3 involving role R and $\forall\, R.D$ where D is a concept overlap with R's quantifiers $C'_1,\ldots,C'_m$. Let X' be an abbreviation for $X \sqcap \forall\, \mathsf{has\text{-}sports\text{-}tool}.\mathsf{ST}$ and Y' an abbreviation for $Y \sqcap \forall\, \mathsf{has\text{-}sports\text{-}tool}.\mathsf{ST}$. Then, in our example, we have $X_5 = \{X', Y'\}$. In X_6, we collect the conjunctions of elements of X_5 over all occurring roles if a conjunction has a non-empty overlap with each of the $C_1,\ldots,C_m$. Since, in our example, we have only one role, we get $X_6 = X_5$. Finally, the re-

sults in X_2 (conjunctions of atomic and negated atomic concepts) and the ones in X_6 (conjunctions of number restrictions and $\forall$-quantifications) are combined into X_7 which is returned by the algorithm. In our example, X_7 consists of 21 concepts from which we will only list the ones which are unique w.r.t. to equivalence: $\{E_1, \ldots, E_{12}\} = \{\mathsf{SB}, \mathsf{TSB}, \mathsf{ISB}, \mathsf{TSB} \sqcap \mathsf{ISB}, \mathsf{SB} \sqcap X', \mathsf{TSB} \sqcap X', \mathsf{ISB} \sqcap X', \mathsf{TSB} \sqcap \mathsf{ISB} \sqcap X', \mathsf{SB} \sqcap Y', \mathsf{TSB} \sqcap Y', \mathsf{ISB} \sqcap Y', \mathsf{TSB} \sqcap \mathsf{ISB} \sqcap Y'\}$ as desired.

Theorem 85 For concepts $C_1, \ldots, C_m$ and a Bayesian network BN representing a P-Classic TBox, algorithm *compute-concept-candidates* returns the set $Can(C_1, \ldots, C_m)$.

Proof. We give only a sketch of the proof. Algorithm *compute-concept-candidates* terminates because the maximum number of iterations is bounded by the maximum depth of $C_1, \ldots, C_m$. It is sound since every output concept has a non-empty overlap with $C_1, \ldots, C_m$. It is also complete because the algorithm recursively checks all possible concepts resulting from the concept-forming operators of $\mathcal{ALN}$ for a non-empty overlap with $C_1, \ldots, C_m$. □

The set of concept candidates computed by Algorithm 2 can easily be transformed into a set in which all pairs of concepts are not equivalent. Therefore, later no additional algorithm for transforming $p\text{-}lcs(C_1, \ldots, C_m)$ into a minimal $p\text{-}lcs(C_1, \ldots, C_m)$ will be necessary. Now recall and precision must be determined for each candidate by means of the formulae given in Definitions 79 and 80. This can be done straightforwardly by algorithms taking concepts E and $C_1, \ldots, C_m$ as input parameters and returning $r_{E,C_1,\ldots,C_m}$ and $p_{E,C_1,\ldots,C_m}$, respectively. The set $p\text{-}lcs(C_1, \ldots, C_m)$ contains only those triples whose quality measures dominate those of other triples.

Algorithm 3 computes the largest subset of dominant triples of $\{(E_1, r_{E_1,C_1,\ldots,C_m}, p_{E_1,C_1,\ldots,C_m}), \ldots, (E_n, r_{E_n,C_1,\ldots,C_m}, p_{E_n,C_1,\ldots,C_m})\}$. In the example, we get $p\text{-}lcs(\mathsf{FB}, \mathsf{TB}) = \{(\mathsf{SB} \sqcap X', 0.22, 1), (\mathsf{SB} \sqcap Y', 0.22, 1), (\mathsf{TSB} \sqcap X', 0.24, 0.354), (\mathsf{TSB} \sqcap Y', 0.24, 0.354), (\mathsf{ISB} \sqcap X', 0.26, 0.345), (\mathsf{ISB} \sqcap Y', 0.26, 0.345)\}$. As a result we get six possible retrieval concepts. $\mathsf{SB} \sqcap X'$ is the (crisp) LCS of FB and TB. Naturally, this concept has a precision of 1.0 since, according to Definition 68, $lcs(\mathsf{FB}, \mathsf{TB})$ is a concept which (completely) subsumes FB and TB. Alternatively, the result suggests the use of $\mathsf{TSB} \sqcap X'$ or $\mathsf{ISB} \sqcap X'$ as retrieval concepts. Both concepts have a better recall measure, and using them for retrieval results in a smaller set of information items. On the other hand, $\mathsf{TSB} \sqcap X'$ and $\mathsf{ISB} \sqcap X'$ have a worse precision measure than $\mathsf{SB} \sqcap X'$. Hence, the probability of meeting an individual which does not incorporate the commonalities represented by the concepts FB and TB is higher. The three concepts involving Y' have the same quality measures than the ones involving X'. The reason is that from our P-Classic TBox it follows that $P(Number(\mathsf{has\text{-}sports\text{-}tool}) = 0) = 1.0$, i.e. we do not need to consider them.

Procedure 2 compute-concept-candidates($C_1, \ldots, C_m, BN$)

Initialization: $X_1, \ldots, X_7 := \emptyset$, $pclass := P^*$, $d := max\{depth(C_1), \ldots, depth(C_m)\}$
  for $D \in$ *literals* (T-Box) **do**
    insert D into X_1 if $\forall i \in \{1, \ldots, m\} : P_{pclass}(D \sqcap C_i) > 0$
  end for
  for $E := D_1 \sqcap \ldots \sqcap D_k \in 2^{X_1}$ **do**
    insert E into X_2 if $\forall i \in \{1, \ldots, m\} : P_{pclass}(E \sqcap C_i) > 0$
  end for
  for $R \in$ *roles* $(\{C_1, \ldots, C_m\})$ **do**
    for $(i, j) \in$ *number-restrictions* $(R, \{C_1, \ldots, C_m\})$ **do**
      insert $(\geq\ l\ R) \sqcap (\leq\ u\ R)$ into X_3
      such that $0 \leq l \leq j \wedge u \leq i \leq bound(R) \wedge l \leq u$
    end for
    for $\{C_1^{d-1}, \ldots, C_m^{d-1}\} \in$ *all-quantification* $(R, \{C_1, \ldots, C_m\})$ **do**
      for $rpclass \in$ *relevant-p-classes* $(\{C_1^{d-1}, \ldots, C_m^{d-1}\}, \{C_1, \ldots, C_m\})$ **do**
        insert results of
        compute-concept-candidates($\{C_1^{d-1}, \ldots, C_m^{d-1}\}$, $rpclass$)
        into X_4
      end for
    end for
    for $E := A \sqcap \forall\ R.B : A \in X_3, B \in X_4$ **do**
      insert E into X_5
    end for
  end for
  for $E := D_1 \sqcap \ldots \sqcap D_k \in 2^{X_5}$ **do**
    insert E into X_6 if $\forall i \in \{1, \ldots, m\} : P_{pclass}(E \sqcap C_i) > 0$
  end for
  for $E := A \sqcap B : A \in X_2, B \in X_6$ **do**
    insert E in X_7 if $\forall i \in \{1, \ldots, m\} : P_{pclass}(E \sqcap C_i) > 0$
  end for
  return $X_2 \cup X_7$

Procedure 3 compute-minimal-plcs($(E_1, r_1, p_1), \ldots, (E_n, r_n, p_n)$)

  $p\text{-}lcs(C_1, \ldots, C_m) := sort(((E_1, r_1, p_1), \ldots, (E_n, r_n, p_n)), p_i)$
  for $i = 1$ to n **do**
    eliminate all (E', r', p') from $p\text{-}lcs(C_1, \ldots, C_m)$ with $r' < r_i$ and $p' < p_i$
  end for.

Theorem 86 Let $C_1, \ldots, C_m$ be $\mathcal{ALN}$ concepts. Then, in the worst case, computing $p\text{-}lcs(C_1, \ldots, C_m)$ takes time exponential in m.

Proof. This result follows from Proposition 76 since computing the set of PLCS candidates of $C_1, \ldots, C_m$ is a subtask of computing $p\text{-}lcs(C_1, \ldots, C_m)$. □

Propositions 78, 76, and 81 show the sources of complexity for the presented inference task. Due to the subterms $P(C_1 \sqcup \cdots \sqcup C_m)$ and $P(E \sqcap (C_1 \sqcup \cdots \sqcup C_m))$ occurring in Definitions 79 and 80, the computation of the precision and the recall measure take time exponential in the number of m. Also the computation of the set of PLCS candidates takes time exponential in the number of concepts. In practice, however, the exponential behavior of the computation comes into effect only for knowledge bases with many overlapping concepts. Thus, when building a TBox, the number of concept overlaps should be kept small and the threshold θ should be set appropriately.

7.3.3 Summary

In this section, we contributed to the problem of similarity-based information retrieval on the basis of the DL $\mathcal{ALN}$. It is shown that in certain cases the computation of commonalities with the (crisp) LCS operation yields too general retrieval concepts which can result in an information flood in a retrieval context. In order to circumvent this problem, we introduced a probabilistic LCS for a probabilistic extension of the DL $\mathcal{ALN}$. It is proved that the retrieval concepts provided by this operation are in some sense optimal and can be used as an alternative to retrieval concepts computed by a crisp LCS operation. By demonstrating the performances of the PLCS operator with an example we showed that meaningful retrieval results can be achieved with this operator. In the retrieval approach we integrated known information retrieval techniques with formally investigated inference services of DLs.

It has been shown that, in the information system scenario, it is even possible to successfully apply a theory for integrating probabilistic and logical representation formalisms such that a probabilistic abstraction operator for information retrieval applications could be implemented (see [Kaplunova, 1999] for details on the implementation).

Chapter 8

Conclusion

We have discussed the application of description logics to information processing tasks of different areas. Applications range from verification of specifications for telecommunication systems over support for the development of very large ontologies in a bio-informatics domain to deductive information systems in an agent-oriented retrieval scenario. Description logics constitute the common basis of many projects in a lively area of research. We conclude with an assessment of what problems have been solved in this Habilitation Thesis and present some apparent extensions of the work presented here.

8.1 Assessment

For modeling application problems as inference problems, tableaux calculi for expressive description logics such as $\mathcal{ALCNH}_{R^+}$, $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$ and $\mathcal{ALCRP}(\mathcal{D})$ had to be developed. The development of extended and new optimization techniques and their empirical evaluation with the DL system RACE shows that significant practical problems can be represented as inference problems. We have emphasized the importance of sound and complete (and terminating) TBox and ABox reasoning for solving subproblems of applications. The advantage is that with this solid theoretical foundation new applications with extended functionaliy can be developed with less effort.

The research summarized in this Habilitation Thesis complements other approaches exploring the description logic "landscape" from a complexity theory point of view [Donini et al., 1997a]. The detailed overviews over theoretical description logic research given in [Baader, 1999] and [Baader & Sattler, 2000] are not to be repeated here.

With the RACE system a very powerful representation and inference system has been developed. Much of the practical work is inspired by the landmark results on

optimization strategies presented in [Horrocks, 1997]. The architecture of RACE is tailored towards application problems with TBoxes and ABoxes and provides new optimization techniques such that larger application projects can use description logic systems which are based on sound and complete algorithms.

As the empirical investigations indicate, there is also price to pay for modeling with expressive languages. For specific combinatorial problems encoded as consistency tests for certain kinds of logical formulae, specialized inference systems are slightly faster than RACE but can be used only if the problems are formalized using the modal logic K_m (see e.g. [Giunchiglia et al., 1999]). For instance, number restrictions come at the price of more complex clash tests which slow down the performance of RACE for pure propositional or modal logic inference problems quite a bit (although RACE adapts its clash testing algorithms to the language constructs used in a certain problem). However, RACE is still one of the fastest systems available today and, since the development of optimized algorithms is a very active area of research, newer versions of RACE will support optimization techniques for even more powerful language features. The integration of new features such as inverse roles into the RACE architecture is possible but it becomes more and more complex to safely integrate new features into optimized inference algorithms. Thus, currently, there is some kind of gap between what is proven at the calculus level and what has to be implemented in order to achieve adequate performance using optimized search and caching strategies.

Another part of this work considers spatioterminological reasoning. For the important combination of spatial and terminological reasoning a theory has been proposed with $\mathcal{ALCRP}(\mathcal{RCC})$. Meanwhile other authors have published complexity analyses [Lutz, 1999b] and have developed a prototypical implementation [Turhan, 1998]. Nevertheless, there does not exist an optimized implementation for $\mathcal{ALCRP}(\mathcal{D})$ (and $\mathcal{ALC}(\mathcal{D})$ as well) that is comparable to the performance of RACE for $\mathcal{ALCNH}_{R^+}$. The results about RACE should provide the basis for developing adequate algorithms for $\mathcal{ALCRP}(\mathcal{D})$ that can be practically implemented. First results on this topic have recently been discussed in [Turhan & Haarslev, 2000; Turhan, 2000]. However, although some aspects of spatial reasoning can be captured with $\mathcal{ALCRP}(\mathcal{RCC})$, it is only a first step.

In order to achieve soundness, completeness and termination of the ABox consistency algorithm, syntactic restrictions have to be imposed on the formulae used in $\mathcal{ALCRP}(\mathcal{D})$ knowledge bases. Furthermore, the DL part of $\mathcal{ALCRP}(\mathcal{D})$ does not offer number restrictions, and role hierarchies and transitive roles can only be introduced via concrete domains. An important insight was that all interesting concrete domains provide enough expressivity that syntactic restrictions are indeed necessary to ensure termination.

The application examples discussed in this thesis demonstrate that the results pro-

vide a foundation for information systems that can answer queries in a distributed scenario and by reasoning about conceptual and spatial information.

8.2 Outlook

We have emphasized that qualitative spatial relations and the RCC-8 calculus are only taken as an example for dealing with some aspects of spatial reasoning. Topological relations are only one kind of knowledge relevant for spatial reasoning. For instance, reasoning about shape, position and orientation are to be investigated in the context of conceptual reasoning with description logics as well. In addition, the integration of quantitative knowledge into spatioterminological reasoning is an important area of research.

A generalization of the work on spatioterminological default reasoning could be investigated by extending description logics with autoepistemic operators [Donini et al., 1997b; Rosati, 1997]. With these operators even integrity conditions (restrictions on the data model rather than on the represented world) can be expressed. An investigation of these operators in the context of spatioterminological default reasoning could lead to fruitful results.

The idea of defining roles based on concrete domain predicates in $\mathcal{ALCRP}(\mathcal{D})$ and using a concrete domain consistency tester can be generalized. Rather than using an *external* reasoner and a concrete domain it might be possible to reason about properties of relations *within* the description logic formalism. In other words: it is interesting to investigate ways to integrate the semantics of spatial relations into the semantics of the description logic formalism itself. Research about this has just started [Wessel et al., 2000]. The new language is called $\mathcal{ALC}_{RA}$ and extends $\mathcal{ALC}$ with role axioms such that, for instance, the meaning of RCC-8 relations can be directly provided by specifying role axioms which, in turn, directly correspond to the composition table of RCC-8.

The language $\mathcal{ALC}_{RA}$ would be useful for enhanced reasoning about conceptual knowledge as well [Wessel et al., 2000]. This will be explained with a small example. Consider a man whose brother has a sister which, in turn, has a sister whose daughter is a computer science student. If we claim that all his nieces are indeed no computer science students, then it should be obvious that something is wrong. However, if we consider the $\mathcal{ALC}$ concept term $\mathsf{Man} \sqcap (\exists\, \mathsf{brother} \,.\, \exists\, \mathsf{sister} \,.\, \exists\, \mathsf{sister} \,.\, \exists\, \mathsf{daughter} \,.\, \mathsf{Computer\_science\_student}) \sqcap \forall\, \mathsf{niece} \,.\, \neg \mathsf{Computer\_science\_student}$ without representing the relationships between the roles, then the inherent inconsistency of the concept would not be detected.

The following role axioms are required in order to derive the inconsistency (the operator $\circ$ denotes role composition).

$$\textbf{brother} \circ \textbf{sister} \sqsubseteq \text{sister}$$
$$\textbf{sister} \circ \textbf{daughter} \sqsubseteq \text{niece}$$
$$\textbf{daughter} \circ \textbf{sister} \sqsubseteq \text{daughter}$$
$$\textbf{sister} \circ \textbf{sister} \sqsubseteq \text{sister}$$

However, if $\mathcal{ALC}_{RA}$ is decidable or not, is currently an open research question.

With the availability of expressive description logic inference systems it will become possible to extend the development of UML (Unified Modeling Language, e.g. [Page-Jones, 2000]). Based on description logics it might be possible to define a semantics for parts of UML such that UML cannot only be used for presentation or communication purposes during the design phase of a software project but – with an optimized inference system – can also be used for solving problems without code for algorithms being written manually. Furthermore, it might be possible to use a DL system for verifying UML state space models. Investigations for verifying state space models in UML with model checking techniques are described in [Lilius & Paltor, 1999c; Lilius & Paltor, 1999b; Lilius & Paltor, 1999a].

As we have argued, the logical approach pursued in this thesis provides a foundation for information systems with inference components. ABoxes are used to represent information about specific objects of a certain domain. However, as long as ABoxes (and TBoxes) have to be kept in main memory, the spectrum of possible applications is somewhat limited. Thus, it would be very interesting to investigate algorithms for implementing ABox persistency with transactions and rollback. It should be noted however, that revision in the context of logical representation formalism is still a research topic on its own. In the context of spatial knowledge, revision and retraction are definitely exciting areas for future research.

8.3 Résumé

In recent years, the research field covered in this Habilitation Thesis has seen many contributions from different groups and perspectives. The application examples presented in this thesis demonstrate that the results are important for knowledge representation and computer science in general. The intention behind the work in this field is to develop a representation "medium" that can be used for modeling in such a way that automatic inferences are supported and application (sub)problems can be automatically solved by invoking inference service. The insights gained by developing optimization algorithms convinced us that the services provided by automatic inference systems can hardly be provided by ad hoc software development within a

reasonable amount of time and with the necessary reliability. The key to the successful development of applications with description logics are the *inference services* rather than particular data structures for storing information. Expressive description logics and optimized inference systems are a necessary prerequisite for success in a practical context.

Formal inference systems in general and description logics in particular can be called a changing discipline. Previously, there was a gap between theoretical results and implemented modeling and inference systems. Implemented systems either used incomplete algorithms or supported only "weak" representation languages. Now, with the optimization techniques of today, it becomes clear that theoretical results about decidability and complexity for expressive description logics are directly relevant for practical work. Much has been achieved but still much has to be learned. I hope that this work stimulates new research results in the near future.

Bibliography

Abiteboul, S., Hull, R., & Vianu, V. (1995). *Foundation of Databases*. Addison-Wesley.

Abrett, G. & Burstein, M. (1987). The KREME knowledge editing environment. *Int. J. Man-Machine Studies*, 27:103–126.

Achilles, E., Hollunder, B., Laux, A., & Mohren, J. (1991). $\mathcal{KRIS}$: Knowledge Representation and Inference System – User Guide. Technical report, Deutsches Forschungszentrum für Künstliche Intelligenz GmbH, Document D91-14.

Aiello, L., Doyle, J., & Shapiro, S., editors (1996). *Fifth International Conference on Principles of Knowledge Representation, Cambridge, Mass., Nov. 5-8, 1996.*

Allen, J. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843.

Allen, J. (1991). The RHET system. *ACM SIGART Bulletin*, 2(3):1–7.

Allgayer, J. (1990). SB-ONE+ - dealing with sets efficiently. In *Proceedings ECAI-90*, pages 13–18.

Antoniou, G. (1997). *Nonmonotonic Reasoning*. MIT Press.

Areces, C., Bouma, W., & de Rijke, M. (1999). Description logics and feature interaction. In [Lambrix et al., 1999].

Baader, F. (1990). Terminological cycles in KL-ONE-based knowledge representation languages. In *Proceedings of the Eighth National Conference on Artificial Intelligence, AAAI-90*, pages 621–626, Boston (USA).

Baader, F. (1991). Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence, IJCAI-91*, pages 446–451, Sydney (Australia).

Baader, F. (1999). Logic-based knowledge representation. In Wooldridge, M. & Veloso, M., editors, *Artificial Intelligence Today, Recent Trends and Developments*, number 1600 in Lecture Notes in Computer Science, pages 13–41. Springer Verlag.

Baader, F., Buchheit, M., & Hollunder, B. (1996). Cardinality restrictions on concepts. *Artificial Intelligence*, 88(1–2):195–213.

Baader, F., Franconi, E., Hollunder, B., Nebel, B., & Profitlich, H. (1994). An empirical analysis of optimization techniques for terminological representation systems or: Making $\mathcal{KRIS}$ get a move on. *Applied Artificial Intelligence. Special Issue on Knowledge Base Management*, 4:109–132.

Baader, F. & Hanschke, P. (1991a). A scheme for integrating concrete domains into concept languages. In *Twelfth International Conference on Artificial Intelligence, Darling Harbour, Sydney, Australia, Aug. 24-30, 1991*, pages 452–457.

Baader, F. & Hanschke, P. (1991b). A scheme for integrating concrete domains into concept languages. Technical Report DFKI-RR-91-10, German Center for AI (DFKI).

Baader, F. & Hanschke, P. (1992). Extensions of concept languages for a mechanical engineering application. In Ohlbach, H., editor, *Proceedings, GWAI-92: Advances in Artificial Intelligence, 16th German Conference on Artificial Intelligence*, pages 132–143. Springer-Verlag, Berlin.

Baader, F. & Hollunder, B. (1990). $\mathcal{KRIS}$: Knowledge representation and inference system, system description. DFKI Technical Memo TM-90-03, Deutsches Forschungszentrum für Künstliche Intelligenz, Kaiserslautern.

Baader, F. & Hollunder, B. (1991a). $\mathcal{KRIS}$: Knowledge Representation and Inference System, system description. *ACM SIGART Bulletin*, 2(3):8–14. Also published as [Baader & Hollunder, 1990].

Baader, F. & Hollunder, B. (1991b). A terminological knowledge representation system with complete inference algorithms. In *Proceedings of the First International Workshop on Processing Declarative Knowledge*, volume 572 of *Lecture Notes in Computer Science*, pages 67–85, Kaiserslautern (Germany). Springer–Verlag.

Baader, F. & Hollunder, B. (1992). Embedding defaults into terminological knowledge representation formalisms. In [Nebel et al., 1992], pages 306–317.

Baader, F. & Hollunder, B. (1993). How to prefer more specific defaults in terminological default logic. In *Proc. IJCAI-93*, pages 669–674.

Baader, F. & Hollunder, B. (1995a). Embedding defaults into terminological representation systems. *J. Automated Reasoning*, 14:149–180.

Baader, F. & Hollunder, B. (1995b). Priorities on defaults with prerequisites, and their application in treating specificity in terminological default logic. *J. Automated Reasoning*, 15:41–68.

Baader, F., Hollunder, B., Nebel, B., Profitlich, H., & Franconi, E. (1992). An empirical analysis of optimization techniques for terminological representation systems, or: Making $\mathcal{KRIS}$ get a move on. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning, KR-92*, pages 270–281, Boston (USA).

Baader, F. & Küsters, R. (1998). Computing the least common subsumer and the most specific concept in the presence of cyclic $\mathcal{ALN}$-concept descriptions. In Herzog, O. & Günter, A., editors, *Proceedings of the 22nd Annual German Conference on Artificial Intelligence, KI-98*, volume 1504 of *Lecture Notes in Computer Science*, pages 129–140, Bremen, Germany. Springer–Verlag.

Baader, F. & Küsters, R. (2000). Matching in description logics with existential restrictions. In Cohn, A., Giunchiglia, F., & Selman, B., editors, *Proceedings of the Seventh International Conference on Knowledge Representation and Reasoning (KR2000)*, pages 261–272, San Francisco, CA. Morgan Kaufmann Publishers.

Baader, F., Küsters, R., & Molitor, R. (2000). Rewriting concepts using terminologies. In Cohn, A., Giunchiglia, F., & Selman, B., editors, *Proceedings of the Seventh International Conference on Knowledge Representation and Reasoning (KR2000)*, pages 297–308, San Francisco, CA. Morgan Kaufmann Publishers.

Baader, F., McGuinness, D., Nardi, D., & Patel-Schneider, P., editors (2001). *Description Logic Handbook*. Cambridge University Press.

Baader, F. & Molitor, R. (2000). Building and structuring description logic knowledge bases using least common subsumers and concept analysis. In *Proceedings of the 8th International Conference on Conceptual Structures (ICCS2000)*, Lecture Notes in Artificial Intelligence. Springer Verlag.

Baader, F. & Sattler, U. (1998). Description logics with concrete domains and aggregation. In Prade, H., editor, *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI-98)*, pages 336–340. John Wiley & Sons Ltd.

Baader, F. & Sattler, U. (1999). Expressive number restrictions in description logics. *Journal of Logic and Computation*, 9(3):319–350.

Baader, F. & Sattler, U. (2000). Tableau algorithms for description logics. In Dyckhoff, R., editor, *Proceedings of the International Conference on Automated Reasoning with Tableaux and Related Methods (Tableaux 2000)*, volume 1847 of *Lecture Notes in Artificial Intelligence*, pages 1–18, St Andrews, Scotland, UK. Springer-Verlag.

Baader, F. & Schlechta, K. (1993). A semantics for open normal defaults via a modified preferential approach. In *Proceedings of the European Conference on Symbolic and Quantitative Approaches to Reasoning und Uncertainty, (ECSQARU 93)*, volume 747 of *Lecture Notes in Computer Science*, pages 9–16, Granada (Spain). Springer-Verlag, New York.

Baader et al., F., editor (2000). *Proceedings of the International Workshop on Description Logics (DL'2000), August 17 - August 19, 2000, Aachen, Germany.*

Badea, L. & Niehuys-Cheng, S. (2000). Refining concepts in description logics. In [Baader et al., 2000], pages 31–44.

Bartelme, N. (1995). *Geoinformatik: Modelle, Strukturen, Funktionen.* Springer-Verlag, Berlin.

Bennett, B. (1994). Spatial reasoning with propositional logics. In [Doyle et al., 1994], pages 51–62.

Bennett, B., Isli, A., & Cohn, A. (1997). When does a composition table provide a complete and tractable proof procedure for a relational constraint language? In *Proceedings of IJCAI'97 workshop on Spatial and Temporal Reasoning.*

Borgida, A. (1995). Description logics in data management. *IEEE Transactions on Knowledge and Data Engineering*, 7(5):671–682.

Borgida, A., Brachman, R., McGuinness, D., & Resnick, L. (1989). CLASSIC: A structural data model for objects. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Portland, Oregon.*

Borgida, A. & Brachman, R. J. (1993). Loading data into description reasoners. In *Proc. ACM SIGMOD Conf. on Data Management*, pages 217–226, Washington, D.C.

Borgida, A., Isbell, C., & McGuinness, D. (1996). Reasoning with black boxes: Handling test concepts in classic. In Padgham et al., L., editor, *Proceedings of the International Workshop on Description Logics, Nov. 2-4, 1996, Cambridge, Massachusetts*, pages 87–91. AAAI Press, Menlo Park. Technical Report WS-96-05.

Borgida, A. & McGuinness, D. (1996). Asking queries about frames. In [Aiello et al., 1996], pages 340–349.

Borgida, A. & Patel-Schneider, P. (1994). A semantics and complete algorithm for subsumption in the CLASSIC description logic. *Journal of Artificial Intelligence Research*, 1:277–308.

Brachman, R. (1977). What's in a concept: Structural foundations for semantic networks. *International Journal of Man-Machine Studies*, 9:127–152.

Brachman, R. (1979). On the epistemological status of semantic networks. In Findler, N., editor, *Associative Networks: Representation and Use of Knowledge by Computers*, pages 3–50. Academic Press, New York.

Brachman, R. (1983). What IS-A is and isn't. *IEEE Computer*, 16(10):30–36.

Brachman, R. (1984). 'I lied about the trees': or, defaults and definitions in knowledge representation. *AI Magazine*, 6(3):80–93.

Brachman, R. (1992). "Reducing" CLASSIC to practice: Knowledge representation theory meets reality. In *Principles of Knowledge Representation and Reasoning, Third International Conference, Cambridge, Mass., Oct. 25-29, 1992*, pages 247–258.

Brachman, R., Fikes, R., & Levesque, H. (1983a). KRYPTON: A functional approach to knowledge representation. *IEEE Computer*, 16(10):67–73. A revised version appears in [Brachman & Levesque, 1985, 411–430].

Brachman, R., Fikes, R., & Levesque, H. (1983b). KRYPTON: Integrating terminology and assertion. In *Proc. of the Third National Conference on Artificial Intelligence, AAAI-83*, pages 31–35.

Brachman, R., Gilbert, V., & Levesque, H. (1985). An essential hybrid reasoning system: Knowledge and symbol level accounts of KRYPTON. In *Proc. Ninth International Conference on Artificial Intelligence, IJCAI-85, Los Angeles, CA*. Morgan Kaufmann, San Mateo.

Brachman, R. & Levesque, H. (1984). The tractability of subsumption in frame-based description languages. In *Proc. of the Fourth National Conference on Artificial Intelligence, AAAI-84*, pages 34–37.

Brachman, R. & Levesque, H., editors (1985). *Readings in Knowledge Representation*. Morgan Kaufmann, Los Altos.

Brachman, R., Levesque, H., & Reiter, R., editors (1989). *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning (KR'89), Toronto, Canada*. Morgan Kaufmann, San Mateo.

Brachman, R., McGuinness, D., Patel-Schneider, P., Resnick, L., & Borgida, A. (1991). Living with CLASSIC: When and how to use a KL-ONE-like language. In [Sowa, 1991], pages 401–456.

Brachman, R. & Schmolze, J. (1985). An overview of the KL-ONE knowledge representation system. *Cognitive Science*, pages 171–216.

Bresciani, P., Franconi, E., & Tessaris, S. (1995). Implementing and testing expressive description logics. In *Proceedings of DL'95, International Workshop on Description Logics*, pages 131–139, Rome (Italy).

Brill, D. (1994). *Loom Reference Manual, Version 2.0*. Marina del Rey.

Buchheit, M., Donini, F., & Schaerf, A. (1993a). Decidable reasoning in terminological knowledge representation systems. *Journal of Artificial Intelligence Research*, 1:109–138.

Buchheit, M., Donini, F. M., & Schaerf, A. (1993b). Decidable reasoning in terminological knowledge representation systems. *J. of Artificial Intelligence Research*, 1:109–138.

Buchheit, M., Klein, R., & Nutt, W. (1994). Configuration as model construction: The constructive problem solving approach. In Sudweeks, F. & Gero, J., editors, *Proc. 4th International Conference on Artificial Intelligence in Design, Lausanne (Switzerland)*. Kluwer, Dordrecht.

Buchheit, M., Klein, R., & Nutt, W. (1995). Constructive problem solving: A model construction approach towards configuration. Technical Report DFKI-TM-95-01, German Center for AI (DFKI).

Cadoli, M. (1995). *Tractable Reasoning in Artificial Intelligence*. Lecture Notes in Artificial Intelligence 941. Springer-Verlag, Berlin.

Calvanese, D., Giacomo, G. D., Lenzerini, M., Nardi, D., & Rosati, R. (1998). Description logic framework for information integration. In [Cohn et al., 1998], pages 2–13.

Cardiff, J., Catarci, T., & Santucci, G. (1998). Exploitation of interschema knowledge in a multidatabase system. In *Proc. of the 4th KRDB Workshop, Athens, Greece*. http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-8/.

Catarci, T. & Lenzerini, M. (1993). Representing and using interschema knoweledge in cooperative information systems. *Int. Journal of Intelligent and Cooperative Information Systems*, 2(4):375–398.

Caviness, B. & Johnson, J. (1998). *Quantifier Elimination and Cylindrical Algebraic Decomposition*. Springer-Verlag.

Chomicki, J. & Saake, G. (1998). *Logics for Databases and Information Systems*. Kluwer Academic Publishers.

Clarke, E. & Kurshan, R. (1996). Computer-aided verification. *IEEE Spectrum*, 33(6):61–67.

Cohen, W., Borgida, A., & Hirsh, H. (1992). Computing least common subsumers in description logics. In *Proceedings AAAI-92*, pages 754–760. AAAI Press/The MIT Press.

Cohn, A., Bennett, B., Gooday, J., & Gotts, N. (1997). Representing and reasoning with qualitative spatial relations. In [Stock, 1997], pages 97–134.

Cohn, A., Giunchiglia, F., & Selman, B., editors (2000). *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR'2000), Breckenridge, Colorado, USA, 2000*.

Cohn, T., Schubert, L., & Shapiro, S., editors (1998). *Proceedings of Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98), Trento, Italy, June 2-5, 1998*.

Cunis, R. (1991). Modellierung technischer Systeme in der Begriffshierarchie. In [Cunis et al., 1991], chapter 5.

Cunis, R., Günter, A., & Strecker, H., editors (1991). *Das PLAKON-Buch – Ein Expertensystemkern für Planungs- und Konfigurierungsaufgaben in technischen Domänen*, volume 266. Springer-Verlag.

Davis, M., Logemann, G., & Loveland, D. (1962). A machine program for theorem proving. *Communication of the ACM*, 5:394–397.

De Giacomo, G. & Lenzerini, M. (1996). TBox and ABox reasoning in expressive description logics. In [Aiello et al., 1996].

Donini, F., Lenzerini, M., Nardi, D., & Nutt, W. (1997a). The complexity of concept languages. *Information and Computation*, 134(1):1–58.

Donini, F., Lenzerini, M., Nardi, D., Nutt, W., & Schaerf, A. (1998). An epistemic operator for description logics. *Artificial Intelligence*, 100(1-2):225–274.

Donini, F., Lenzerini, M., Nardi, D., & Schaerf, A. (1996). Reasoning in description logics. In Brewka, G., editor, *Principles of Knowledge Representation*. CSLI Publications.

Donini, F. M., Lenzerini, M., Nardi, D., & Nutt, W. (1991a). The complexity of concept languages. In *Proceedings of the Second International Conference on the Principles of Knowledge Representation and Reasoning (KR-91)*, pages 151–162. Morgan Kaufmann, Los Altos. A detailed version appeared as DFKI Research Report RR-95-07, Kaiserslautern.

Donini, F. M., Lenzerini, M., Nardi, D., & Nutt, W. (1991b). Tractable concept languages. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence, IJCAI-91*, pages 458–463. Morgan Kaufmann, Los Altos. A detailed version appeared as DFKI Research Report RR-95-07, Kaiserslautern.

Donini, F. M., Lenzerini, M., Nardi, D., Nutt, W., & Schaerf, A. (1992). Adding epistemic operators to concept languages. In [Nebel et al., 1992], pages 342–353.

Donini, F. M., Lenzerini, M., Nardi, D., Nutt, W., & Schaerf, A. (1994). Queries, rules and definitions as epistemic sentences in concept languages. In Lakemeyer, G. & Nebel, B., editors, *The Theoretical Foundations of Knowledge Representation and Reasoning*, Lecture Notes in Artificial Intelligence LNAI 810, pages 113–132. Springer-Verlag.

Donini, F. M., Nardi, D., & Rosati, R. (1997b). Autoepistemic description logics. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence IJCAI-97*, pages 136–141.

Doyle, J., Sandewall, E., & Torasso, P., editors (1994). *Fourth International Conference on Principles of Knowledge Representation, Bonn, Germany, May 24-27, 1994*.

Drollinger, D. (1993). Intelligentes Backtracking in Inferenzsystemen am Beispiel Terminologischer Logiken (in German). Document D-93-21, Deutsches Forschungszentrum für Künstliche Intelligence, Germany.

Eschenbach, C. (1999). A predication calculus for qualitative spatial representations. In Freksa, C. & Mark, D., editors, *Proc. International Conference on Spatial Information Theory, COSIT '99, Stade, Germany*, pages 157–172. Springer-Verlag.

Fensel, D., Erdmann, M., & Studer, R. (1998). Ontobroker: The very high idea. In *Proceedings of the 11th International Flairs Conference (FLAIRS-98), Sanibal Island, Florida*.

Fikes, R. (1981). Klonetalk. In [Schmolze & Brachman, 1982]. Published as BBN Research Report 4842, Bolt Beranek and Newman Inc., June 1982.

Fischer, M. (1992). The integration of temporal operators into a terminological representation system. Technical report, Technical University of Berlin, Computer Science Department, Project KIT, KIT-Report 92.

Fitting, M. (1996). *First-Order Logic and Automated Theorem Proving*. Springer-Verlag, Berlin.

Frank, A. (1997). Spatial ontology. In [Stock, 1997], pages 135–153.

Frank, A. (1998). Different types of "time" in GIS. In Egenhofer, O. & Golledge, R., editors, *Spatial and Temporal Reasoning in Geographic Information Systems*, pages 40–62. Oxford University Press.

Freeman, J. (1995). *Improvements to propositional satisfiability search algorithms*. PhD thesis, University of Pennsylvania, Computer and Information Science.

Garcia-Molina, H., Papakonstantinou, Y., Quass, D., Rajaraman, A., Sagiv, Y., Ullman, J., Vassalos, V., & Widom, J. (1997). The TSIMMIS approach to mediation: Data models and languages. *Journal of Intelligent Information Systems*.

Genesereth, M. & Nilsson, N. (1987). *Logical foundations of artificial intelligence*. Morgan-Kaufmann.

Gerevini, A. (1997). Reasoning about time and actions in artificial intelligence: Major issues. In [Stock, 1997], pages 43–70.

Ginsberg, M. (1993). Dynamic backtracking. *Journal of Artificial Intelligence Research*, 1:25–46.

Ginsberg, M. & McAllester, D. (1994). Gsat and dynamic backtracking. In [Doyle et al., 1994], pages 226–237.

Giunchiglia, E., Giunchiglia, F., & Tacchella, A. (1999). *SAT, KSATC, DLP and TA: a comparative analysis. In [Lambrix et al., 1999], pages 110–114. Also published as CEUR Workshop Proceedings, Volume 22, http://SunSITE.Informatik.RWTH-Aachen.DE/Publications/CEUR-WS/Vol-22/.

Giunchiglia, E. & Tacchella, A. (2000). A subset-matching size-bounded cache for satisfiability of modal logics. In *Proceedings International Conference Tableaux'2000*, pages 237–251. Springer-Verlag.

Giunchiglia, F. & Sebastiani, R. (1996). A SAT-based decision procedure for $\mathcal{ALC}$. In [Aiello et al., 1996], pages 304–314.

Günter, A., editor (1995). *Wissensbasiertes Konfigurieren – Ergebnisse aus dem Projekt PROKON*. infix, Sankt Augustin.

Haarslev, V. (1998). A fully formalized theory for describing visual notations. In Marriott, K. & Meyer, B., editors, *Visual Language Theory*, pages 261–292. Springer-Verlag, Berlin.

Haarslev, V., Horrocks, I., Möller, R., & Patel-Schneider, P. (1999a). DL Benchmark Suite. Available at URL http://kogs-www.informatik.uni-hamburg.de/~moeller/dl-benchmark-suite.html.

Haarslev, V., Lutz, C., & Möller, R. (1998). Foundations of spatioterminological reasoning with description logics. In [Cohn et al., 1998], pages 112–123.

Haarslev, V., Lutz, C., & Möller, R. (1999b). A description logic with concrete domains and a role-forming predicate operator. *Journal of Logic and Computation*, 9(3):351–384.

Haarslev, V. & Möller, R. (1997a). SBox: A qualitative spatial reasoner—progress report. In Ironi, L., editor, *11th International Workshop on Qualitative Reasoning, Cortona, Tuscany, Italy, June 3-6, 1997, Pubblicazioni N. 1036, Istituto di Analisi Numerica C.N.R. Pavia (Italy)*, pages 105–113.

Haarslev, V. & Möller, R. (1997b). Spatioterminological reasoning: Subsumption based on geometrical inferences. In [Rousset et al., 1997], pages 74–78.

Haarslev, V. & Möller, R. (1999a). Applying an $\mathcal{ALC}$ ABox consistency tester to modal logic SAT problems. In Murray, N. V., editor, *Proceedings, International Conference on Automatic Reasoning with Analytic Tableaux and Related Methods, TABLEAUX'99, Saratoga Springs, NY, USA*, number 1617 in Lecture Notes in Artificial Intelligence, pages 24–28. Springer-Verlag, Berlin.

Haarslev, V. & Möller, R. (1999b). An empirical evaluation of optimization strategies for ABox reasoning in expressive description logics. In [Lambrix et al., 1999], pages 115–119.

Haarslev, V. & Möller, R. (1999c). Expressive ABox reasoning with number restrictions, role hierarchies, and transitively closed roles. Technical Report FBI-HH-M-288/99, University of Hamburg, Computer Science Department. Available at URL http://kogs-www.informatik.uni-hamburg.de/~haarslev/publications/report-FBI-288-99.ps.gz.

Haarslev, V. & Möller, R. (1999d). RACE system description. In [Lambrix et al., 1999], pages 130–132.

Haarslev, V. & Möller, R. (1999e). RACE System Download Page. Available at URL http://kogs-www.informatik.uni-hamburg.de/~race/.

Haarslev, V. & Möller, R. (2000a). Consistency testing: The RACE experience. In *Proceedings International Conference Tableaux'2000*, pages 57–61. Springer-Verlag.

Haarslev, V. & Möller, R. (2000b). Expressive ABox reasoning with number restrictions, role hierachies, and transitively closed roles. In [Cohn et al., 2000].

Haarslev, V. & Möller, R. (2000c). High performance reasoning with very large knowledge bases. In [Baader et al., 2000], pages 143–152.

Haarslev, V. & Möller, R. (2000d). Optimizing TBox and ABox reasoning with pseudo models. In [Baader et al., 2000], pages 153–162.

Haarslev, V., Möller, R., & Schröder, C. (1994). Combining spatial and terminological reasoning. In Nebel, B. & Dreschler-Fischer, L., editors, *KI-94: Advances in Artificial Intelligence – Proc. 18th German Annual Conference on Artificial Intelligence, Saarbrücken, Sept. 18–23, 1994*, volume 861 of *Lecture Notes in Artificial Intelligence*, pages 142–153. Springer-Verlag, Berlin.

Haarslev, V., Möller, R., & Turhan, A.-Y. (1999c). RACE user's guide and reference manual version 1.1. Technical Report FBI-HH-M-289/99, University of Hamburg, Computer Science Department. Available at URL http://kogs-www.informatik.uni-hamburg.de/~haarslev/publications/report-FBI-289-99.ps.gz.

Haarslev, V., Möller, R., & Wessel, M. (2000a). The description logic $\mathcal{ALCNH}_{R+}$ extended with concrete domains. Technical Report FBI-HH-M-290/00, University of Hamburg, Computer Science Department. Available at URL http://kogs-www.informatik.uni-hamburg.de/~haarslev/publications/report-FBI-290-00.ps.gz.

Haarslev, V., Möller, R., & Wessel, M. (2000b). *Diagrammatic Representation and Reasoning*, chapter Visual Spatial Query Languages: A Semantics Using Description Logic. Springer-Verlag.

Halpern, J. & Shoham, Y. (1991). A propositional model logic of time intervals. *Journal of the Association of Computing Machinery*, 38(4):935–962.

Hanschke, P. (1993). A declarative integration of terminological, constraint-based, data-driven and goal-directed reasoning. Technical Report DFKI-RR-93-46, German Center for AI (DFKI).

Hayes, P. (1977). In defense of logic. In *Proc. International Joint Conference on Artificial Intelligence*, pages 559–565.

Hayes, P. (1979). The logic of frames. In Metzing, D., editor, *Frame Conceptions and Text Understanding*, pages 46–61. De Gruyter and Co., Berlin. Reprinted in [Brachman & Levesque, 1985, 288–295].

Hoffmann, J. & Köhler, J. (1999). A new method to query and index sets. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence IJCAI-99*, pages 462–467. Morgan-Kaufmann Publishers.

Hollunder, B. (1994). *Algorithmic Foundations of Terminological Knowledge Representation Systems*. PhD thesis, University of Saarbrücken, Department of Computer Science.

Hollunder, B. & Baader, F. (1991). Qualifying number restrictions in concept languages. In Allen, J., Fikes, R., & Sandewall, E., editors, *Second International Conference on Principles of Knowledge Representation, Cambridge, Mass., April 22-25, 1991*, pages 335–346. A detailed version appeared as DFKI Research Report RR-91-03, Kaiserslautern.

Hollunder, B., Laux, A., Profitlich, H., & Trenz, T. (1991). $\mathcal{KRIS}$-manual. Technical report, Deutsches Forschungszentrum für Künstliche Intelligenz GmbH.

Hollunder, B. & Nutt, W. (1990). Subsumption algorithms for concept languages. In *Proc. of the 9th European Conference on Artificial Intelligence (ECAI'90), Stockholm, Sweden*. An extended version is published as DFKI Research Report DFKI-RR-90-04.

Horrocks, I. (1997). *Optimising Tableaux Decision Procedures for Description Logics*. PhD thesis, University of Manchester.

Horrocks, I. (1998). Using an expressive description logic: FaCT or fiction? In [Cohn et al., 1998], pages 636–647.

Horrocks, I. (1999). FaCT and iFaCT. In [Lambrix et al., 1999], pages 133–135. Also published as CEUR Workshop Proceedings, Volume 22, http://SunSITE.Informatik.RWTH-Aachen.DE/Publications/CEUR-WS/Vol-22/.

Horrocks, I. (2000). Benchmark analysis with FACT. In *Proceedings International Conference Tableaux'2000*, pages 62–65. Springer-Verlag.

Horrocks, I. & Patel-Schneider, P. (1998a). DL systems comparison. In *Proceedings of DL'98, International Workshop on Description Logics*, pages 55–57, Trento(Italy).

Horrocks, I. & Patel-Schneider, P. (1998b). FaCT and DLP: Automated reasoning with analytic tableaux and related methods. In *Proceedings International Conference Tableaux'98*, pages 27–30.

Horrocks, I. & Patel-Schneider, P. (1998c). Optimising propositional modal satisfiability for description logic subsumption. In *AISC'98, Artificial Intelligence and Symbolic Computation*, pages 234–246.

Horrocks, I. & Patel-Schneider, P. (1999). Optimising description logic subsumption. *Journal of Logic and Computation*, 9(3):267–293.

Horrocks, I. & Sattler, U. (1999). A description logic with transitive and inverse roles and role hierarchies. *Journal of Logic and Computation*, 9(3):385–410.

Horrocks, I., Sattler, U., Tessaris, S., & Tobies, S. (1999a). Query containment using a DLR ABox. LTCS-Report 99-15, LuFG Theoretical Computer Science, RWTH Aachen, Germany. See http://www-lti.informatik.rwth-aachen.de/Forschung/Reports.html.

Horrocks, I., Sattler, U., & Tobies, S. (1999b). Practical reasoning for expressive description logics. In Ganzinger, H., McAllester, D., & Voronkov, A., editors, *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705 in Lecture Notes in Artificial Intelligence, pages 161–180. Springer-Verlag.

Horrocks, I., Sattler, U., & Tobies, S. (2000). Reasoning with individuals for the description logic SHIQ. In MacAllester, D., editor, *Proceedings of the 17th International Conference on Automated Deduction (CADE-17)*, Lecture Notes in Computer Science, Germany. Springer Verlag.

Horrocks, I. & Tessaris, S. (2000). Answering conjunctive queries over dl aboxes: A preliminary report. In [Baader et al., 2000], pages 173–180.

Horrocks, I. & Tobies, S. (2000). Reasoning with axioms: Theory and practice. In Cohn, A. G., Giunchiglia, F., & Selman, B., editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Seventh International Conference (KR2000)*, pages 285–296, San Francisco, CA. Morgan Kaufmann Publishers.

Israel, D. & Brachman, R. (1984). Some remarks on the semantics of representation languages. In Brodie, M., Mylopoulos, J., & Schmidt, J., editors, *On Conceptual Modeling: Perspectives from Artificial Intelligence Databases and Programming Languages*. Springer-Verlag, New York.

Kaczmarek, T., Bates, R., & Robins, G. (1986). Recent developments in NIKL. In *Proc. of the Fifth National Conference on Artificial Intelligence, AAAI-86*, pages 578–587.

Kalmes, J. (1988). SB-Graph user manual. Technical report, University of the Saarland, Germany, Computer Science Department, SFB 314, Memo Nr. 30.

Kalmes, J. (1990). SB-Graph. Technical report, University of the Saarland, Germany, Computer Science Department, SFB 314, Memo Nr. 44. (in German).

Kaplunova, A. (1999). Inhaltsbasierter Informationszugriff durch die Berechnung von Konzeptgemeinsamkeiten in einer probabilistischen Beschreibungslogik (in German). Diploma Thesis (Diplomarbeit).

Kashyap, V. & Sheth, A. (1998). Semantic heterogeneity in global information systems: the role of metadata, context and ontologies. In *Cooperative Information Systems: Trends and Directions*. Academic Press.

Katoen, J.-P. (1999). *Concept, Algorithms, and Tools for Model Checking*. Arbeitsberichte des Instituts für Mathematische Maschinen und Datenverarbeitung (Informatik), Vol. 32, Number 1.

Kautz, H. A. & Ladkin, P. B. (1991). Integrating metric and qualitative temporal reasoning. In *Proc. of AAAI-91*, pages 241–246, Anaheim, CA.

Kietz, J. & Morik, K. (1994). A polynomial approach to the constructive induction of structural knowledge. *Machine Learning*, 14:193–217.

Kindermann, C. (1992). Retraction of object descriptions in BACK. Technical report, Technical University of Berlin, Computer Science Department, KIT Project Group, KIT-Report 105.

Kindermann, C. & Quantz, J. (1990). Implementation of the BACK system version 4. Technical report, Technical University of Berlin, Computer Science Department, KIT Project Group, KIT-Report 78.

Kindermann, C. & Randi, P. (1990). Object recognition and retrieval in the BACK system. Technical report, Technical University of Berlin, Computer Science Department, KIT Project Group, KIT-Report 86.

Klusch, M. (1998). *Cooperative Information Agents on the Internet (in German).* PhD thesis, University of Kiel.

Kobsa, A. (1991a). First experiences with the SB-ONE knowledge representation workbench in natural-language applications. *ACM SIGART Bulletin*, 2(3):70–76.

Kobsa, A. (1991b). Utilizing knowledge: The components of the sb-one knowledge representation workbench. In [Sowa, 1991], pages 457–486.

Koller, D., Levy, A., & Pfeffer, A. (1997). P-Classic: A tractable probabilistic description logic. In *Proc. of AAAI 97*, pages 390–397, Providence, Rhode Island.

Kuper, G., Libkin, L., & Paredaens, J. (2000). *Constraint Databases*. Springer.

Lambrix, P., Borgida, A., Lenzerini, M., Möller, R., & Patel-Schneider, P., editors (1999). *Proc. of the 1999 International Workshop on Description Logics*. Also published as CEUR Workshop Proceedings, Volume 22, http://SunSITE.Informatik.RWTH-Aachen.DE/Publications/CEUR-WS/Vol-22/.

Lambrix, P., Shahmehri, N., & Wahlöf, N. (1998). A default extension to description logics for use in an intelligent search engine. In *Proc. of the 1st Hawaiian Int. Conf. on System Science*.

Lambrix et al., P., editor (1999). *Proceedings of the International Workshop on Description Logics (DL'99), July 30 - August 1, 1999, Linköping, Sweden*.

Lenzerini, M. & Schaerf, A. (1991). Concept languages as query languages. In *Proceedings of the Ninth National Conference on Artificial Intelligence AAAI-91*, pages 471–476.

Levy, A., Rajaraman, A., & Ordille, J. (1996). Query-answering algorithms for information agents. In *Proceedings, AAAI'96, 14th National Conference on Artificial Intelligence*.

Lilius, J. & Paltor, I. P. (1999a). The production cell: An exercise in the formal verification of a UML model. Technical Report 288, Turku Centre for Computer Science.

Lilius, J. & Paltor, I. P. (1999b). The semantics of UML state machines. Technical Report 273, Turku Centre for Computer Science.

Lilius, J. & Paltor, I. P. (1999c). vUML: a tool for verifying UML models. Technical Report 272, Turku Centre for Computer Science.

Lipkis, T. (1981). A KL-ONE classifier. In [Schmolze & Brachman, 1982]. Published as BBN Research Report 4842, Bolt Beranek and Newman Inc., June 1982.

Lutz, C. (1998). Representation of Topological Information in Description Logics (in German). Master's thesis, University of Hamburg, Computer Science Department.

Lutz, C. (1999a). The complexity of reasoning with concrete domains (revised version). LTCS-Report 99-01, LuFG Theoretical Computer Science, RWTH Aachen, Germany. See http://www-lti.informatik.rwth-aachen.de/Forschung/Papers.html.

Lutz, C. (1999b). Reasoning with concrete domains. In Dean, T., editor, *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence IJCAI-99*, pages 90–95, Stockholm, Sweden. Morgan-Kaufmann Publishers.

Lutz, C., Haarslev, V., & Möller, R. (1997). A concept language with role-forming predicate restrictions. Technical Report FBI-HH-M-276/97, University of Hamburg, Computer Science Department.

Lutz, C. & Möller, R. (1997). Defined topological relations in description logics. In [Rousset et al., 1997], pages 15–19.

MacGregor, R. (1988). A deductive pattern matcher. In *Proc. of the Seventh National Conference on Artificial Intelligence, AAAI-88*, pages 403–408.

MacGregor, R. (1991a). The evolving technology of classification-based knowledge representation systems. In [Sowa, 1991], pages 385–400.

MacGregor, R. (1991b). Inside the LOOM description classifier. *ACM SIGART Bulletin*, 2(3).

MacGregor, R. (1994). A description classifier for the predicate calculus. In *Proc. of the Twelfth National Conference on Artificial Intelligence, AAAI-94*, pages 213–220.

MacGregor, R. & Bates, R. (1987). The LOOM knowledge representation language. Technical report, University of Southern California, Information Sciences Institute, Marina del Rey, Technical Report ISI/RS-87-188.

MacGregor, R. & Brill, D. (1992). Recognition algorithms for the Loom classfier. In *Proc. of AAAI'92, Thenth Conference on Artificial Intelligence*, pages 774–779, Menlo-Park, Cambridge, London. AAAI-Press/The MIT Press.

MacGregor, R. & Burstein, M. (1991). Using a description classifier to enhance knowledge representation. *IEEE Expert*, 6(3):41–47.

Mantay, T. & Möller, R. (1998). Content-based information retrieval by computation of least common subsumers in a probabilistic description logic. In Wache, H., Duschka, O., Fensel, D., Lenzerini, M., & Rousset, M.-C., editors, *Workshop on Intelligent Information Integration, ECAI-98*, pages 141–155, Brighton, England. Technical Report No. 10, Technologie-Zentrum Informatik, Universität Bremen, 1998.

Mantay, T., Möller, R., & Kaplunova, A. (1999). Computing probabilistic least common subsumers in description logics. In Burgard, W., Christaller, T., & Cremers, A., editors, *Proceedings KI-99, 23. Deutsche Jahrestagung für Künstliche Intelligenz*, pages 89–100. Springer-Verlag.

Mark, W. (1981). Realization. In [Schmolze & Brachman, 1982]. Published as BBN Research Report 4842, Bolt Beranek and Newman Inc., June 1982.

Mays, E., Dionne, R., & Weida, R. (1991a). K-Rep system overview. *ACM SIGART Bulletin*, 2(3):93–97.

Mays, E., Lanka, S., Dionne, B., & Weida, R. (1991b). A persistent store for large shared knowledge bases. *IEEE Transactions on Knowledge and Data Engineering*, 3(1):33–41.

McAllester, D. (1982). Reasoning utility package user's manual. Technical report, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, AI Memo 551.

McCray, A. & Nelson, S. (1995). The representation of meaning in the UMLS. *Methods of Information in Medicine*, 34(1/2):193–201.

McGuinness, D. (1996). *Explaining Reasoning in Description Logics*. PhD thesis, Graduate School–New Brunswick, Rutgers, The State University of New Jersey.

McGuinness, D. & Borgida, A. (1995). Explaining subsumption in description logics. In *Proc. of the 14th International Joint Conference on Artificial Intelligence, IJCAI-95, Montreal, Canada*, pages 816–821. Springer-Verlag, New York.

McGuinness, D. L. (1998). *Frontiers in Artificial Intelligence and Applications*, chapter Ontological Issues for Knowledge-Enhanced Search. IOS-Press, Washington, DC.

Meghini, C., Sebastiani, F., Straccia, U., & Thanos, C. (1993). A model of information retrieval based on a terminological logic. In *Proc. ACL SIGIR-93, Int. Conference on Research and Development in Information Retrieval, Pittsburg, PA*, pages 298–307.

Meghini, C. & Straccia, U. (1996). A relevance terminological logic for information retrieval. In *Proceedings of SIGIR*, pages 197–205.

Meiri, I. (1996). Combining qualitative and quantitative constraints in temporal reasoning. *Artificial Intelligence*, 87 (1-2):343–385.

Minsky, M. (1975). A framework for representing knowledge. In Winston, P., editor, *The Psychology of Computer Vision*, pages 211–277. McGraw Hill, New York. Reprinted in [Brachman & Levesque, 1985, 245–262].

Möller, R. & Haarslev, V. (2001). Description logic systems. In [Baader et al., 2001].

Möller, R., Haarslev, V., & Lutz, C. (1997). Spatioterminological reasoning based on geometric inferences: The $\mathcal{ALCRP}(\mathcal{D})$ approach. Technical Report FBI-HH-M-277/97, University of Hamburg, Computer Science Department.

Möller, R., Haarslev, V., & Neumann, B. (1998). Semantics-based Information Retrieval. In *Int. Conf. on Information Technology and Knowledge Systems*, Vienna, Budapest.

Möller, R., Haarslev, V., & Neumann, B. (2001). Expressive description logics for agent-based information retrieval. In *Knowledge Engineering and Agent Technology*. IOS Press.

Möller, R., Neumann, B., & Wessel, M. (1999a). Towards computer vision with description logics: Some recent progress. In *Proceedings Integration of Speech and Image Understanding, Corfu, Greece*, pages 101–115. IEEE Computer Society, Los Alamitos.

Möller, R. & Wessel, M. (1999). Terminological default reasoning about spatial information: A first step. In C. Freksa, D. M., editor, *Proc. International*

Conference on Spatial Information Theory, COSIT '99, Stade, Germany, pages 189–204. Springer-Verlag.

Möller, R., Wessel, M., Haarslev, V., & Turhan, A.-Y. (1999b). On terminological default reasoning about spatial information: Extended abstract. In [Lambrix et al., 1999], pages 155–159.

Nebel, B. (1988). Computational complexity of terminological reasoning in BACK. *Artificial Intelligence*, 34:371–383.

Nebel, B. (1990a). *Reasoning and Revision in Hybrid Representation Systems*, volume 422 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, Berlin.

Nebel, B. (1990b). Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43:235–249.

Nebel, B. (1991). Terminological cycles: Semantics and computational properties. In [Sowa, 1991], pages 331–361.

Nebel, B. (1995). Computational properties of qualitative spatial reasoning: First results. In Wachsmuth, I., Rollinger, C.-R., & Brauer, W., editors, *Proceedings, KI-95: Advances in Artificial Intelligence, 19th Annual German Conference on Artificial Intelligence, Bielefeld, Germany, Sept. 11-13*, volume 981 of *Lecture Notes in Artificial Intelligence*, pages 233–244. Springer-Verlag, Berlin.

Nebel, B., Rich, C., & Swartout, W., editors (1992). *Proceedings of Third International Conference on Principles of Knowledge Representation and Reasoning (KR'92)*. Morgan Kaufmann.

Nebel, B. & von Luck, K. (1987). Issues of integration and balancing in hybrid knowledge representation systems. In Morik, K., editor, *Proceedings GWAI-87*, pages 114–123. Springer-Verlag, Berlin.

Nebel, B. & von Luck, K. (1988). Hybrid reasoning in BACK. In Ras, Z. & Saitta, L., editors, *Methodologies for Intelligent Systems*, pages 260–269. North-Holland.

Neuwirth, A. (1993). Inferences for temporal object descriptions in a terminological representation system. Technical report, Technical University of Berlin, Computer Science Department, Project KIT, KIT-Report 107.

Newell, A. (1982). The knowledge level. *Artificial Intelligence*, 18:87–127.

Padgham, L. & Nebel, B. (1993). Combining classification and nonmonotonic inheritance reasoning: A first step. In Ras, Z. W. & Komorowski, J., editors, *Methodologies for Intelligent Systems (ISMIS'93)*, pages 132–141. Springer-Verlag.

Padgham, L. & Zhang, T. (1993). A terminological logic with defaults. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence IJCAI-93*, pages 662–668.

Page-Jones, M. (2000). *Fundamentals of object-oriented design in UML.* Addison-Wesley.

Papazoglou, M. & Schlageter, G. (1998). *Cooperative Information Systems: Trends and Directions.* Academic Press.

Patel-Schneider, P. (1984). Small can be beautiful in knowledge representation. In *Proceedings IEEE Workshop on Principle of Knowledge-Based Systems, Denver, CO.* Revised and Extended Version, AI Tech. Rept. No. 37, Schlumberger Palo Alto Research.

Patel-Schneider, P. (1986). A four-valued semantics for frame-based description languages. In *Proc. of the Fifth National Conference on Artificial Intelligence, AAAI-86*, pages 344–348.

Patel-Schneider, P. (1987a). *Decidable, Logic-Based Knowledge Representation.* PhD thesis, Department of Computer Science, University of Toronto, Ontario, Canada. Available as Technical Report 201/87.

Patel-Schneider, P. (1987b). A hybrid, decidable, logic-based knowledge representation system. *Computational Intelligence*, 3:64–77.

Patel-Schneider, P. (1989a). A four-valued semantics for terminological logics. *Artificial Intelligence*, 38:319–351.

Patel-Schneider, P. (1989b). Undecidability of subsumption in NIKL. *Artificial Intelligence*, 39:263–272.

Patel-Schneider, P. (1999). Performance of DLP on random modal formulae. In [Lambrix et al., 1999], pages 120–124. Also published as CEUR Workshop Proceedings, Volume 22, http://SunSITE.Informatik.RWTH-Aachen.DE/Publications/CEUR-WS/Vol-22/.

Patel-Schneider, P., McGuinness, D., Brachman, R., & Resnick, L. (1991). The CLASSIC knowledge representation system: Guiding principles and implementation rationale. *ACM SIGART Bulletin*, 2(3):108–113.

Patel-Schneider, P., Owsnicki-Klewe, B., Kobsa, A., Guarino, N., MacGregor, R., Mark, W., MacGuinness, D., Nebel, B., Schmiedel, A., & Yen, J. (1990). Term subsumption languages in knowledge representation. *AI Magazine*, 11(2):16–23.

Patel-Schneider, P. & Swartout, W. (1993). *Description-Logic Knowledge Representation System Specification (KRSS).* http://www-db.research.bell-labs.com/user/pfps/papers/krss-spec.ps.

Peltason, C. (1991). The BACK system – an overview. *ACM SIGART Bulletin*, 2(3):114–119.

Peltason, C., Schmiedel, A., Kindermann, C., & Quantz, J. (1989). The BACK system revisited. Technical report, Technical University of Berlin, Computer Science Department, KIT Project Group, KIT-Report 75.

Poole, D., Mackworth, A., & Goebel, R. (1998). *Computational Intelligence.* Oxford University Press.

Quantz, J., Dunker, G., Bergmann, F., & Kellner, I. (1995). The FLEX system. Technical report, Technical University of Berlin, Computer Science Department, Project KIT-VM11, KIT-Report 124.

Quantz, J. & Royer, V. (1992). A preference semantics for defaults in terminological logics. In [Nebel et al., 1992], pages 294–305.

Quillian, M. (1967). Word concepts: A theory and simulation of some basic semantic capabilities. *Behavioral Science*, 12:410–430. Reprinted in [Brachman & Levesque, 1985].

Quillian, M. (1968). Semantic memory. In Minsky, M., editor, *Semantic Information Processing*, pages 216–270. MIT Press, Cambridge, Mass.

Randell, D., Cui, Z., & Cohn, A. (1992). A spatial logic based on regions and connections. In Nebel, B., Rich, C., & Swartout, W., editors, *Principles of Knowledge Representation and Reasoning, Cambridge, Mass., Oct. 25-29, 1992*, pages 165–176. Morgan Kaufman.

Reiter, R. (1980). A logic for default reasoning. *Artificial Intelligence*, 13:81–132.

Renz, J. (1998). A canonical model of the region connection calculus. In [Cohn et al., 1998], pages 330–341.

Renz, J. & Nebel, B. (1997). On the complexity of qualitative spatial reasoning: A maximal tractable fragment of the region connection calculus. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI'97)*, pages 522–527.

Resnick, L., Borgida, A., Brachman, R., McGuinness, D., Patel-Schneider, P., & Zalondek, K. (1995). CLASSIC: Description and reference manual for the Common Lisp implementation, version 2.3. Technical report, AT&T Bell Labs, Murray Hill, NY.

Robins, G. (1986). The NIKL manual. Technical report, University of Southern California, Los Angeles, Information Sciences Institutes, The Knowledge Representation Project.

Rohatgi, V. K. (1976). *An Introduction to Probability Theory and Mathematical Statistics*. Wiley Series in Probability and Mathematical Statistics.

Rosati, R. (1997). Autoepistemic description logics. Technical Report PhD thesis, IX-97-6, Dipartimento di Informatica e Sistemistica, Universita Degli Study di Roma "La Sapienza".

Rousset et al., M.-C., editor (1997). *Proceedings of the International Workshop on Description Logics, DL'97, Sep. 27-29, 1997, Gif sur Yvette, France*. Universite Paris-Sud, Paris.

Sattler, U. (1996). A concept language extended with different kinds of transitive roles. In Görz, G. & Hölldobler, S., editors, *20. Deutsche Jahrestagung für Künstliche Intelligenz*, number 1137 in Lecture Notes in Artificial Intelligence, pages 333–345. Springer-Verlag, Berlin.

Schaerf, A. (1994). *Query Answering in Concept-Based Knowledge Representation Systems: Algorithms, Complexity, and Semantic Issues*. PhD thesis, Dipartimento di Informatica e Sistemistica, Universita di Roma "La Sapienza", Via Salaria 113, I-00198 Roma, Italy.

Schank, R. (1975). *Conceptual Information Processing*. North Holland, Amsterdam.

Schaub, T. (1997). *The Automation of Reasoning with Incomplete Information - From Semantic Foundations to Efficient Computation*. Lecture Notes in Artificial Intelligence 1409. Springer-Verlag, Berlin.

Schild, K. (1991a). A correspondence theory for terminological logics: Preliminary report. In *Twelfth International Conference on Artificial Intelligence, Darling Harbour, Sydney, Australia, Aug. 24-30, 1991*, pages 466–471.

Schild, K. (1991b). A tense-logical extension of terminological logics. Technical report, Technical University of Berlin, Computer Science Department, Project KIT, KIT-Report 92.

Schild, K. (1993). Combining terminological logics with tense logic. In Filgueiras, M. & Damas, L., editors, *Progress in Artificial Intelligence, 6th Portuguese Conference on AI, EPIA '93*, volume 727 of *LNAI*, pages 105–120, Porto, Portugal. Springer-Verlag 1993.

Schmidt, R. (1991). Algebraic terminological representation. Technical report, Max Planck Institute for Computer Science, MPI-Report MPI-I-91-216.

Schmidt-Schauss, M. (1989). Subsumption in KL-ONE is undecidable. In [Brachman et al., 1989], pages 421–431.

Schmidt-Schauss, M. & Smolka, G. (1991). Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26.

Schmiedel, A. (1988). A temporal constraint handler for the BACK system. Technical report, Technical University of Berlin, Computer Science Department, Project KIT, KIT-Report 70.

Schmiedel, A. (1990). A temporal terminological logic. In *Proc. of the Ninth National Conference on Artificial Intelligence, AAAI-90*, pages 640–645.

Schmiedel, A. (1993). Persistent maintenance of object descriptions using BACK. Technical report, Technical University of Berlin, Computer Science Department, KIT Project Group, KIT-Report 112.

Schmolze, J. (1985). The language and semantics of NIKL. Technical report, BBN Laboratories, Cambridge, MA.

Schmolze, J. (1989). Terminological knowledge representation systems supporting n-ary terms. In [Brachman et al., 1989], pages 432–443.

Schmolze, J. & Brachman, R., editors (1982). *Proceedings of 1981 KL-ONE Workshop*. Published as BBN Research Report 4842, Bolt Beranek and Newman Inc., June 1982.

Schmolze, J. & Israel, D. (1983). KL-ONE: Semantics and classification. Technical report, Research in Knowledge Representation for Natural Language Understanding - Annual Report 1983, BBN Report No. 5421, BBN Laboratories, Camb., MA.

Schmolze, J. & Lipkis, T. (1983). Classification in the KL-ONE knowledge representation system. In *Proc. Eigth International Conference on Artificial Intelligence, IJCAI-83, Karlsruhe, Germany*, pages 330–332. Morgan Kaufmann.

Schmolze, J. & Mark, W. (1991). The NIKL experience. *Computational Intelligence*, 7(1):48–69.

Schröder, C., Möller, R., & Lutz, C. (1996). A partial logical reconstruction of PLAKON/KONWERK. In Baader, F., Bürckert, H.-J., Günter, A., & Nutt, W., editors, *Proceedings of the Workshop on Knowledge Representation and Configuration WRKP'96*, number D-96-04 in DFKI-Memos, pages 55–64.

Schulz, S. & Hahn, U. (2000). Knowledge engineering by large-scale knowledge reuse – Experience from the medical domain. In [Cohn et al., 2000], pages 601–610.

Selman, B., Levesque, H., & Mitchell, D. (1992). A new method for solving hard satisfiability problems. In *Proc. of AAAI'92*, pages 440–446.

Sowa, J., editor (1991). *Principles of Semantic Networks: Explorations in the Representation of Knowledge*. Morgan Kaufmann Publishers, San Mateo.

Stallman, R. & Sussman, G. (1977). Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis. *Artificial Intelligence*, 9(2):135–196.

Stickel, M. (1982). A nonclausal connection-graph resolution theorem-proving program. In *Proc. of the Third National Conference on Artificial Intelligence, AAAI-82*, pages 229–233.

Stock, O., editor (1997). *Spatial and Temporal Reasoning*. Kluwer Academic Publishers, Dordrecht.

Straccia, U. (1998). A Fuzzy Description Logic. In *Proc. AAAI-98*, Wisconsin.

Tarski, A. (1951). *A Decision Method for Elementary Algebra and Geometry*. University of California Press, Berkeley, CA.

Tobies, S. (2000). The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *Journal of Artificial Intelligence Research*, 12:199–217.

Tresp, C. B. & Molitor, R. (1998). A Description Logic for Vague Knowledge. Technical Report LTCS-Report 98-01, LuFg Theoretische Informatik, RWTH Aachen.

Truemper, K. (1998). *Effective Logic Computation*. John Wiley & Sons.

Turhan, A.-Y. (1998). Design and implementation of description logic provers for $\mathcal{ALC}(\mathcal{D})$ and $\mathcal{ALCRP}(\mathcal{D})$ (in german). Bachelors Thesis (Studienarbeit).

Turhan, A.-Y. (2000). Optimierungsmethoden für den Erfüllbarkeitstest bei Beschreibungslogiken mit konkreter Domäne (in German). Diploma Thesis (Diplomarbeit).

Turhan, A.-Y. & Haarslev, V. (2000). Adapting optimization techniques to description logics with concrete domains. In [Baader et al., 2000], pages 247–256.

Vilain, M. (1985). The restricted language architecture of a hybrid representation system. In *Proc. Ninth International Conference on Artificial Intelligence, IJCAI-85, Los Angeles, CA*. Morgan Kaufmann.

von Luck, K., Nebel, B., Peltason, C., & Schmiedel, A. (1987). The BACK system revisited. Technical report, Technical University of Berlin, Computer Science Department, KIT Project Group, KIT-Report.

Wahlöf, N. (1996). A default extension to description logics and its applications. Master's thesis, Linköping University, Thesis 591, Department of Computer and Information Science.

Weida, R. (1996). Closed terminologies in description logics. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence AAAI-96*, pages 592–599.

Wessel, M., Haarslev, V., & Möller, R. (2000). $\mathcal{ALC}_{RA}$ - $\mathcal{ALC}$ with role axioms. In [Baader et al., 2000], pages 267–276.

Woods, W. (1975). What's in a link: Foundations for semantic networks. In Bobrow & Collins, editors, *Representation and Understanding: Studies in Cognitive Science*, pages 157–209. Academic Press, New York. Reprinted in [Brachman & Levesque, 1985, 141–168].

Woods, W. (1991). Understanding subsumption and taxonomy: A framework in progress. In [Sowa, 1991], pages 45–94.

Woods, W. & Schmolze, J. (1992). The KL-ONE family. In Lehmann, F., editor, *Semantic Networks in Artificial Intelligence*, pages 133–177. Pergamon Press, Oxford.

Wright, J., Weixelbaum, E., Vesonder, G., Brown, K., Palmer, S., Berman, J., & Moore, H. (1993). A knowledge-based configurator that supports sales, engineering, and manufacturing at AT&T network systems. *AI Magazine*, 14(3):69–80.

Yen, J., Juang, H.-L., & MacGregor, R. (1991a). Using polymorphism to improve expert system maintainability. *IEEE Expert*, 6(2):48–55.

Yen, J., Neches, R., & MacGregor, R. (1991b). CLASP: Integrating term subsumption systems and productions systems. *IEEE Transactions on Knowledge and Data Engineering*, 3(1):25–32.

Zaniold, C., Ceri, S., Faloutsos, C., Snodgrass, R., Subrahmanian, V., & Zicari, R. (1997). *Advanced Database Systems*. Springer-Verlag, Berlin.